

INTEGRATED IMAGE AND GRAPHICS TECHNOLOGIES

David Zhang
Mohamed Kamel
George Baciu

Kluwer Academic Publishers

INTEGRATED IMAGE AND GRAPHICS TECHNOLOGIES

**THE KLUWER INTERNATIONAL SERIES IN
ENGINEERING AND COMPUTER SCIENCE**

INTEGRATED IMAGE AND GRAPHICS TECHNOLOGIES

edited by

David D. Zhang

*Hong Kong Polytechnic University
Hong Kong*

Mohamed Kamel

*University of Waterloo
Canada*

George Baci

*Hong Kong Polytechnic University
Hong Kong*

KLUWER ACADEMIC PUBLISHERS

NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

TEAM LinG - Live, Informative, Non-cost and Genuine!

eBook ISBN: 1-4020-7775-0
Print ISBN: 1-4020-7774-2

©2004 Kluwer Academic Publishers
New York, Boston, Dordrecht, London, Moscow

Print ©2004 Kluwer Academic Publishers
Boston

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Kluwer Online at: <http://kluweronline.com>
and Kluwer's eBookstore at: <http://ebooks.kluweronline.com>

TEAM LinG - Live, Informative, Non-cost and Genuine!

CONTENTS

PREFACE	<i>xiii</i>
----------------	-------------

Chapter 1

INTRODUCTION	1
---------------------	----------

David Zhang, Mohamed Kamel and George Baci

1.1	Image and Graphics Technologies	1
1.2	Integrated Technologies	2
1.3	Book Perspective	4

Chapter 2

TRANSFORMING 3D MESH SURFACES INTO IMAGES BY PARAMETERIZATION	11
--	-----------

Jingqi Yan, Pengfei Shi and David Zhang

2.1	Introduction	11
2.2	Chart Parameterization	12
2.3	Transforming Meshes Into Images	20
2.4	Applications	23
2.5	Conclusion	27

Chapter 3

3D MODELING BASED ON ATTRIBUTED HYPERGRAPHS	31
--	-----------

Li Rong and Andrew K.C. Wong

3.1	Introduction	31
3.2	Attributed Hypergraph Representation (AHR)	33
3.3	3D Object Modeling Using AHR and AH Operators	42
3.4	Augmented Reality Using AHR	47
3.5	Experiments of Modeling and Augmented Reality	50

3.6	Conclusions	54
-----	-------------	----

Chapter 4

VISIBILITY CULLING FOR INTERACTIVE DYNAMIC SCENES	57
--	-----------

George Baciu and Ki-Wan Kwok

4.1	Introduction	57
4.2	Overview	60
4.3	Ray Parameterization	62
4.4	Visibility within a Vertical Directional Plane	63
4.5	Visibility Culling on Static Objects	66
4.6	Dynamic Scene Occlusion Culling	67
4.7	Conclusion	69

Chapter 5

IMAGE-BASED COLLISION DETECTION	75
--	-----------

George Baciu and Wingo Sai-Keung Wong

5.1	Introduction	75
5.2	Simulation Space	79
5.3	Object Space vs. Image Space Collision Detection	81
5.4	Ray Casting	83
5.5	Rendering Passes	85
5.6	Interference Region	87
5.7	Optimal MOR's	87

Chapter 6

FOURIER PROCESSING IN THE GRAPHICS PIPELINE	95
--	-----------

Edward Angel and Kenneth Moreland

6.1	Introduction	95
6.2	Convolution	96
6.3	Hardware Implementation	97
6.4	The Fourier Transform	98

6.5	Vertex and Fragment Programming	103
6.6	Using the GPU for the FFT	106
6.7	Examples	107
6.8	Conclusions	108

Chapter 7

TRANSFORMATION IMAGE INTO GRAPHICS 111

Zonghua Zhang, Xiang Peng and David Zhang

7.1	Introduction	111
7.2	Overviews	113
7.3	An Example System Based on Fringe Projection	119
7.4	Experimental Results	124
7.5	Conclusion Remarks and Future Work	126

Chapter 8

AN INTRODUCTION TO IMAGE-BASED RENDERING 131

Heung Yeung Shum, Yin Li and Sing Bing Kang

8.1	Introduction	131
8.2	Rendering with No Geometry	133
8.3	Rendering with Implicit Geometry	139
8.4	Rendering with Explicit Geometry	141
8.5	Trade-off between Images and Geometry	143
8.6	Rendering with Layered Geometry	148
8.7	Discussions	153

Chapter 9

IMAGE-BASED RELIGHTING: REPRESENTATION AND COMPRESSION 161

Tien-Tsin Wong and Pheng-Ann Heng

9.1	Introduction	161
9.2	Computational Model	162
9.3	Sampling	164
9.4	Relighting	165

9.5	Intra-Pixel Compression	168
9.6	Inter-Pixel Compression	171
9.7	Inter-Channel Compression	173
9.8	Overall Evaluation	175
9.9	Conclusions and Future Directions	177

Chapter 10

CONSTRUCTION OF COMPLEX ENVIRONMENTS FROM A SET OF DEPTH IMAGES **181**

Enhua Wu, Yanci Zhang and Xuehui Liu

10.1	Introduction	181
10.2	Typical Algorithms	182
10.3	Framework of Hybrid Modeling	183
10.4	Pick up of Valid Samples	184
10.5	Hybrid Representation	187
10.6	Real Time Rendering	191
10.7	Summary and Conclusions	193

Chapter 11

QUADRATIC SURFACE RECONSTRUCTION FROM MULTIPLE VIEWS USING SQP **197**

Rubin Gong and Gang Xu

11.1	Introduction	197
11.2	Formulation	199
11.3	Sequential Quadratic Programming	201
11.4	Outline of the Method	206
11.5	Experimental Results	206
11.6	Summary and Conclusions	214

Chapter 12

PHOTO-REALISTIC CONVERSATION AGENT **219**

Bo Zhang, Zicheng Liu and Baining Guo

12.1	Introduction	219
------	--------------	-----

12.2	E-Partner System Architecture	222
12.3	Facial Animation	223
12.4	Geometry-driven Expression Synthesis	229
12.5	Conclusion and Future Work	238

Chapter 13

3D SEISMIC VOLUME VISUALIZATION 241

Chunyu Ma and Jon Rokne

13.1	Introduction	241
13.2	Seismic Exploration	242
13.3	Visualizing Volume Data and Computer Graphics	246
13.4	Conclusion	259

Chapter 14

GRAPHICAL REPRESENTATION OF FINGERPRINT IMAGES 263

Jie Zhou, David Zhang, Jinwei Gu and Nannan Wu

14.1	Introduction	263
14.2	Minutiae-based Representation Fingerprint	265
14.3	Modeling Orientation Fields	267
14.4	Generation of Synthetic Fingerprint Images	276
14.5	Complete Representation of Fingerprints	278
14.6	Summary	280

Chapter 15

IMAGE BASED MODELING AND ANALYSIS OF TEXTILE MATERIALS 283

Jinlian Hu and Binjie Xin

15.1	Introduction	283
15.2	Modeling of Fabric Pilling	284
15.3	Modeling of Polar Fleece Fabric Appearance	291
15.4	Modeling of Fabric Wrinkling	297
15.5	Summary and Conclusions	304

Chapter 16

VIRTUAL PRODUCT PRESENTATION BASED ON IMAGES AND GRAPHICS 309

Zhigeng Pan, Mingmin Zhang and Tian Chen

16.1	Introduction	309
16.2	Related Work	311
16.3	Image-based Virtual Presentation of Products	316
16.4	Graphics-based Virtual Presentation of Products	322
16.5	Summary and Conclusions	325

Chapter 17

3D IMAGING AND APPLICATIONS 331

Xiaoyi Jiang and Hanspeter Bieri

17.1	Introduction	331
17.2	3D Imaging Principles	332
17.3	3D Sensors	335
17.4	Data Fusion	339
17.5	Model Generation	344
17.6	Other Related Topics	344
17.7	Applications	346
17.8	Conclusions	346

Chapter 18

RANGE IMAGE SEGMENTATION USING LOCAL APPROXIMATION OF SCAN LINES WITH APPLICATION TO CAD MODEL ACQUISITION 351

Inas Khalifa, Medhat Moussa and Mohamed Kamel

18.1	Introduction	351
18.2	Range Image Segmentation	353
18.3	CAD Model Building	359
18.4	Experimental Results	361
18.5	Conclusion and Future Work	369

Chapter 19**SHAPE-BASED IMAGE RETRIEVAL APPLIED TO TRADEMARK IMAGES 373***Ossama El Badawy and Mohamed Kamel*

19.1	Introduction	373
19.2	Overview of Current Methods	376
19.3	Shape Analysis	377
19.4	Shape Retrieval Architecture	380
19.5	Experimental Results	386
19.6	Conclusion and Future Work	391

Chapter 20**MULTI-RESOLUTION IMAGE REGISTRATION USING MULTI-CLASS HAUSDORFF FRACTION 393***Haikel Salem Alhichri and Mohamed Kamel*

20.1	Introduction	393
20.2	Hausdorff Fraction	395
20.3	Transformation Space	396
20.4	Multi-resolution Image Registration	396
20.5	Multi-class Hausdorff Fraction	398
20.6	Experimental Results	399
20.7	Conclusion	400

INDEX 407

This page intentionally left blank

PREFACE

The ultimate goal of image technology is to seamlessly perform visual functions equivalent to the recognition and the reconstruction power of living beings. Computer graphics has been mainly driven by engineering design processes and has established itself as a dominating methodology. Traditionally, image technology and computer graphics technology have been concerned with different goals. In computer graphics, computers are used to create pictures, animations and simulations. Image technology, on the other hand, consists of techniques and methodologies that modify or interpret existing pictures. Many methods proposed and used in these two areas often overlap and cross-fertilization between them can impact their progress. In the past, image technology and computer graphics have been typically combined in subtle ways, mainly in applications. In fact, the convergence of image processing and computer graphics has become the main research stream in both the computer graphics community as well as in the computer vision and image processing community. The image, vision and graphics research streams culminate with the interactive fusion of digital image and computer graphics. Therefore, it is useful to study approaches and methodologies that foster the integration of image and graphics technologies. This will provide the background and inspiration for some new creative methods or techniques.

This book provides a collection of 20 chapters containing tutorial articles and applications, in a unified way, the basic concepts, theories and characteristic features of integrating different facets of Image and Graphics, with recent developments and significant applications. The articles, written by recognized international experts, demonstrate the various ways in which this integration can be made possible in order to design methodologies and their applications efficiently. With the exception of the first chapter that serves as an introduction to image and graphics, each chapter provides detailed technical analysis of the development in the respective area, keeping a cohesive character with other chapters. Although there is an extensive coverage of problems and solutions that make the integration of graphics and image technologies more practical, it is generally difficult to compile in one volume all the possible techniques and design issues that arise in the multitude of application domains. We have attempted as much as possible to incorporate three main streams in this book: (1) From graphics to image, (2) From image to graphics, and (3) Applications of image and graphics integration.

The book, which is unique in its characters, is useful to graduate students and researchers in computer science, electrical engineering, systems science, and information technology not only as a reference book, but also as a textbook for some parts of the curriculum of courses in image processing and graphics. Researchers and practitioners in industry and R&D laboratories working in the fields of image processing, computer vision and graphics, system design, pattern recognition will also benefit from the new perspectives and techniques described in the book.

We take this opportunity to thank all the authors for agreeing to contribute chapters for the book. We owe a vote of thanks to Susan Lagerstrom-Fife and Sharon Palleschi of Kluwer Academic Publisher, USA, for taking the initiative in bringing the volume out. The technical/software support provided by Martin Kyle and Lily Yu is also acknowledged.

David Zhang
Mohamed Kamel
George Baci

November 2003

Chapter 1

INTRODUCTION

David Zhang ¹, Mohamed Kamel ² and George Baciú ¹

1. Department of Computing, The Hong Kong Polytechnic University, Hong Kong

2. Department of Systems Design, University of Waterloo, Ontario N2L 3G1, Canada

{csdzhang, csgeorge}@comp.polyu.edu.hk, mkamel@uwaterloo.ca

1.1 Image and Graphics Technologies

Image is visual communication and graphics is the art of communicating beyond words. From the Egyptian hieroglyphics to the CAD systems used today in aerospace, architecture, mechanical and VLSI designs, images have enhanced the acuity of our perception of 3D space and have reinforced concepts and associations within highly complex objects, while drawings have enhanced the creative process of shape design, color and reflections. Combining the two processes within the hyper-grid of computational display processors has become the architectural challenge of our era. The evolution of imagery is culminating in the interleaving of the two basic tasks of visual communication: image processing and graphic design. Computer generated imagery has become so pervasive that it is in fact expected rather than appreciated. However, new challenges continue in the simulation of new physical phenomena and the faithful reproduction of panoramas. As the wheel of technical innovation turns, image and graphics techniques evolve out of our passion to experience new visual dimensions.

1.1.1 Graphics Technology

While the computer graphics technology has often been the choice for preproduction and special effects in film and animated features, currently the exposure to digital graphics comes mostly from 3D computer games. These have become a new form of interactive art and communication for all ages. Graphics technology is now propelled by both photo- and non-photo- realism in technical design, art and animation.

Triangles have been rooted as the basic rendering primitive for 3D objects. Hence, in the last two decades, the focus of graphics hardware designers has been

mainly on satisfying the demand for increasing the triangle rendering rates and decreasing the triangle size. This has created a threshold paradox beyond which the ever increasing triangle resolution does not add to the visual quality of micro-features but rather it contributes strictly to the degradation of fill rates. In essence, as the median triangle size decreases below the pixel resolution the return on our polygon investment for the visible detail and realism diminishes drastically at the cost of the fill rate performance. The problem of polygon contention at the pixel level again leads to addressing the problem of resolving the visual quality issues at the image level by supporting fast anti-aliasing and complex shading in hardware.

For the next level of image quality enhancement, the solutions to the current problems will converge to non-uniform supersampling [1] and multi-pass rendering algorithms [6], as shown for example in collision detection applications [2]. Further enhancements can be made available with programmable mask patterns resolving, for example, anisotropic texture filtering, depth of field, and motion blur.

1.1.2 Image Technology

The image capturing and processing technology has been around for a number of years. The convergence of *digital image processing* and *computer vision* has transformed the imaging technology from low level binary representation and color reproduction and enhancement to more complex shape identification and understanding for automatic decision making purposes. Driven by the demands in robotics and machine learning, computer vision has focused on the detection of features and structures in complex 2D and 3D representations of objects. The very belief that machines may at some point process data in similar or equivalent processes as the human visual system has lead to numerous open problems. The remaining challenges continue to persist in resolution, compression, structural encoding and segmentation. More recently the concentration of the research work in image analysis has been the image reassembly and composition from multiple sources. The range of work extends from low level filtering, anti-aliasing and blurring techniques to panoramic synthesis, depth of field, video wall image tiling, dynamic illumination, shape extraction and blending. Early attempts to scalable hardware-based image composition resulted in the massively parallel PixelFlow depth-composition system [11].

The sophistication in image-based rendering techniques has climaxed in the attempt to resolve lighting constraints in image composition [5,4,14].

1.2. Integrated Technologies

In most areas of science and engineering as well as in entertainment and education, it seems that the main driver of new visualization, interaction and

communication applications is hybrid integration of the three main fields: image processing, computer graphics and computer vision [8]. For example, large interactive displays are beginning to appear in commercial applications for both image-based queries as well as for visualizing product designs [16,17]. Furthermore, virtual 2D and 3D devices such as projection keyboards [18] are beginning to attract consumers as the size of computational devices such as Palm organizers are getting increasingly difficult to operate with high accuracy and speed. In all these applications and many others, one area of study by itself is not sufficient any longer. Digital image processing, computer vision and computer graphics must work in concert in order to provide designers, decision makers and consumers with better qualitative information.

Both image processing and computer graphics have contributed most significantly towards the enhancement of computer hardware architecture and data transmission performance. High-end architectures with advanced features could best be demonstrated by benchmarking a combination of image tools and interactive visual applications. This has also pushed the development costs for hardware systems. Currently, the popularity of visual applications has grown tremendously due to the new video and graphics features with lower cost penetration into the home entertainment markets. However, the significant cost constraints on products for the consumer market restrain the interfaces and levels of performance required by the higher end visual design and simulation in application fields such as scientific, medical, manufacturing, and other industrial forms of engineering design.

One of the recent attempts to meet the growing requirements of interactive 3D applications is the design of graphics systems supported by parallel rendering chips. An example of such a recent system is the SAGE (Scalable Advanced Graphics Environment) architecture demonstrated by Deering [6]. Other attempts to build systems out of arrays of inexpensive game chips by Stoll et al [13], were initially not designed to be clustered, and therefore the resulting systems were less effective for the highly complex engineering simulation applications. The performance of a single SAGE board has been benchmarked to render in excess of 80 million fully lit, textured, anti-aliased triangles per second. The architecture has also been remarkable in the support for basic image processing techniques such as high quality anti-aliasing filters running at video rates for the first time. This has been achieved by replacing the concept of a frame buffer by a fully doubled buffered sample buffer that stores between 1 and 16 non-uniformly placed samples per final output pixel. Furthermore, the raster of samples is subject to convolution by a 5-5 programmable reconstruction and bandpass filter that replaced the traditional RAMDAC. The convolution operation during rasterization at video rates with a delay of less than one additional scanline of latency while the reconstruction filter processes up to 400 samples per output pixel, with full support

for any radially symmetric filter including Mitchell-Netravali filters. More importantly, the SAGE architecture supports scalable tiling for additional fill rates, resolutions and performance.

The proactive research in hardware design for high-performance imaging and graphics emphasizes once again the commercial requirements of combining graphics and image techniques. This has also been reflected in the high performance architectures realized in the NVIDIA's GeForce chip sets, Sony Playstation, ATI's Radeon and many other experimental and commercial systems.

All these developments point to one of the primary needs for quality real-time antialiasing within the rendering pipeline. This requires massive supersampling with large area resampling and bandpass filters all of which lead to the integration of image and graphics techniques closer to the hardware level. It is indeed the joint use of all these areas at multiple levels of computational processing, both hardware and software, that holds the greatest potential for future applications. In this book, we have attempted to bring together some of the fundamental issues in the current integration of image and graphics technologies and recent applications that contribute presently to the evolution of visual communication.

These fields are in the process of a natural convergence with an abundance of research material being disseminated every year. Although there is an extensive coverage of problems that is difficult to compile altogether under one volume, we tried our best to incorporate some recent representative work that falls into one of the three streams: (1) From graphics to image, (2) From image to graphics, and (3) Applications of image and graphics integration.

1.3. Book Perspective

This book embraces the explosive growth in the hybridization of image generation stemming from the combination of image processing, computer vision and computer graphics rendering. The quest has always been the visual understanding of concepts, structures and design, Figure 1.1.

1.3.1 From Graphics to Images

The graphics-to-image stream has been the most natural progression in the evolution of graphics technology. The three-dimensional modeling and rendering techniques have always been subject to a series of transformations from 3D to 2D in order to capture the visible parts of a three-dimensional scene. As shown in the chapters compiled in this section, this data stream is no longer limited to a unidirectional flow from graphical modeling and design to binary image synthesis. Currently, the image resolution is often used in the feedback loop to the graphics modeling and animation systems for level of detail, rendering quality and object

interference detection. The chapters compiled for this part of the book add a new dimension of data streaming that allows for better control of the image formation.

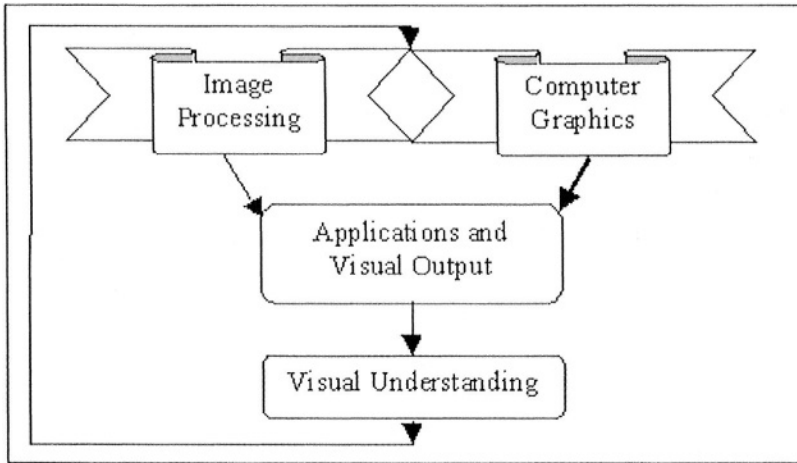


Figure 1.1. Image and graphics technologies.

Part I of this book is dedicated to the transformations required in the data stream from 3D models to the formation of a 2D digital image. In Chapter 2, Jingqi Yan, Pengfei Shi and David Zhang will first address the problem of mesh parameterization for transforming a 3D mesh into images. The chapter provides a review of several classic and popular parameterization algorithms based on solving the linear or non-linear systems, for instance, barycentric mapping, conformal mapping, harmonic mapping, and geometrical stretch minimizing. The chapter also introduces some recent methods for opening an arbitrary mesh into a topological disk by cutting along an appropriate set of edges for partitioning it into several charts. These techniques are applied to texture mapping, remeshing, and mesh compression.

In Chapter 3, Li Rong and Andrew K.C. Wong introduce a new mathematical framework for 3D computer vision based on the attributed hypergraph representation (AHR). This model incorporates the data structure and algorithms that build, manipulate and transform augmented AHRs and demonstrate the integration of machine vision and computer graphics.

In Chapter 4, George Baciuc and Kwok Ki Wan investigate recent occlusion culling methods that compute the potential visible feature sets for large dynamic outdoor scenes at interactive rates. The new methods use occlusion maps on a dual ray-space to encode the visible sets with respect to a view cell and utilize the

new features of the advanced graphics hardware in order to construct and maintain the occlusion maps.

Chapter 5, George Baciú and Wong Sai-Keung discuss new developments in collision detection that combine both object space culling and image-space interference tests in order to provide a more balanced computational load for interactive animation and motion response in 3D virtual environments and interactive games. Complex object interactions between rigid and deformable objects of arbitrary surface complexity can be attained at interactive rates.

In Chapter 6, Edward Angel and Kenneth Moreland address the processing of the data at the lower end of the graphics-to-image stream and provide an implementation of the Fast Fourier Transform on a GPU by exploiting the new features of the recent graphics hardware that is able to carry out floating point operations with large arrays on Graphics Processing Units (GPUs).

1.3.2 From Images to Graphics

Part II of this book is dedicated to methods and techniques that provide solutions for the inverse data stream from images to 3D graphics. The image-to-graphics stream is based on driving graphics applications from images and video sequences. The most exemplary topic of this stream is image-based rendering which is covered extensively in this part of the book.

In Chapter 7, Zonghua Zhang, Xiang Peng and David Zhang begin with an overview of state-of-the-art range image acquisition methods. The chapter provides an introduction to the construction of complete 3D graphical models using multi-view range image registration and integration. It also presents a novel color texture acquisition method and discusses popular approaches to the geometric description of 3D graphical models resulting from range image registration and integration.

Image-based rendering (IBR) techniques have shown to be highly effective in using images and geometry for the representation of 3D scenes. In Chapter 8, Heung Yeung Shum, Yin Li and Sing Bing Kang explore the issues in trading off the use of images and geometry by revisiting plenoptic sampling analysis and the notions of view dependency and geometric proxies. Furthermore, they introduce a practical IBR technique called *pop-up light field* which models a sparse light field by a set of *coherent layers*. These layers incorporate both color and matting information, and allow rendering in real time without introducing any aliasing.

The IBR theme is continued in Chapter 9. Tien-Tsin Wong and Pheng-Ann Heng introduce illumination control in IBR by modifying the plenoptic function into a new function called the *plenoptic illumination function*. This function explicitly specifies the illumination component of image representations in order to support relighting and view interpolation. Furthermore, new compression

techniques for IBR data based on *intra-pixel*, *inter-pixel* and the *inter-channel* correlations are presented.

In Chapter 10, Enhua Wu and Yanci Zhang discuss the research issues and state-of-the-art of construction of complex environments from a set of depth images. A new automatic construction algorithm is introduced in this chapter. The algorithm uses a hybrid representation of complex models by a combination of points and polygons achieving the real-time walkthrough of a complex 3D scene.

In Chapter 11, Gang Xu and Rubin Gong propose using SQP (Sequential Quadratic Programming) to directly recover 3D quadratic surface parameters from multiple views. A surface equation is used as a constraint. In addition to the sum of squared reprojection errors defined in the traditional bundle adjustment, a Lagrangian term is added to force recovered points to satisfy the constraint. The recovered quadratic surface model can be represented by a much smaller number of parameters than point clouds and triangular patches.

In Chapter 12, Bo Zhang, Zicheng Liu, and Baining Guo introduce image-based facial animation (IBFA) techniques. These allow for rapid construction of photo-realistic talking heads. The authors address the main issues in creating a conversation agent using IBFA, the appearance of a conversation agent and lip-synching for different languages. The techniques are demonstrated on an English speaker and a Mandarin Chinese speaker, in the E-Partner system.

1.3.3 Systems and Applications

In Chapter 13, Jon Rockne discusses methods for 3D visualization of data gathered from seismic exploration programs after it has been processed into a 3D seismic datavolume. After a general overview of 3D volume visualization is provided, the main problem addressed is the identification of subsurface structures of interest.

In Chapter 14, Jie Zhou, David Zhang, Jinwei Gu, and Nannan focus on research issues in the graphical representation of fingerprints. The authors introduce a minutiae-based representation and provide models for the graphical representation of orientation fields. The problem of the generation of synthetic fingerprint images is addressed and a complete fingerprint representation is introduced.

In Chapter 15, Jinlian Hu and Binjie Xin analyze and model the fabric appearance using image-based techniques and state-of-the-art of objective evaluation of textile surfaces for quality control in the textile industry. Details are given on the three major textile surface appearance attributes, namely, pilling, wrinkling and polar fleece fabric appearance. Modeling methods for different textile materials with different surface features are described. These include template matching for pilling modeling, morphological fractals for polar fleece fabrics and photometric stereo for 3D wrinkling.

In Chapter 16, Zhigeng Pan, Mingmin Zhang and Tian Chen compare several online 3D product presentation web sites and then focus on the current virtual shopping malls web sites which integrate Virtual Reality into E-commerce. The work is substantiated by the two applications developed by the authors, EasyMall, a virtual shopping mall, and EasyShow, a virtual presentation of textile products.

In Chapter 17, Xiaoyi Jiang and Hanspeter Bieri address the difficulties in the traditional synthetic modeling approaches and indicate new directions for modeling complex objects and environments through data acquisition from various active sensors, data fusion, and integration models.

In Chapter 18, Inas Khalifa, Medhat Moussa, and Mohamed Kamel discuss the accurate extraction of geometric and topological information from CAD drawings based on local approximation of scan lines. This particular application contains numerous basic fundamental problems that reflect the convergence of image and graphics techniques for the purpose of reverse-engineering architectural and engineering drawings.

In Chapter 19, Ossama El Badawy and Mohamed Kamel extend the 3D shape understanding problem by a new query-by-example retrieval method that is able to match a query image or graphical sketch to one of the images in the database, based on a whole or partial match of a given shape. The method has two key components: the architecture of the retrieval and the features that are invariant with respect to basic transformations.

In Chapter 20, Haikel Salem Alhichri and Mohamed Kamel propose a new image registration method, based on the Hausdorff fraction similarity measure and a multi-resolution search of the transformation space. This method is then applied to problems involving translations, scaling, and affine transformations in both image and graphic pictures.

References

- [1] K. Akeley, D. Kirk, L. Seiler, and P. Slusallek. "When will ray-tracing replace rasterization?" In *SIGGRAPH02, Online Panel*, http://online.cs.nps.navy.mil/DistanceEducation/online.siggraph.org/2002/Panels/01_WhenWillRayTracingReplaceRasterization/session.html, July, 2002.
- [2] G. Baciú and S. K. Wong. "Image-based techniques in a hybrid collision detector". *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 254-271, 2003.
- [3] B. G. Blundell, A. J. Schwarz. "The classification of volumetric display systems: characteristics and predictability of the image space". *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 66-75, January-March, 2002.

- [4] P. Debevec. "Rendering synthetic objects into real scenes: bridging traditional and image based graphics with global illumination and high dynamic range photography". In *SIGGRAPH 98*, July, 1998
- [5] P. E. Debevec and J. Malik. "Recovering high dynamic range radiance maps from photographs". In *SIGGRAPH 97*, pp. 369-378, August, 1997.
- [6] M. Deering and D. Naegle. "The SAGE graphics architecture". In *Proceedings of SIGGRAPH 2002*, ACM Press/ACM SIGGRAPH, pp. 683-692.
- [7] O. El Badawy and M. Kamel. "Shape-based image retrieval applied to trademark images". *International Journal of Image and Graphics*, vol. 2, no. 3, pp. 375-393, 2002.
- [8] O. Faugeras and Q. T. Luong. "The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications". *MIT Press*, 2001.
- [9] M. Haubner, C. Krapichler, A. Lösch, K.-H. Englmeier, and W. van Eimeren. "Virtual reality in medicine-computer graphics and interaction techniques". *IEEE Transactions on Information Technology in Biomedicine*, vol. 1, no. 1, pp. 61-72, March 1997.
- [10] Z. C. Li. "Discrete techniques for 3-D digital images and patterns under transformation". *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 6, pp. 883-894, December, 1998.
- [11] S. Molnar, J. Eyles, and J. Poulton. "Pixelflow: high-speed rendering using image composition". *Computer Graphics, Proceedings of SIGGRAPH 92*, pp. 231-240, July, 1992.
- [12] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. "Image-based modeling and photo editing". *Proceedings of SIGGRAPH 2001*, pp. 433-442, August, 2001.
- [13] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt and P. Hanrahan. "Lightning-2: a high performance display subsystem for PC clusters". *Computer Graphics Proceedings, Annual Conference Series, ACM*, pp. 141-148, 2001.
- [14] J. Unger, A. Wenger, T. Hawkins, A. Gardner and P. Debevec. "Capturing and rendering with incident light fields". *Eurographics Symposium on Rendering 2003*, pp. 1-10, Per Christensen and Daniel Cohen-Or , Eds.
- [15] P. L. Williams, N. L. Max, C. M. Stein. "A high accuracy volume renderer for unstructured data". *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 37-54, January-March, 1998.
- [16] J.A. Paradiso. "Tracking contact and free gesture across large interactive surfaces". *Communications of the ACM*, vol. 46, no. 7, pp. 62-68, 2003.
- [17] M. Tsang, G. Fitzmaurice, G. Kurtenbach and A. Khan. "Game-like navigation and responsiveness in non-game applications". *Communications of the ACM*, vol. 46, no. 7, pp. 56-61, 2003.
- [18] C. Tomasi, A. Rafii and I. Torunoglu. "Full-size projection keyboard for handheld devices". *Communications of the ACM*, vol. 46, no. 7, pp. 70-75, 2003.

This page intentionally left blank

Chapter 2

TRANSFORMING 3D MESH SURFACES INTO IMAGES BY PARAMETERIZATION

Jingqi Yan ¹, Pengfei Shi ¹ and David Zhang ²

1. *Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University*

2. *Department of Computing, The Hong Kong Polytechnic University, Hong Kong*

{jqyan,pfshi}@sjtu.edu.cn, csdzhang@comp.polyu.edu.hk

Abstract This chapter addresses the problem of mesh parameterization for transforming 3D mesh surfaces into images. We begin with a survey of the algorithms for parameterizing a *chart*, a simply connected mesh isomorphic to a topological disk, into a 2D domain and follow this with a review of several classic and popular parameterization algorithms based on solving the linear or non-linear systems, for instance, barycentric mapping, conformal mapping, harmonic mapping, and geometrical stretch minimizing. We then introduce state-of-the-art methods for opening an arbitrary mesh into a topological disk by cutting along an appropriate set of edges or for partitioning it into several charts. The chapter ends with a discussion of some related applications, such as texture mapping, remeshing, and mesh compression.

Keywords: Mesh parameterization, barycentric mapping, conformal mapping, harmonic mapping, geometrical stretch, computational topology, polygonal schema, mesh simplification, mesh partitioning, texture mapping, remeshing, mesh compression

2.1. Introduction

Recent advances in 3D data acquisition, together with the software for data aligning and mesh generation, allow us to accurately digitize the geometrical shape and surface properties (such as color, texture, and normal) of many physical objects. A proliferation of meshes with increasingly complex structure and realistic details are readily available, coming from a variety of sources including 3D scanners, modeling software, and computer vision.

However, the common meshes usual with their (sometimes highly) irregular connectivity, random sampling, and huge size (including the excessive cost on the flat regions) are far from ideal for subsequent processes such as, Finite Element computations, real-time textured visualization, compression, transmission, and multiresolution analysis. Instead, meshes with (semi-) regular connectivity, nearly equilateral triangles, appropriate sampling density, or image-based texture representation are much preferable inputs to most existing geometry processing algorithms.

One of the most promising techniques for improving the efficiency and broadening the applications of these manipulations is to transform the meshes into images. Once a mesh is parameterized into a 2D rectangular domain, the geometrical information and all of its intrinsic properties can be alternatively represented in the form of images with the same parameterized coordinates. Then, surface properties can be conveniently stored, alternated, or rendered with mapping techniques, such as texture [57,3,39,8,24,47-50,34-36,19], bump [4,45], and normal mapping [15,8,19], which are increasingly supported by most graphics systems. In addition, flexible remeshing techniques [5-6,19,1], together with powerful image processing tools, allow us to convert an arbitrarily complex mesh into a new mesh with a variety of characteristics including uniformity, regularity, semi-regularity, curvature sensitive resampling, and feature preservation, all of which are important for subsequent applications. Furthermore, mesh compression can also benefit from sophisticated image-compression coders.

The remainder of this chapter is organized as follows. Section 2.2 surveys several typical chart parameterization algorithms. Section 2.3 introduces some methods for opening an arbitrary mesh into a topological disk by cutting along an appropriate set of edges and for partitioning it into several charts. Section 2.4 discusses some interesting applications related to the mesh-to-image transform. In Section 2.5 we offer some conclusions.

2.2. Chart Parameterization

In this chapter, we state the chart parameterization problem as follows: Given a chart M , i.e., a simply connected triangular mesh isomorphic to a topological disk, construct a mapping f between M and an isomorphic planar triangulation $P \subset \mathbb{R}^2$ that best preserves some user-specific characteristic of M . Throughout the chapter, we denote by $\mathbf{x}_i$ the 3D position of the i^{th} vertex in the original mesh M , and by $\mathbf{u}_i$ the 2D position (parameter value) of the corresponding vertex in the 2D mesh P . We also use the self-explanatory notation: $\mathbf{x}_i = (x_i, y_i, z_i)^T$, $\mathbf{u}_i = (u_i, v_i)^T$. Specially assume, by re-labelling vertices if necessary, that $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are the internal vertices and $\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \dots, \mathbf{x}_N$ are the boundary vertices in any anticlockwise sequence with respect to the boundary of M .

Due to its primary importance for the subsequent mesh manipulations, chart parameterization has for a number of years been the subject of research, and not only in Computer Graphics. These years have seen the publication of a large body of work on parameterization, with some earlier work referring to the problem as *surface flattening* [3,39,51,55]. Throughout the previous work, almost every parameterization technique sought, either implicitly or explicitly, to produce least-distorted parameterizations, and varied only in the distortions they addressed and the minimization processes that were used. In the following, we broadly classify these methods into two categories, linear and non-linear, according to the different minimization processes, and briefly review them. Because of the large amount of published work on chart parameterization, our review is necessarily incomplete.

2.2.1 Linear Methods

Linear methods are used to convert chart parameterization problems into computable sparse linear systems with sophisticated Conjugate Gradient methods [18,54]. We broadly divide these methods into three categories according to the distortions that they addressed: barycentric mapping, conformal mapping, and harmonic mapping. Such classifications are not strict however. Some methods, for instance, are both conformal and harmonic.

Barycentric Mapping. Tutte [53] introduced barycentric mapping as an early method for making a straight line drawing of a planar graph. Within this mapping, any internal vertex is a linear convex combination of its neighbors in the parameterized domain. The general procedure of barycentric mapping for chart parameterization can be described as follows [14]:

- 1) Choose $\mathbf{u}_{n+1}, \mathbf{u}_{n+2}, \dots, \mathbf{u}_N$ as the vertices of any K -sided convex polygon $P \subset \mathbb{R}^2$ in an anticlockwise sequence.
- 2) For each $i \in \{1, 2, \dots, n\}$, choose any set of real numbers $\lambda_{i,j}$ for $j = 1, 2, \dots, N$ such that

$$\lambda_{i,j} = 0, (i, j) \notin E, \lambda_{i,j} > 0, (i, j) \in E, \sum_{j=1}^N \lambda_{i,j} = 1. \quad (2.1)$$

And define $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$ as the solutions of the linear system of equations,

$$\mathbf{u}_i = \sum_{j=1}^N \lambda_{i,j} \mathbf{u}_j, \quad i = 1, 2, \dots, n. \quad (2.2)$$

Note that it can be rewritten in the form

$$\mathbf{u}_i - \sum_{j=1}^n \lambda_{i,j} \mathbf{u}_j = \sum_{j=n+1}^N \lambda_{i,j} \mathbf{u}_j, \quad i = 1, 2, \dots, n. \quad (2.3)$$

By considering the two components u_i and v_i of $\mathbf{u}_i$ separately this is equivalent to the two matrix equations

$$\mathbf{A}\mathbf{u} = \mathbf{b}_1, \quad \mathbf{A}\mathbf{v} = \mathbf{b}_2, \quad (2.4)$$

where $\mathbf{u}$ and $\mathbf{v}$ are the column vectors $(u_1, u_2, \dots, u_n)^T$ and $(v_1, v_2, \dots, v_n)^T$ respectively. The matrix $\mathbf{A}$ is $n \times n$ having elements

$$a_{i,i} = 1, \quad a_{i,j} = -\lambda_{i,j}, \quad j \neq i. \quad (2.5)$$

The existence and uniqueness of a solution to the equations in (2.2) is thus equivalent to be the non-singularity of the matrix $\mathbf{A}$.

Tuette [53] proposed equations (2.2) in the special case of drawing a planar graph with straight lines when $\lambda_{i,j} = 1/d_i$ for all $(i, j) \in E$, $i = 1, 2, \dots, n$ (i.e., $\mathbf{u}_i$ is the barycentre of its neighbors). He proved in this case that a unique solution was guaranteed. In his method, the weights were evenly assigned, which can be regarded as a generalization of uniform parameterization for point sequences.

Floater [14] generalized barycentric mapping by allowing every internal vertex to be any convex combination of its neighbors and provided an algorithm for choosing the convex combinations so that the local shape of the original mesh patch is preserved. The basic idea of his algorithm is shown in Figure 2.1 and can be described as follows:

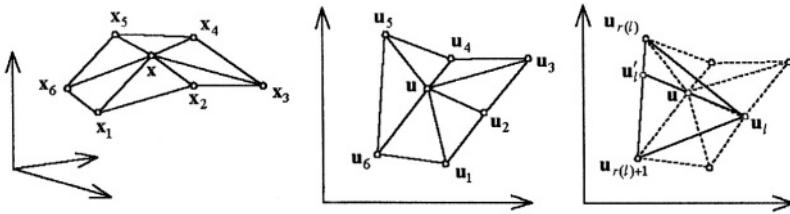


Figure 2.1. A local illustration for the Floater's Parameterization.

Firstly, he adopted the method proposed by Welch and Witkin [56] for making local parameterizations to emulate the so-called geodesic polar map, a local mapping known in differential geometry that preserves arc length in each radial

direction. Suppose $\mathbf{x}$ is any internal vertex of the input 3D mesh and $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d$ is its neighbors in an anticlockwise sequence. Let $\text{ang}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ denote the angle between the vectors $\mathbf{a}-\mathbf{b}$ and $\mathbf{c}-\mathbf{b}$. Then, choose any $\mathbf{u} \in \mathbb{R}^2$ and $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d \in \mathbb{R}^2$ satisfying, for $k = 1, 2, \dots, d$,

$$\|\mathbf{u}_k - \mathbf{u}\| = \|\mathbf{x}_k - \mathbf{x}\|, \text{ang}(\mathbf{u}_k, \mathbf{u}, \mathbf{u}_{k-1}) = 2\pi \text{ang}(\mathbf{x}_k, \mathbf{x}, \mathbf{x}_{k+1}) / \theta \quad (2.6)$$

where $\theta = \sum_{k=1}^d \text{ang}(\mathbf{x}_k, \mathbf{x}, \mathbf{x}_{k+1})$ and $\mathbf{x}_{d+1} = \mathbf{x}_1$, $\mathbf{u}_{d+1} = \mathbf{u}_1$. In the implementation, $\mathbf{u} = 0$ and $\mathbf{u}_1 = (\|\mathbf{x}_1 - \mathbf{x}\|, 0)$ are commonly used for configuration and then $\mathbf{u}_2, \dots, \mathbf{u}_d$ are computed in sequence.

Secondly, for each $l \in \{1, 2, \dots, d\}$, the straight line through $\mathbf{u}_l$ and $\mathbf{u}$ intersects the polygon at a unique second point $\mathbf{u}'_l$ which is either a vertex $\mathbf{u}_{r(l)}$ or locates on a line segment with endpoints $\mathbf{u}_{r(l)}$ and $\mathbf{u}_{r(l)+1}$. In either case, there is a unique $r(l) \in \{1, 2, \dots, d\}$, and unique $\delta_{l,k}$, $k = 1, 2, \dots, d$ given in equations (2.7) such that $\delta_{l,l} > 0$, $\delta_{l,r(l)} > 0$, $\delta_{l,k} \geq 0 (k \neq l, k \neq r(l))$, $\sum_{k=1}^d \delta_{l,k} = 1$, and

$$\mathbf{u} = \sum_{k=1}^d \delta_{l,k} \mathbf{u}_k.$$

$$\delta_{l,k} = \begin{cases} \text{area}(\mathbf{u}, \mathbf{u}_{r(l)}, \mathbf{u}_{r(l)+1}) / \text{area}(\mathbf{u}_l, \mathbf{u}_{r(l)}, \mathbf{u}_{r(l)+1}), & k = l \\ \text{area}(\mathbf{u}_l, \mathbf{u}, \mathbf{u}_{r(l)+1}) / \text{area}(\mathbf{u}_l, \mathbf{u}_{r(l)}, \mathbf{u}_{r(l)+1}), & k = r(l) \\ \text{area}(\mathbf{u}_l, \mathbf{u}_{r(l)}, \mathbf{u}) / \text{area}(\mathbf{u}_l, \mathbf{u}_{r(l)}, \mathbf{u}_{r(l)+1}), & k = r(l) + 1 \\ 0, & \text{others} \end{cases} \quad (2.7)$$

Select l throughout $\{1, 2, \dots, d\}$, then

$$\mathbf{u} = \frac{1}{d} \sum_{l=1}^d \sum_{k=1}^d \delta_{l,k} \mathbf{u}_k = \sum_{k=1}^d \left(\frac{1}{d} \sum_{l=1}^d \delta_{l,k} \right) \mathbf{u}_k. \quad (2.8)$$

Finally, make $\lambda_k = \frac{1}{d} \sum_{l=1}^d \delta_{l,k}$, $k = 1, 2, \dots, d$ such that $\lambda_k > 0$, $\sum_{k=1}^d \lambda_k = 1$ and

thus $\mathbf{u} = \sum_{k=1}^d \lambda_k \mathbf{u}_k$, a linear convex combination of its neighbors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$.

From the above definitions, Floater's method considered the weights for convex combination in terms of the local edge-length distortion and conformality, which was shape preserving in some sense. Note that the weights in his method are local and static so that it is not completely effective in preserving the shape of the 3D mesh during the dynamic parameterization procedure.

Recently, Levy and Mallet [35] took additional linear constraints into account and used the modified barycentric mapping method for non-distorted texture mapping.

Conformal Mapping. There are many methods that seek to minimize the distortion of the corresponding angles between the 3D mesh and its 2D parameterization (a conformal map [24,51,36,20]). Here, because of its simplicity, we introduce the method presented by Pinkall and Polthier [46], which was recently developed for intrinsic parameterization [10] combining authalic mapping.

Pinkall and Polthier first defined the Dirichlet energy of a linear mapping f between two triangles for chart parameterizations by:

$$E_D(f) = \frac{1}{4} (\cot(\text{ang}(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_1)) \|\mathbf{u}_2 - \mathbf{u}_1\|^2 + \cot(\text{ang}(\mathbf{x}_3, \mathbf{x}_1, \mathbf{x}_2)) \|\mathbf{u}_3 - \mathbf{u}_2\|^2 + \cot(\text{ang}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)) \|\mathbf{u}_1 - \mathbf{u}_3\|^2) \quad (2.9)$$

As an immediate consequence they defined the Dirichlet energy of a mapping f between two triangulated mesh surfaces as the sum of all energies on triangles:

$$E_D(f) = \sum_{i=1}^{\# \text{triangles}} E_D(f_i) \\ = \frac{1}{4} \sum_{(i,j) \in E} (\cot \alpha_i + \cot \beta_i) \|\mathbf{u}_i - \mathbf{u}_j\|^2 \quad (2.10)$$

where (i, j) is one edge of the input 3D mesh patch, and α_i and β_i are the angles opposite the edge (i, j) in the two adjacent triangles. For boundary edges there is only one triangle incident to this edge so that one term is assumed to be zero.

Mimicking the differential case, Pinkall and Polthier proposed to define the *discrete conformal map* as the critical point (also known as the minimum) of the Dirichlet energy. Since this energy is quadratic, the derivation results in a simple linear system,

$$\frac{\partial}{\partial \mathbf{u}_i} E_D(f) = \sum_{j \in \text{Nhr}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (\mathbf{u}_i - \mathbf{u}_{ij}) = 0, \text{ for } i = 1, 2, \dots, n \quad (2.11)$$

where $\mathbf{u}_i$ is any internal vertex, $\mathbf{u}_{ij}$ are its neighbors, and α_{ij} and β_{ij} are the angles opposite the edge (i, ij) in the two adjacent triangles, as shown in Figure 2.2. It has a provably unique solution that is easy to compute once we fix the boundaries, $\mathbf{u}_{n+1}, \mathbf{u}_{n+2}, \dots, \mathbf{u}_N$, in the parameter domain.

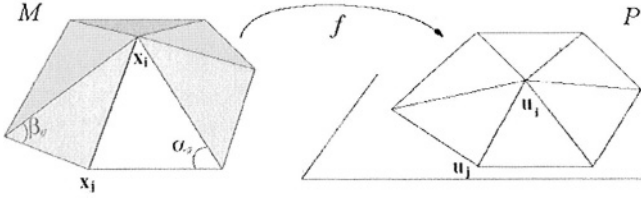


Figure 2.2. A 3D 1-ring and its associated parameterized domain.

Although there is generally no way to flatten a curved, discrete surface without the distortion of angles, the Dirichlet energy depends only on the 3D angles, and that in the differential case, the minimum of the Dirichlet energy is indeed conformal.

Harmonic Mapping. Harmonic mapping can be visualized as follows. Imagine M to be composed of elastic, triangular rubber sheets sewn together along their edges. Stretch the boundary of M over the boundary of the polygon P according to the mapping g . Using the elasticity theory, Eck et al. [12] reinterpreted the harmonic mapping by minimizing the total spring-like energy $E_h(f)$ over this configuration of rubber sheets:

$$E_h(f) = \frac{1}{2} \sum_{(i,j) \in E} K_{i,j} \|\mathbf{u}_i - \mathbf{u}_j\|^2, \quad (2.12)$$

where $K_{i,j}$ are the spring constants.

There are several methods for choosing the spring constants to approximate the harmonic mapping in a piecewise linear way. One such method, introduced by Kent et al. [29], chose the spring constants as either all equal (i.e., the uniform spring constants) or inversely proportional to edge lengths as measured in the original mesh. Applying the method of Kent et al. to complex meshes, however, neither choice consistently produces satisfactory results. Eck et al. [12] presented the following more competitive method for choosing the spring constants: Let $L(i, j)$ and $Area(i, j, k)$ denote the length of one edge (i, j) and the area of one triangle (i, j, k) , respectively, both as measured in M . Each interior edge (i, j) is incident to two triangles, say (i, j, k_1) and (i, j, k_2) . Then

$$\begin{aligned} K_{i,j} = & (L^2(i, k_1) + L^2(j, k_1) - L^2(i, j)) / Area(i, j, k_1) \\ & + (L^2(i, k_2) + L^2(j, k_2) - L^2(i, j)) / Area(i, j, k_2) \end{aligned} \quad (2.13)$$

For the boundary edges, the formula for computing the spring constants has only one term.

The system (2.12) is positive definite although the spring constants computed by the equations in (2.13) can assume negative values and its unique minimum can be found by solving a sparse linear least-squares problem. In contrast with harmonic mapping itself, the piecewise linear approximation of Eck et al. is not always an embedding. In these cases the uniform spring constants are used instead. Other than the simplicity of computation, the total performance of the method by Eck et al. is comparable to that of the nonlinear method, also based on elasticity theory, by Maillot et al. [39].

2.2.2 Non-Linear Methods

To find a good parameterization from the mesh M to the 2D domain P , many researchers took into consideration the combination of the distortions on the corresponding edge-length and triangle-area [39], the Dirichlet Energy per parameter area [26], or the average and maximum geometric stretch [49,50], which resulted in the non-linear metrics as well as the non-linear minimization process. Since the geometric-stretch metric differs greatly from the metrics mentioned in the previous sections, we introduce it in detail as follows.

Given a triangle T with $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$, $\mathbf{u}_i = (u_i, v_i)$ as the coordinates of its three vertices in the 2D domain, and the corresponding 3D coordinates $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, the unique affine mapping f is

$$\begin{aligned} f(\mathbf{u}) = & (Area(\mathbf{u}, \mathbf{u}_2, \mathbf{u}_3)\mathbf{x}_1 + Area(\mathbf{u}_1, \mathbf{u}, \mathbf{u}_3)\mathbf{x}_2 \\ & + Area(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u})\mathbf{x}_3) / Area(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \end{aligned} \quad (2.14)$$

Since the mapping is affine, its partial derivatives are constant over (u, v) and given by

$$\begin{cases} f_u = (\mathbf{x}_1(v_2 - v_3) + \mathbf{x}_2(v_3 - v_1) + \mathbf{x}_3(v_1 - v_2)) / (2Area(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)) \\ f_v = (\mathbf{x}_1(u_3 - u_2) + \mathbf{x}_2(u_1 - u_3) + \mathbf{x}_3(u_2 - u_1)) / (2Area(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)) \end{cases} \quad (2.15)$$

The Jacobian matrix of the mapping f is $J_f(u, v) = \begin{bmatrix} f_u & f_v \end{bmatrix}$. Its singular values, Γ and γ , represent the largest and smallest lengths obtained when mapping unit-length vectors from the 2D domain to the mesh, i.e., the largest and smallest geometric stretch. They are given by

$$\begin{cases} \Gamma = \sqrt{\frac{1}{2} \left((a_f + c_f) + \sqrt{(a_f - c_f)^2 + 4b_f^2} \right)} \\ \gamma = \sqrt{\frac{1}{2} \left((a_f + c_f) - \sqrt{(a_f - c_f)^2 + 4b_f^2} \right)} \end{cases} \quad (2.16)$$

where $a_f = f_u \cdot f_u$, $b_f = f_u \cdot f_v$, $c_f = f_v \cdot f_v$. Then, two stretch norms over triangle T are defined by:

$$L_2(T) = \sqrt{(\Gamma^2 + \gamma^2)/2} = \sqrt{(a_f + c_f)/2}, \quad L_\infty(T) = \Gamma \quad (2.17)$$

The norm $L_2(T)$ corresponds to the root-mean-square stretch, and the worst-case norm $L_\infty(T)$ is the greatest stretch over all directions in the domain. Once the area of the parameterization of T becomes zero (i.e., degenerate) or negative (i.e., flipped), both norms are defined to infinity.

Two analogous norms over the entire mesh $M = \{T_i\}$ are given by:

$$L_2(M) = \sqrt{\frac{\sum_{T_i \in M} (L_2(T_i))^2 A'(T_i)}{\sum_{T_i \in M} A'(T_i)}}, \quad L_\infty(M) = \max_{T_i \in M} L_\infty(T_i). \quad (2.18)$$

Such stretch norms can be normalized by multiplying with a factor, $\sqrt{\sum_{T_i \in M} A(T_i) / \sum_{T_i \in M} A'(T_i)}$, where $A(T_i)$ and $A'(T_i)$ are the area of triangle T in the 2D domain and in the mesh, respectively.

To minimize the non-linear metrics $L_2(M)$ and $L_\infty(M)$, it starts from a solution produced by one of the linear approaches (such as the uniform spring-like energy minimization), and then performs several optimization iterations. Within each iteration the neighborhood stretches of vertices are considered and listed in decreasing order, then, for each vertex, a line search along a randomly chosen search direction in the 2D kernel of the polygons enclosed by its neighbors is performed to minimize its neighborhood stretch as well as the total stretch.

2.2.3 Examples

Figure 2.3 shows chart parameterizations using several methods. The simple mesh used here is the head of the Cat model present in the work of Hoppe et al. [25]. The boundary of this mesh is mapped into a square using the chord-length parameterization [14]. The corner vertices are marked as the red dots in Figure 2.3 (a). According to [50], the geometric-stretch minimizing method is often better at capturing high-frequency detail over the entire surfaces.

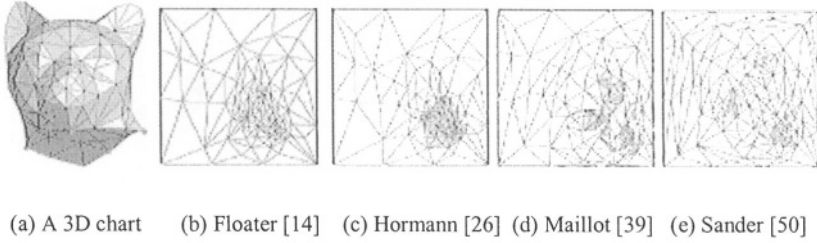


Figure 2.3. Chart parameterization by several methods.

2.3. Transforming Meshes into Images

In this section, we introduce two schemes for transforming an arbitrary mesh into an image using parameterization. One is the single-chart transform scheme: open the mesh into a single chart by cutting along an appropriate set of edges and then parameterize it into a 2D rectangular domain. The other is the multi-chart transform scheme: to avoid the excessive distortion from the single-chart transform, the mesh is firstly partitioned into several charts by some criteria, then each chart is parameterized into a 2D domain, and finally the parameterized charts are packed into an image.

2.3.1 Single-Chart Transform

To transform a manifold mesh of arbitrary genus to the representation of a single image by parameterization, the first task is to find an initial cut that opens the mesh topologically isomorphic to a disk. The cut mesh can then be regarded as a single patch and transformed into a 2D rectangular domain as an image by patch parameterization.

It is well known that any manifold polygonal mesh can be opened into a topological disk by cutting along an appropriate set of edges, called a polygonal schema [41,31,13]. Such an algorithm presented by Dey et al. [11] and developed by Gu et al. [19] works as follows.

First, let B be the set of boundary edges in the original mesh. This set remains frozen throughout the algorithm, and is always a subset of the final cut. Second, arbitrarily select a triangle as the seed to remove from the mesh, and then perform two removing subprocedures: edge-triangle removals and vertex-edge removals. In the edge-triangle removing subprocedure, repeatedly identify an edge $e \notin B$ adjacent to exactly one triangle, and remove both the edge and the triangle. Note that the two remaining edges of the triangle are left in the simplicial complex, even if they are dangling. In order to obtain a result of "minimal radius", the candidate edge-triangle removals are ordered according to their geodesic distance from the

seed triangle. When this subprocedure terminates, a topological disk that includes all of the faces of the mesh has been removed. Thus, the remaining edges must form a topological cut of the mesh but also include many unnecessary edges. In the vertex-edge removing subprocedure, repeatedly identify a vertex adjacent to exactly one edge (i.e. a dangling edge), and remove both the vertex and the edge. This second subprocedure for the vertex-edge removals terminates when all the dangling edges have been trimmed away, leaving just the connected loops. At this moment, the resulting cut may be serrated, i.e., it is not made up of the shortest paths. Finally, each cut-path in the cut is straightened by computing a constrained shortest path that connects its two adjacent cut-nodes and stays within a neighborhood of the original cut-path. Specially, for a closed mesh of genus zero, the resulting cut consists of a single vertex so that two edges incident to the remaining vertex are added back to the cut.

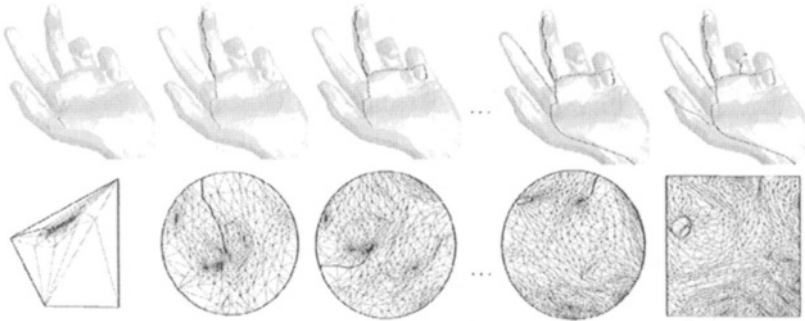


Figure 2.4. Transforming a mesh to a single-chart image by Gu et al. [19].

Such an initial cut only guarantees the mesh can be opened into a topological disk. However, the transform between the mesh and its 2D version agreeing with mapping the initial cut to a rectangle in the 2D domain might produce a large distortion. Thus, the second work of transforming a mesh into a single image is to optimize the cut as well to improve the quality of the transform. Gu et al. [19] proposed an iterative algorithm with the cut optimization and the patch parameterization by turns. A result of their algorithm is shown in Figure 2.4. The basic strategy is as follows: Repeatedly, parameterize the mesh into a circular domain under the current cut using a shape-preserving parameterization of Floater [14]; find an "extremal" vertex, which is one vertex of the triangle with the maximum geometric stretch [50]; find the shortest path (as measured in the mesh M) from the extremal vertex to the current cut; and finally update the cut by adding this shortest path. This procedure terminates when the geometric stretch of the entire mesh using the geometric-stretch parameterization [50] (at this time, the

mesh is parameterized into a rectangular domain instead of a circular domain) cannot be reduced.

2.3.2 Multi-Chart Transform

Although single-chart transforms can create completely regular meshes by remeshing techniques and avoid the discontinuity across multiple charts, they may introduce significant geometric stretch and noticeable artefacts when the models are high genus or have very sharp edges.

The more common ways to avoid excessive distortion involve partitioning an arbitrary mesh into several charts and then parameterizing each chart into a 2D domain, referred to as a multi-chart parameterization [50,36]. However, partitioning the surface into many charts has drawbacks: multi-chart transforms can only create the semi-regular meshes by using current remeshing techniques and often produce serious discontinuities across the charts. Thus, a balance must be considered between the distortion of parameterization and the drawbacks of multiple charts.

Various mesh-partitioning algorithms have been developed in the past ten years. In the work of Pederson [44] and Krishnamurthy and Levoy [30], the user had to partition the model in an interactive way. To perform automatic segmentation, Maillot et al. [39] partitioned the mesh by a bucketing of face normals. Eck et al. [12] developed a Voronoi-based partition. Kalvin and Taylor [28] partitioned the surface into a set of "superfaces". Their algorithm merged two adjacent faces under a planarity threshold. Mangan and Whitaker [40] presented the curvature-based partitioning algorithm by generalizing morphological watersheds to 3D surfaces. Several multiresolution methods [33,23] decomposed the model into several charts corresponding to the simplices of the base complex. Garland et al. [17] described a process of hierarchical face clustering with quadric error metric. Following them, Sander et al. [50] improved the partitions with boundary straightening. More recently, Levy et al. [36] partitioned the mesh into charts with feature extraction and region growing.

Once the mesh is partitioned into several charts, a chart-merging operation is performed if it results in any chart with fewer than three corners or if the boundary between one chart and any adjacent chart consists of more than one connected component. This merging operation guarantees that each chart remains isomorphic to a topological disk. Furthermore, the boundaries of the charts can be optimized with the shortest paths in the mesh constrained not to intersect each other [50].

After the charts of an arbitrary mesh are created, patch parameterization can be used to individually transform each of them into a 2D polygonal domain. Then, each parameterized chart polygon is uniformly resized to coincide with the area as measured in the 3D mesh, or with the average geometric stretch between the 2D domain and the corresponding 3D chart. Finally, the parameterized charts are

packed into a rectangular image [42,2,50,36]. An example for the multi-chart transform is shown in Figure 2.5.

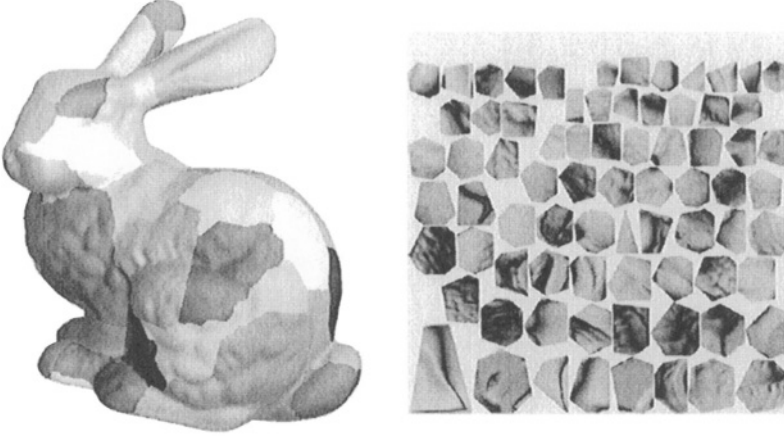


Figure 2.5. Transforming a mesh into a multi-chart image by Sander et al. [50].

2.4. Applications

This section introduces several significant applications related to the mesh-to-image transform. At present, these applications and their implied techniques are attracting an increasing research attention, and not only in the computer graphics community.

Texture Mapping. Texture mapping is a common technique for dealing with color on a surface. Once a mesh is parameterized into a 2D rectangular domain (i.e., image), the one-to-one correspondence between the vertices $\mathbf{x}_i$ in the mesh and the vertices $\mathbf{u}_i$ in the image is built: $f(\mathbf{x}_i) = \mathbf{u}_i$; and then any point $\mathbf{p}_i$ on the surface tiled with the mesh can be mapped to one pixel $\mathbf{I}_i$ of the image:

Suppose $\mathbf{p}_i$ is in the interior (or on the boundary) of the triangle $\{\mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l\}$ in the mesh, represented by $\mathbf{p}_i = \alpha_i \mathbf{x}_j + \beta_i \mathbf{x}_k + \gamma_i \mathbf{x}_l$, $\alpha_i, \beta_i, \gamma_i \geq 0$, $\alpha_i + \beta_i + \gamma_i = 1$ using the barycentric mapping, then build the mapping

$$f(\mathbf{p}_i) = \alpha_i f(\mathbf{x}_j) + \beta_i f(\mathbf{x}_k) + \gamma_i f(\mathbf{x}_l) = \alpha_i \mathbf{u}_j + \beta_i \mathbf{u}_k + \gamma_i \mathbf{u}_l = \mathbf{I}_i. \quad (2.19)$$

Thus, the color on a surface can be conveniently stored into an image [8,50,19]; on the contrary, texturing the mesh M is then as simple as pasting a picture onto the parameter domain, and mapping each triangle of the original mesh M with the part of the picture present within the associated triangle in the parameter plane, also referred to as 3D painting [36]. See Figure 2.6 as an example.

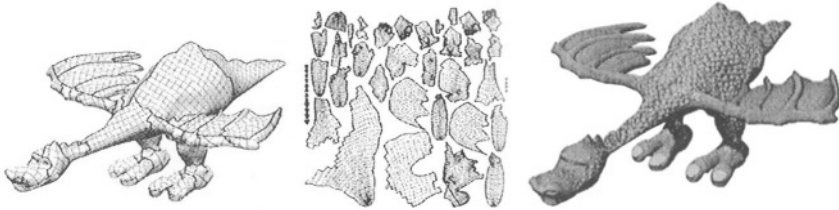


Figure 2.6. The mesh-to-image transform used for 3D painting by Levy et al. [36].

Texture mapping is just one instance of mapping. In fact, almost all of the intrinsic properties (displacement [7,32], normal [15,8,19], curvature [1], geometry [19], etc) on a surface can be alternatively represented in the form of 2D images as surface color by mapping. Other forms of mapping can use the same texture coordinate parameterization, but contain something other than surface color. For instance, displacement mapping [7] contains perturbations of the surface position, typically used to add surface detail to a simple model; bump mapping [4] is similar, but instead gives perturbations of the surface normal, used to make a smooth surface appear bumpy but not change the surface's silhouette; and normal mapping [15] can also make a smooth surface appear bumpy, but contain the actual normal instead of just a perturbation of the normal.

Texture mapping is available in most current graphics systems, including workstations and PCs. Other mapping techniques also become effective using the hardware accelerated OpenGL functions [45,43,6].

Remeshing. With advances in the hardware for 3D data acquisition and the software for data alignment and mesh generation, a mass of meshes with complex structures and vivid details are readily available, coming from a variety of sources including 3D scanners, modeling software, and output from computer vision algorithms. However, they often suffer from irregular connectivity, random sampling, and huge size, which are far from ideal for subsequent applications: for instance, Finite Element computations, detailed but real-time rendering, mesh compression and transmission, and multiresolution analysis. Instead, meshes with (semi-) regular connectivity, nearly equilateral triangles, or appropriate sampling density are preferable inputs to most existing geometry processing algorithms.

Remeshing, i.e., modifying the sampling and connectivity of a geometry to generate a new mesh with reasonable faithfulness to the origin, is therefore a fundamental technique for efficient mesh processing.

A great deal of work has already been done on remeshing [38,12,33,22-23,5-6,27]. The majority of these techniques are semi-regular remeshing techniques based on a base 3D mesh (also referred to as control mesh) constructed by mesh simplification [25,16,37]. Once an arbitrary mesh is transformed into an image by parameterizations, it becomes much more flexible in terms of the quality of the remeshing, including its uniformity, regularity, semi-regularity, curvature sensitive resampling, and feature preservation, with the signal processing and halftoning tools for images [5-6,19,1], as shown in Figure 2.7.

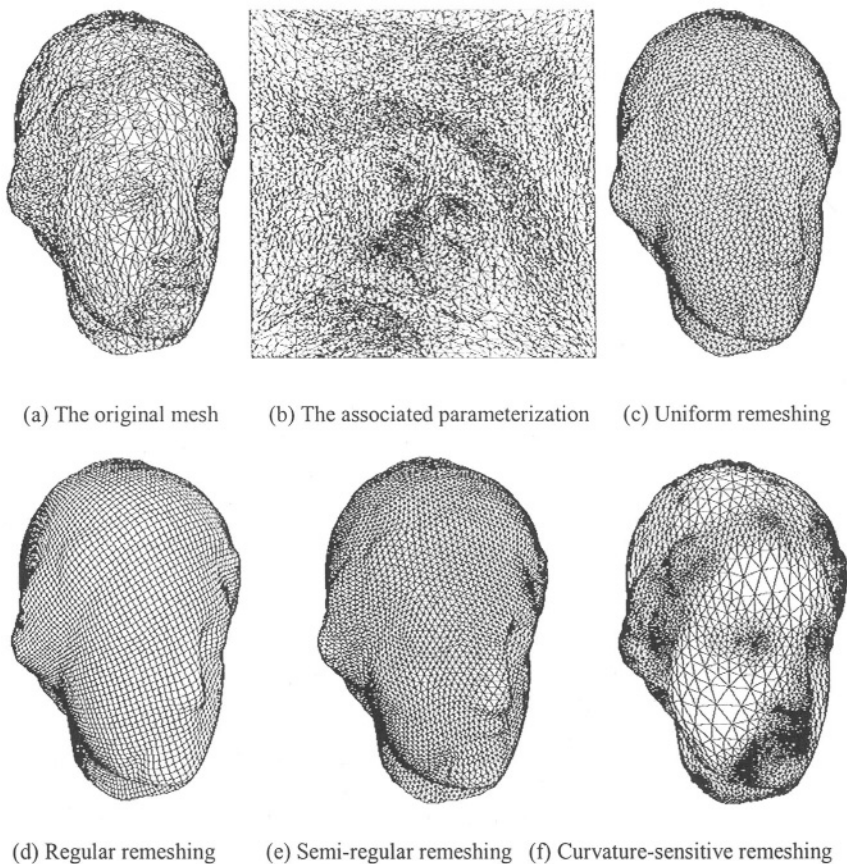


Figure 2.7. The mesh-to-image transform used for remeshing by Alliez et al. [1].

Mesh Compression. Mesh compression has recently been an active area of research. Several skilful schemes have been proposed for almost lossless compression [52,21]. The mesh connectivity is encoded in an average of less than two bits per triangle with an appropriate triangle-traverse order, and the mesh geometry (i.e., vertex positions) is compressed within the desired accuracy by quantization, local prediction, and entropy encoding with an appropriate vertex-traverse order. In addition, a multiresolution mesh compression approach has also been developed based on subdivision schemes. It works as follows: the mesh is firstly simplified into a control mesh, a coarse one with fewer triangles. The control mesh is then refined in several successive subdivisions in which the finer geometric displacements are represented by a set of vector-valued or even scale-valued wavelet coefficients [38, 12,32,23].

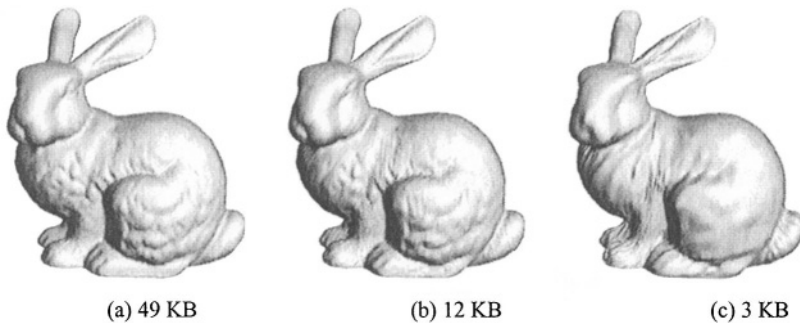


Figure 2.8. The mesh-to-image transform used for compression by Gu et al. [19].

More recently, Gu et al. [19] presented an image-based mesh compression scheme. One of their results is shown in Figure 2.8. As introduced in the previous sections of this chapter, the mesh is firstly transformed into an image by parameterizations; and then it can be encoded as an image. In our opinion, this is a promising avenue for mesh compression. Many effective image-compression coders, such as the wavelet coder [9], can be exploited to compress the meshes. Besides the mesh geometry, all of the intrinsic properties (such as color, normal, and curvature) of the mesh can be alternatively represented in the form of convenient images, sharing the same parameterization, so it is easy to construct a unified compression. Furthermore, using the powerful processing tools for images, it offers more flexible control of the quality of the mesh compression, decompression and reconstruction.

2.5. Conclusion

In this chapter, we surveyed current mesh-to-image transform technologies. We first focused on the methods for parameterizing a chart, i.e. a simple connected mesh isomorphic to a topological disk, into a 2D domain, and also introduced several distortions such as shape-preserving, angle-preserving, edge-length-preserving, and geometric stretch between the 3D mesh and its associated parameterization. Subsequently, the single-chart and multi-chart transform schemes for the arbitrary meshes were discussed, together with several interesting applications.

Acknowledgment

This work is supported by National Science Foundation of China (No. 69931010, No. 60273200).

References

- [1] P. Alliez, M. Meyer, and M. Desbrun. "Interactive geometry remeshing". In *Proceedings of ACM SIGGRAPH 2002*, Addison Wesley, pages 347–354, 2002.
- [2] Y. Azar and L. Epstein. "On 2d packing". *Journal of Algorithms*, vol. 25, pp. 290–310, 1997.
- [3] C. Bennis, J.-M. Vézien, G. Iglésias, and A. Gagalowicz. "Piecewise surface flattening for non-distorted texture mapping". *Computer Graphics (ACM SIGGRAPH 91)*, vol.25, no. 4, pp. 237-246, July, 1991.
- [4] J. Blinn. "Simulation of wrinkled surfaces". *Computer Graphics (ACM SIGGRAPH 78)*, vol. 12, pp. 286-292, 1978.
- [5] M. Botsch and L. Kobbelt. "Resampling feature and blend regions in polygonal meshes for surface anti-aliasing". In *Proceedings of Eurographics 2001*, pp. 402–410, September, 2001.
- [6] M. Botsch, C. Rossli, and L. Kobbelt. "Feature sensitive sampling for interactive remeshing". In *Proceedings of Vision, Modeling and Visualization 2000*, pp. 129-136, 2000.
- [7] R.L. Cook. "Shade trees". *Computer Graphics (ACM SIGGRAPH 84)*, vol. 18, pp. 223-231, July, 1984.
- [8] J. Cohen, M. Olano, and D. Manocha. "Appearance-preserving simplification". In *Proceedings of ACM SIGGRAPH 1998*, pp. 115-122, 1998.
- [9] G. Davis. "Wavelet image compression construction kit". <http://www.geoffdavis.net/dartmouth/wavelet/wavelet.html>.
- [10] M. Desbrun, M. Meyer, and P. Alliez. "Intrinsic parametrizations of surface meshes". In *Proceedings of Eurographics 2002*.

- [11] T.K. Dey and H. Schipper. "A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection". *Discrete and Computational Geometry* 14, pp. 93–110, 1995.
- [12] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. "Multiresolution analysis of arbitrary meshes". In *Proceedings of ACM SIGGRAPH 95*, Addison Wesley, August, 1995.
- [13] J. Erickson and S. Har-Peled. "Cutting a surface into a disk". In *Proceedings of ACM SoCG 2002*.
- [14] M.S. Floater. "Parametrization and smooth approximation of surface triangulations". *Computer Aided Geometric Design* 14, pp. 231–250, 1997.
- [15] A. Fournier. "Normal distribution functions and multiple surfaces". *Graphics Interface '92 Workshop on Local Illumination*, pp. 45–52, 1992.
- [16] M. Garland and P. Heckbert. "Simplifying surfaces with color and texture using quadric error metrics". In *Proceedings of IEEE Visualization 98*, pp. 263–269, 1998.
- [17] M. Garland, A. Willmott, and P. Heckbert. "Hierarchical face clustering on polygonal surfaces". In *Proceedings of ACM Symposium on Interactive 3D Graphics 2001*.
- [18] Gene Golub and Charles Van Loan. *Matrix Computations*. John Hopkins University Press, 2nd edition, 1989.
- [19] X. Gu, S. Gortler, and H. Hoppe. "Geometry images". In *Proceedings of ACM SIGGRAPH 2002*.
- [20] X. Gu and S.T. Yau. "Computing conformal structure of surfaces". In *Communication of Information and Systems*, IP Press, 2002.
- [21] S. Gumhold and W. Straßer. "Real time compression of triangle mesh connectivity". In *Proceedings of ACM SIGGRAPH 98*, pp. 133–140, 1998.
- [22] I. Guskov, W. Sweldens, and P. Schroder. "Multiresolution signal processing for meshes". In *Proceedings of ACM SIGGRAPH 99*, pp. 325–334, 1999.
- [23] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder. "Normal meshes". In *Proceedings of ACM SIGGRAPH 2000*, pp. 95–102, 2000.
- [24] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. "Conformal surface parameterization for texture mapping". *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 181–189, 2000.
- [25] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Mesh optimization". *Computer Graphics*, vol. 27 (*ACM SIGGRAPH 93*), pp. 19–26, 1993.
- [26] K. Hormann and G. Greiner. MIPS: "An efficient global parametrization method". In *Curve and Surface Design: Saint-Malo 1999* (2000), P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, Eds., Vanderbilt University Press, pp. 153–162.
- [27] K. Hormann, U. Labsik, and G. Greiner. "Remeshing triangulated surfaces with optimal parameterizations". *Computer-Aided Design*, vol. 33, pp. 779–788, 2001.
- [28] A. Kalvin and R. Taylor. "Superfaces: polygonal mesh simplification with bounded error". *IEEE Computer Graphics and Application*, vol. 16, pp. 64–77, 1996.
- [29] J.R. Kent, W.E. Carlson, and R.E. Parent. "Shape transformation for polyhedral objects". In *Proceedings of ACM SIGGRAPH 92*, pp. 47–54, July 1992.

- [30] V. Krishnamurthy and M. Levoy. "Fitting smooth surfaces to dense polygon meshes". In *Proceedings of ACM SIGGRAPH 96*, pp. 313-324, ACM Press, 1996.
- [31] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. "Computing a canonical polygonal schema of an orientable triangulated surface". In *Proceedings of ACM SCG 2001*, pp. 80-89, 2001.
- [32] A. Lee, H. Moreton, and H. Hoppe. "Displaced subdivision surfaces". In *Proceedings of SIGGRAPH 2000*.
- [33] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. "MAPS: multiresolution adaptive parameterization of surfaces". In *Proceedings of ACM SIGGRAPH 98*, pp. 95-104, Addison Wesley, 1998.
- [34] B. Levy. "Constrained texture mapping for polygonal meshes". In *Proceedings of ACM SIGGRAPH 2001*, pp. 417-424, August 2001.
- [35] B. Levy and J.L. Mallet. "Non-distorted texture mapping for sheared triangulated meshes". In *Proceedings of ACM SIGGRAPH 98*, Addison Wesley, 1998.
- [36] B. Levy, S. Petitjean, N. Ray, and J.L. Maillot. "Least squares conformal maps for automatic texture atlas generation". In *Proceedings of ACM SIGGRAPH 2002*, Addison Wesley, 2002.
- [37] P. Lindstrom and G. Turk. "Fast and memory efficient polygonal simplification". In *Proceedings of IEEE Visualization 98*, pp. 279-286, 1998.
- [38] M. Lounsbery, T. DeRose, and J. Warren. "Multiresolution analysis for surfaces of arbitrary topological type". *ACM Transactions on Graphics*, vol. 16, no. 1, pp. 34-73, January 1997.
- [39] J.L. Maillot, H. Yahia, and A. Verroust. "Interactive texture mapping". In *Proceedings of ACM SIGGRAPH 93*, pp. 27-34, 1993.
- [40] A.P. Mangan and R.T. Whitaker. "Partitioning 3D surface meshes using watershed segmentation". *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308-321, 1999.
- [41] J. Munkres. *Topology*. Prentice Hall, 2000.
- [42] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. "Rectangle-packing-based module placement". *IEEE ICCAD 1995*, pp. 472-479, 1995.
- [43] M. Olano and A. Lastra. "A shading language on graphics hardware: the pixelflow shading system". In *Proceedings of ACM SIGGRAPH 98*, 1998.
- [44] H. Pedersen. "Decorating implicit surfaces". In *Proceedings of ACM SIGGRAPH 95*, pp. 291-300, ACM Press, 1995.
- [45] M. Peercy, J. Airey, and B. Cabral. "Efficient bump mapping hardware". In *Proceedings of ACM SIGGRAPH 97*, pp. 303-306, 1997.
- [46] U. Pinkall and K. Polthier. "Computing discrete minimal surfaces". *Experimental Mathematics*, vol. 2, no.1, pp. 15-36, 1993.
- [47] D. Piponi and G. Borshukov. "Seamless texture mapping of subdivision surfaces by model pelting and texture blending". In *Proceedings of ACM SIGGRAPH 2000*, pp. 471-478. ACM Press, 2000.
- [48] E. Praun, A. Finkelstein, and H. Hoppe. "Lapped Textures". In *Proceedings of ACM SIGGRAPH 2000*, pp. 465-470, July 2000.

- [49] P. Sander, S. Gortler, J. Snyder, and H. Hoppe. "Signal specialized parametrization". In *Proceedings of Eurographics Workshop on Rendering 2002*, 2002.
- [50] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. "Texture mapping progressive meshes". In *Proceedings of SIGGRAPH 2001*, pp. 409–416, 2001.
- [51] A. Sheffer and E. Sturler. "Parameterization of faceted surfaces for meshing using angle-based flattening". *Engineering with Computers*, vol. 17, no.3, pp. 326–337, 2001.
- [52] G. Taubin and J. Rossignac. "Geometric compression through topological surgery". *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 84–115, 1998.
- [53] W.T. Tutte. "How to draw a graph". In *Proc. London Math Soc*, vol. 13, pp. 743–768, 1963.
- [54] Van der Vorst, H.A. "Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". *SIAMsci*, vol. 13, pp. 631–644, 1992.
- [55] C.L. Wang, S.F. Smith, and M.F. Yuen. "Surface flattening based on energy model". *Computer-Aided Design*, vol. 34, no. 11, pp. 823–833, 2002.
- [56] W. Welch and A. Witkin. "Free-form shape design using triangulated surfaces". In *Proceedings of ACM SIGGRAPH 94*, pp. 247–256, 1994.
- [57] L. Williams. "Pyramidal parametrics". In *Proceedings of ACM SIGGRAPH 83*, pp. 1–11, 1983.

Chapter 3

3D MODELING BASED ON ATTRIBUTED HYPERGRPHS

Li Rong¹ and Andrew K.C. Wong^{1, 2}

1. Pattern Discovery Software, Waterloo, Ontario, Canada

2. Systems Design Eng., University of Waterloo, Canada

{akcwong,lrong}@pdsoftware.com, akcwong@pami.uwaterloo.ca

Abstract This chapter introduces a mathematical framework for 3D visions and modeling based on attributed hypergraph representation (AHR). It presents the data structure and algorithms that build, manipulate and transform augmented AHRs. The AHR model is built upon irregular triangular meshes. A net-like data structure is designed to handle the dynamic changes in AHR to give flexibility to the graph structure. Our research and implementation demonstrate the integration of machine vision and computer graphics.

Keywords: 3D Vision, 3D modeling and reconstruction, representation, attributed graphs, attributed hypergraphs

3.1. Introduction

Computational modeling provides the theoretical fundamentals for a wide spectrum of research topics, such as machine vision and computer animation. With the emergence of virtual reality, intelligent modeling for a broader surrounding becomes increasingly important. The objective of 3D vision and modeling is to derive certain spatial and descriptive information from images and construct 3D models. The constructed 3D model can be represented in various internal structures, from triangular meshes to symbolic representations such as attributed hypergraphs (AHR). Imaging information extracted usually includes object surface colours and textures. Spatial information derived is related to the pose of object surface and edge features in 3D space. In an AHR, these features of the real world are virtually preserved, manipulated and transformed.

3.1.1 Object Modeling Applications

Model Based 3D Computer Vision is a process directly making use of object modeling. In model based 3D vision systems, 3D models can be matched against 2D views for object identification, pose calculation or motion analysis. *Computer Animation* has been making rapid progress largely because of its success in the entertainment industry. In a typical animation system, controlled by the animator, a multi-body is animated from its models according to kinematics and dynamic specifications. An intelligent modeling approach will minimize the operator interactions by allowing automatic derivation of local motion and deformation. *Virtual Reality (VR) and Augmented Virtual Reality (AR)* did not catch enough interest until the maturity of computer graphics and visualization technologies in the 1990s. VR technology today not only presents to users but also *immerses* them with a computer generated virtual world. Extended from VR, AR allows users to interact with a real world superimposed with computer graphics. Hence the combined VR and AR world would appear to the users as if both the real and the virtual objects coexist. Modeling for VR and AR should enable real-time performance and vivid presentation, requiring that the object's shape, texture, colours and physical parameters be registered in a unified representation.

3.1.2 Object Modeling Methodologies

Continuous Modeling. Continuous modelling approximates the entire or a functional part of 3D objects by geometrical primitives (blocks, polyhedrons, cylinders or superquadrics). Kinematics and physical features can be attached to the primitive set. For example, Barr “borrowed” the techniques from linear mechanical analysis to approximate 3D objects [1] using angle-preserving transformations on superquadrics. Terzopoulos *et. al.* defined *deformable superquadrics* [12] with a physical feature based approach: in animation, the behaviour of the deformable superquadrics is governed by motion equations based on physics; in model construction, a model is fitted with 3D visual information by transforming the data and simulating the motion equations through time. It is easy to detect, attach and apply geometric, kinematic and physical parameters to the continuously modelled objects. Nevertheless, it is still difficult to mimic behavioural features since the model lacks a symbolic structure to fit in or to articulate the behavioural languages. For many real world objects, approximation by pre-defined primitives is very difficult if not impossible.

Discrete Modeling. A variety of computer vision applications involve highly irregular, unstructured and dynamic scenes characterized by sharp and non-uniform variations in irregular spatial features and physical properties. Discrete modeling is able to approximate this kind of objects by large patches of simple

primitives, such as polygons or tetrahedrons. Physical and kinematical features can be flexibly associated with one or a group of primitives. Triangular mesh is the most popular discrete modeling method. The abundance of algorithms to manipulate triangular meshes encourages and facilitates their use in many vision and graphics applications [4][6]. Behavioural features such as facial expressions and emotions can be modelled by attaching physical constraints on triangular meshes [13]. Its main drawback is the lack of a structure to perform symbolic operations. Furthermore, the mesh primitives are unstructured and could capture only local features instead of higher level information.

Graph Based Symbolic Modeling. In this approach, a complex object is usually represented by a set of *primitives* and the *relations* among them in the form of a graph. If the primitives are blocks such as cylinders, it is a clone of continuous modeling. If the model consists of a vast number of primitives (e.g., polygons), it is an extension of discrete modeling. The graph representation was first introduced for the script description of a scene in AI [11]. In 3D model synthesis, random graphs were applied to tackle the uncertainties in image processing [15]. In [14], attributed hypergraph model was constructed based on model features and used for 3D recognition and pose determination. Since graph based modeling approaches introduce the concepts of *primitives* and *relations*, it is straightforward to build a hierarchical representation. However, they need further improvements to address the problems in representing highly dynamic and deformable objects.

3.1.3 Problem Definition

Most existing 3D vision systems lack the capability to operate at symbolic or knowledge level. It is hard to *learn* unknown patterns from the sensed data. For computer vision and animation, an AHR is ideal as it is generic and effective for manipulation. Our research attempts: 1) to establish a mathematical framework for AHR; 2) to explore the algorithms that build, manipulate and augment AHRs and 3) to develop an algorithm that controls and generates 3D animations. We base our AHR 3D modelling methodology on triangular mesh approximation. A net-like data structure is then designed to handle the dynamic changes in the representations to overcome the inflexibility of the graph structure.

3.2. Attributed Hypergraph Representation (AHR)

Here, AHR formalism based on category theory is introduced to provide a generalized framework for object modeling, transformation and manipulation. The category framework, embodied the concepts of objects, transformations, invariance and equivalence, allows patterns to be represented with a more general

and elegant mathematical formalism. Category theory also helps to establish links among different notions or aspects of the same problem.

3.2.1 Category Theory and Graph Based Representation

The theory of categories constitutes an autonomous branch of mathematics and has found many applications [10]. Based on a categorical framework, we establish the AHR for 3D object modelling from a mathematical perspective.

Notations and Definitions:

Definition 3.1. A Category C is formed if a class of elements, called objects, is given such that:

- For each pair of objects (O_1, O_2) in C , a map u , called a *morphism* (denoted as $O_1 \xrightarrow{u} O_2$) is given;
- For each object O in C , there is an *identity* morphism I_O , such that, if $O \xrightarrow{I_O} O'$, then $O = O'$;
- The *composition* (denoted as $\otimes$) of maps of morphisms satisfy:
 - 1 If $O_1 \xrightarrow{u} O_2 \xrightarrow{v} O_3$ then $v \otimes u$ is the map that maps O_1 to O_3 , i.e., $O_1 \xrightarrow{v \otimes u} O_3$. If $O_1 \xrightarrow{u} O_2 \xrightarrow{v} O_3 \xrightarrow{w} O_4$ then $O_1 \xrightarrow{v \otimes u} O_3 \xrightarrow{w} O_4$ and $O_1 \xrightarrow{v} O_2 \xrightarrow{w \otimes v} O_4$
 - 2 If $O_1 \xrightarrow{u} O_2$, identity maps always compose to give $I_{O'} \otimes u = u$ and $u \otimes I_O = u$

The categories commonly seen in mathematics have objects that are structured sets with morphisms relating them. For example: *Set*, *Grp* (groups and group homomorphisms) and *Gph* (graphs and the graph morphisms).

Definition 3.2. A morphism u from O to O' , denoted as $O \xrightarrow{u} O'$, is called a *retraction* if for each entity in O' , there is a unique non-null entity in O that corresponds to it by u . A morphism u from O to O' is called a *coretraction* if for each entity in O , by u there is a unique non-null entity in O' that corresponds to it. A morphism u is an *isomorphism*, if it is both a *retraction* and a *coretraction*, and then O is said to be *isomorphic* to O' , denoted as $O \sim O'$. If $O \sim O'$ and $O' \sim O''$ then $O \sim O''$.

Definition 3.3. A *covariant functor* F from category C_1 to category C_2 is defined as a pair of maps $F = (F_{obj}, F_{mor})$: objects of $C_1 \xrightarrow{F_{obj}}$ objects of

C_2 ; morphisms of $C_1 \xrightarrow{F_{mor}} \text{morphisms of } C_2$; or simply $C_1 \xrightarrow{F} C_2$. Usually we refer to a covariant functor as a functor.

Figure 3.1 gives an example of a functor. G_1 and G_2 are two objects in Gph , and a morphism u is defined such that $G_1 \xrightarrow{u} G_2$. A covariant functor $F = (F_{obj}, F_{mor})$ is defined as the mapping from Gph to Set . We have $G_1 \xrightarrow{F_{obj}} S_1$ and $G_2 \xrightarrow{F_{obj}} S_2$. Corresponding to u , the morphism between S_1 and S_2 is F_{mor} .

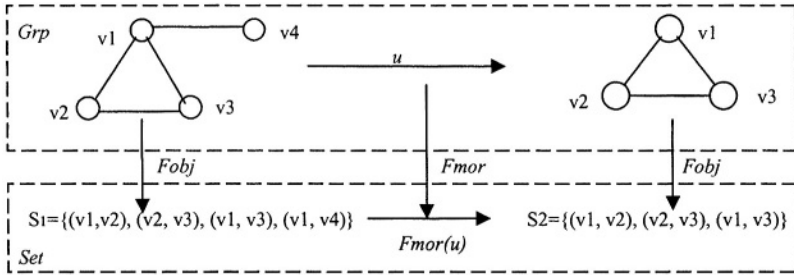


Figure 3.1. An example of a function that maps graph category to set category.

Definition 3.4. If a functor defined on C_1 and C_2 preserves all retractions (coretractions, isomorphisms) from C_1 to C_2 , it is called *retraction preserving* (coretraction preserving, isomorphism preserving) functor.

Definition 3.5. The composition of two functors F_1 on categories C_1, C_2 ($C_1 \xrightarrow{F_1} C_2$) and F_2 on categories C_2, C_3 ($C_2 \xrightarrow{F_2} C_3$) is defined as the functor $F_2 \otimes F_1$ on C_1 and C_3 , such that $C_1 \xrightarrow{F_2 \otimes F_1} C_3$.

Definition 3.6. A graph G is an ordered pair (V, E) where $V = \{v_k \mid 1 \leq k \leq n\}$ is a set of n vertices, and

$$E = \{e_{i,j} \mid e_{i,j} = (v_i, v_j), 1 \leq i \leq n, 1 \leq j \leq n\}$$

is a set of edges. Each $e_{i,j}$ in E relates two vertices v_i and v_j of V .

Definition 3.7. A graph S is a *subgraph* of G (written as $S \subseteq G$) if $V_S \subseteq V_G$ and $E_S \subseteq E_G$. If $S \subseteq G$, G is called a *supergraph* of S .

Definition 3.8. A subgraph S of graph G is *detached* in G if there is no vertex in S adjacent to any vertex of G is not in S ; a non-null graph G is *connected* iff there is no non-null subgraph of G that is detached in G .

Definition 3.9. An *attributed graph* G is a graph defined as an ordered pair $G = (V, E)$ associated with A_V and A_E where G is a graph, A_V is a finite set of vertex attributes and A_E is a finite set of edge attributes. Both are defined on continuous domains. Each vertex in V assumes values of attributes from A_V , and each edge in E assumes values of attributes from A_E .

Definition 3.10. A *hypergraph* is an ordered pair $G = (X, Y)$, where $X = \{v_i \mid 1 \leq i \leq n\}$ is a finite set of vertices and $Y = \{H_j \mid 1 \leq j \leq m\}$ is a finite set of hyperedges; each H_j is a subset of X such that $H_1 \cup H_2 \cup \dots \cup H_m = X$.

Definition 3.11. An *attributed hypergraph* (AH) G is a hypergraph defined as an ordered pair $G = (X, Y)$ associated with A_X and A_Y , where $G = (X, Y)$ is a hypergraph, A_X and A_Y are finite sets of hyperedge attribute respectively. Each vertex in X and each hyperedge in Y may assume values of attribute from A_Y respectively.

3.2.2 Category Theory and Graph Theory for 3D Object Modeling

Early object representations in computer vision and graphics mainly focused on geometrical features. With kinematical constraints attached, such representations were able to analyze motions. Then systems based on physical constraints were developed to solve animation and vision problems in highly deformable situations. The geometrical and physical features adequately characterize the corresponding objects for modeling and manipulation. However, they lack the capability to represent more complex features such as the behavioural ones due to the absence of high level structures. The graph based approaches are designed to solve this problem by mapping the primitive features to vertices and their relations to edges. Symbolic processing on graphs is possible and a few graph based algorithms are effective for common tasks such as searching, traversing and matching.

In the view of category theory, object representation and transformation are equivalent to applying functors on a category. The geometric, physical and graph representations constitute three different categories:

- The geometrical spaces, with geometrical descriptions and the kinematic transformations defined on it, comprise the geometrical category (*Geo*);
- The physical states identified by sets of physical parameters and the state transition functions make up the physical category (*Phy*);
- The graph and the graph morphisms form the graph category *Gph*.

With the concept of functor defined in the scope of computer vision and graphics applications, the graph based representation can be viewed as the abstractions from geometrical and/or physical representations by applying the functors that map *Geo* and *Phy* to *Gph* respectively using categorical languages. The benefits of using category theory on 3D object modelling are:

- It provides a theoretical background for building a generic framework;
- It unifies spatial and algebraic approaches in object modelling;
- The formalism creates links among different aspects of modelling;
- It gives a general formalization of computational process using functors.

However, category theory cannot by itself solve modelling problems in an application domain. Yet, it does render a particular methodology to formalize structure, pattern and shape. It should be adjoined with other mathematical disciplines for defining objects and transformations in an application domain. In this chapter, AHR is used to formulate 3D object modelling, whereas notions in category theory are adopted as the language in the formalization.

3.2.3 Data Structure of Attributed Hypergraph Representation

The Dynamic Structure of AHR. Traditional graph based modeling has a drawback that their graph structures are too rigid. Hence, they are not very appropriate for modeling highly deformable objects. We implement AHR with a net-like dynamic data structure (Figure 3.2) and call it a *dynamic hypergraph* (DAH). In a hypergraph, a subgraph is also a hypergraph. A subset or a superset of a hyperedge is also a hyperedge. Likewise, a hypergraph or any of its subgraphs can be represented by a collection of hyperedges. This property enables us to define a unified data structure for different types of entities in a hypergraph. In the net-like data structure as illustrated, the basic element is called a node (shown as a rectangle). The nodes on the bottom layer represent the vertices; those on the intermediate layers are the hyperedges which are supersets of the nodes on their lower layers; and the node on the top layer (called the *root*) represents the entire hypergraph. There are three types of directional links between the nodes. Links

from the nodes on the lower layer to the higher layer ones represent the relation *subset-of*; those from the higher (upper) layer to the lower layer ones have the relation *superset-of*; and those between the nodes on the same layer are the adjacency relations. If for each node there is an attribute associated with it, the hypergraph becomes an AH. In DAH, the structural changes on AHs can be handled efficiently. For example, the join of two hyperedges is simply the combination of the two corresponding nodes and the re-organization of the associated links. Graph based computations such as the re-organization of a hyperedge based on certain criteria can be performed by reshuffling the links among the nodes. In a DAH, the nodes on the bottom layer and the links among them construct an *elementary graph*.

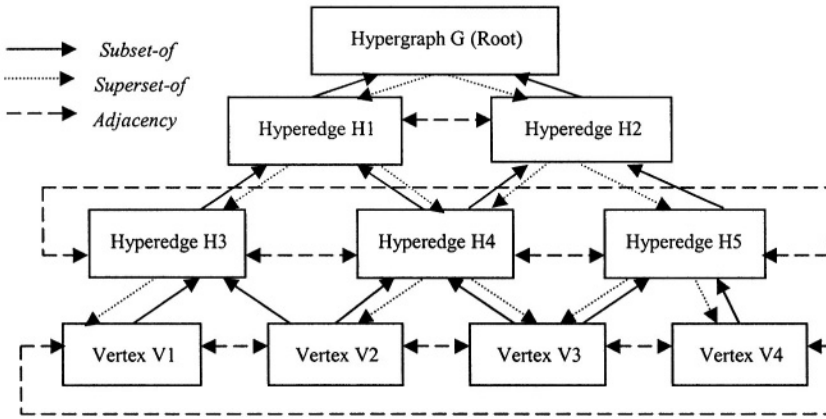


Figure 3.2. The net-like structure for dynamic attributed hypergraph with nodes and links.

Definition 3.12. In a DAH, the nodes on the bottom layer are called the *elementary nodes*; the links among the elementary nodes are called the *elementary edges*. The attributed graph which consists of the elementary nodes and the elementary edges is called the *elementary graph* of the DAH.

If a DAH has n layers, and the node set on layer i is X_i , then DAH $G = (X, Y)$ can be written in the form of $G = (X_1, X_2, \dots, X_n)$ where $X = X_1$ and $Y = (X_2, X_3, \dots, X_n)$. The meanings of the nodes on the intermediate layers depend on the applications. For example, in geometrical modeling, we can have two intermediate layers based on elementary graph:

- The elementary graph characterizes the basic features extracted from the sensory data such as corner points and lines;

- The hyperedges on the first intermediate layer represent the organizations of the bottom layer nodes, forming surface patches;
- The hyperedges on higher layers represent higher level knowledge;
- The root node represents the entire hypergraph.

The elementary graph is critical for a complete and precise representation since it provides the basis to abstract the information required by the nodes on higher layers. We use the triangular meshes to create the elementary graph.

Triangular Meshes of Range Data. The bottom layer of a DAH needs to satisfy two requirements simultaneously: 1) the ability to represent subtle features for general 3D patterns for reconstructing a complete object; and 2) the suitability to integrate the representation into a structural framework. Therefore it is necessary to compile and abstract the raw imaging data. It is ideal to use a mesh based representation due to the unstructured nature of general 3D shapes. We adopted the triangular mesh algorithm proposed in [4].

Hypergraph Based on Triangular Mesh. With triangular meshes, an object can be approximated using its salient geometrical or other extracted features. The structure of the mesh fits well in graph based representations. In the vertex-edge structure, a triangular mesh τ is in the form of $\tau = (V_t, E_t)$ where V_t is the vertex set and E_t is the edge set of the mesh.

Definition 3.13. A representative attributed graph (RAG) $G = (V, E)$ of a triangular mesh τ is an attributed graph constructed in the following manner:

- For each vertex $v_t \in V_t$, there is a vertex $v_a \in V$ corresponding to it;
- For each edge $e_t \in E_t$, there is an edge $e_a \in E$ corresponding to it;
- v_t 's features are mapped to v_a 's features and e_t 's features are mapped to e_a 's features.

In the net-like DAH data structure (refer to Figure 3.2), the elementary nodes on the bottom layer consist of the vertices in the RAG, and the elementary edges are copied from the edges in the RAG. The surface's properties attached to the mesh, such as pose, area, colour, texture, etc. are mapped to the attributes associated with the vertices or edges. Thus, the RAG of the mesh directly constitutes the elementary graph of the AH.

Definition 3.14. Suppose that $X = (x_1, x_2, \dots, x_n)$ is the vertex set of hypergraph $G = (X, Y)$, $E = \{x_{e_1}, x_{e_2}, \dots, x_{e_m}\} (E \subseteq X, e_k \leq n, k = 1, 2, \dots, m)$ is a hyperedge, and $\{a_{e_1}, a_{e_2}, \dots, a_{e_m}\}$ are the vertex attribute sets associated with vertices $\{x_{e_1}, x_{e_2}, \dots, x_{e_m}\}$ respectively. We say that E is a hyperedge induced by attribute value a , if for the selected attribute value a , we have:

- $a \in a_k$ the corresponding vertex $x_k \in E$;
- $a \notin a_k$ if the corresponding vertex $x_k \notin E$;
- E is connected.

Different from classical hyperedges, hyperedges in a DAH can be generated from the topologies of the nodes on any of their lower layers. In a triangular mesh based DAH, a hyperedge can be a collection of triangles in the RAG and represent the organizations of those triangles based on certain constraints. Figure 3.3 (a) shows a sample 3D scene with simple triangles as its surfaces. The AHR generated for navigation guidance in a 3D scene is illustrated in Figure 3.3 (b). For simplicity, it only shows the elementary graph and the hyperedges (as the dashed closed loops). A navigable area is defined by a hyperedge which contains the floor area.

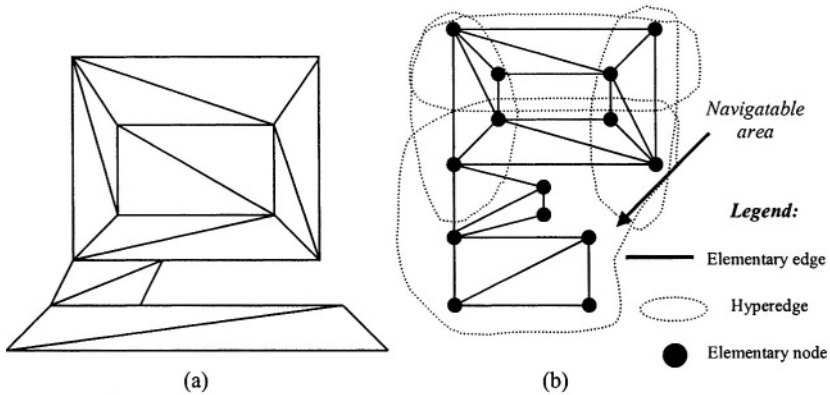


Figure 3.3. (a) A simple 3D indoor scene; (b) the AHR for vision guided navigation.

Another example of hyperedge construction is texture binding for augmented reality (AR). As shown later, to from an AR from real world scenes and virtual elements, natural looking textures have to be extracted from 2D views and then transposed to 3D surfaces. Thus, the features to construct the hyperedges are the

surface textures and connectivity properties of the triangles. Figure 3.4 (a) shows a similar 3D scene as the one in Figure 3.3 but with textures mapped to the triangles. Figure 3.4 (b) shows the corresponding AHR for texture binding. The hyperedges are different from those in Figure 3.3 (b). With the AHR, we can change the wall paper of the room by simply changing the attribute value of the corresponding hyperedge. More details about attributed hyperedge generation will be discussed later.

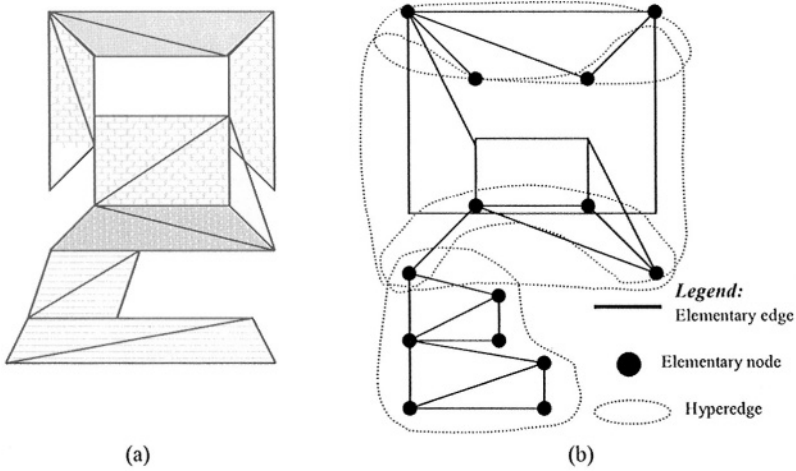


Figure 3.4. (a) A textured indoor 3D scene. (b) The AHR for texture binding in AR.

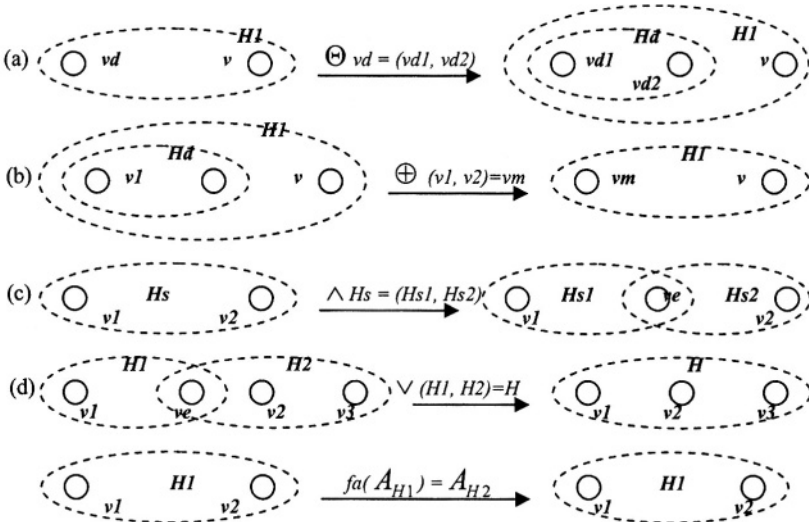


Figure 3.5. The primary operators for an attributed hypergraph: (a) dichotomy; (b) merge; (c) subdivision; (d) join; (e) attribute transition.

Primary Operators on Hypergraph. Corresponding to the motions of a 3D pattern, the transformations on its AHR are represented by a series of operators. Besides the commonly known operators of *union* and *intersection*, Figure 3.5 depicts the operators of *dichotomy*, *merge*, *subdivision*, *join* and *attribute transition*. Compound operators (Figure 3.6) can also be defined. From category theory, primary operators and their compositions constitute the morphisms between different AHRs in the AH category *Ahp*. The functor then corresponds to the motions of the 3D patterns.

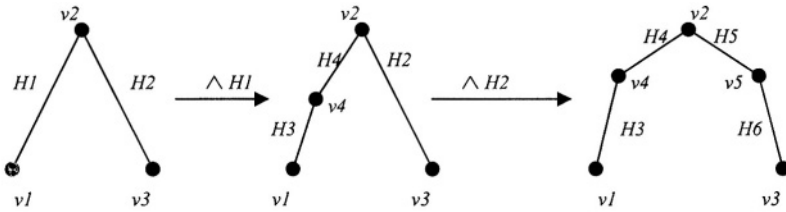


Figure 3.6. A compound operator defined on a simple graph.

3.3. 3D Object Modeling using AHR and AH Operators

Three levels of information are involved in the AHR based modeling methodology: a) *Realization level* is the visual level depicting the actual scene. It is the platform where the reconstructed 3D scene is rendered and back-projected, b) *Feature level* has the features which are used to extract symbolic information, or to reconstruct 3D scenes through the back-projection to the realization level, c) *Symbolic level* is the representation level with symbolic information to form an attributed hypergraph. Figure 3.7 illustrates the relations among the three levels. The use of AHR enables us to manipulate 3D data focusing only on the selected feature types on higher levels. At the representation level instead of the realization and feature levels, transformations are in the form of AH operators on the AHR to represent various changes of the scene, such as augmentation and animation.

When transformations are applied over an AHR, kinematic features and physical features are sufficient to characterize the rigid or non-rigid motions of the object. Based on the kinematic and physical attributes, we further introduce the behavioural attributes, a high-level feature type to model constraints, ego-motions and other intelligent behaviours of the objects. The kinematic and physical attributes are extracted directly from the triangular mesh, while the behavioural attributes are abstracted from the kinematic and physical attributes. The attributed

hypergraph is constructed from the triangular mesh together with all three types of attributes. In the following, we provide the details of the 3D object modelling.

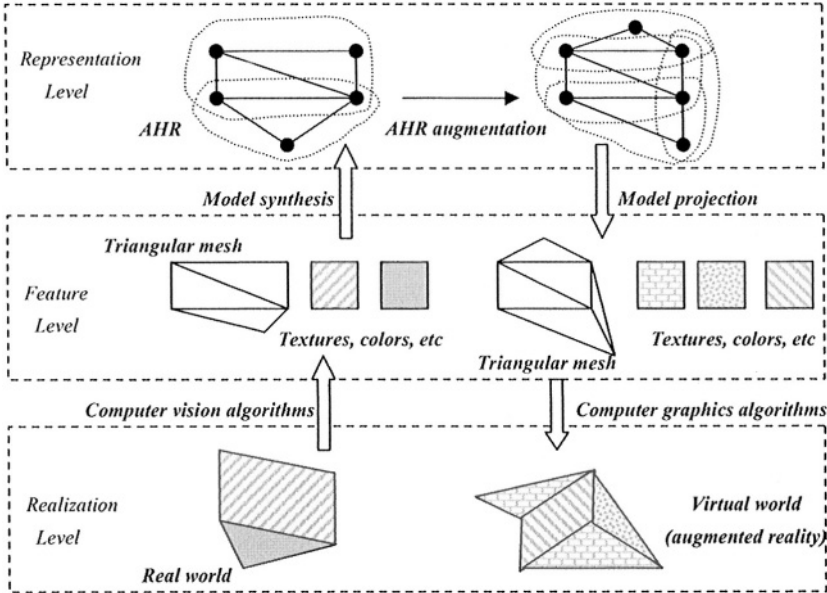


Figure 3.7. The relations among realization level, feature level and representation level.

3.3.1 Modeling of Geometrical Shapes

Modeling geometrical shapes from triangular meshes is straightforward:

Realization Level. The surfaces of a 3D object are approximated by different patches in triangular meshes: each patch itself constitutes a mesh with 2.5D range data. For rendering, they are back-projected onto the viewing plane after being fitted by the surface spline functions [3].

Feature Level. At this level, relevant information is the geometrical features extracted from the meshes. The detection of the geometrical features is by studying neighbouring triangles or neighbouring patches of triangles and the common edges/boundaries among them. Common features include edges, corners, ridges, ravines, peaks, pits and saddles.

Symbolic Level. Task dependent symbolic information about 3D shapes can be extracted. For example, for vision guided navigation purpose, such symbolic

information will include a conceptual map built upon the ridges, ravines, peaks, pits and saddles at the feature level. Figure 3.8 illustrates an example.

In Figure 3.9 (a), the geometric model of a simple terrain is approximated by a triangular mesh consisting of a tetrahedron (hill) and two triangles (plain). Figures 3.9 (b) and (c) show the corresponding AHR and DAH respectively.

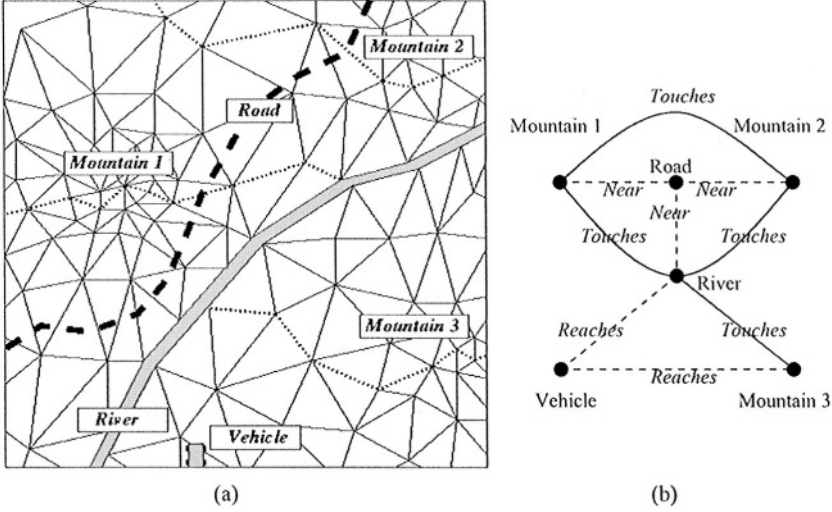


Figure 3.8. (a) A sample outdoor scene. (b) The corresponding conceptual graph.

3.3.2 Modeling of Kinematic Transformations

Kinematic transformations include rotation, translation, scaling and their combinations. If before and after the transformations the point sets are expressed in a fixed homogeneous coordinate system, kinematic transformations can be expressed in the form of matrix multiplications. Translation, rotations, and scalings are achieved by applying a 4×4 scaling/translation joint matrix T_s and a 4×4 rotation matrix R . Since kinematic transformations are about the location changes of an object, they do not introduce structural change in AHR when information is abstracted from the realization level. The transformations apply only to geometric entities such as lines, corners, triangles or patches. In AHR, they apply to the geometric attributes of vertices, hyperedges. With the definition of $F_g = (F_{g-obj}, F_{g-mor})$ that maps T_s and R to the morphisms in the Ahp , we have: $T_s \xrightarrow{F_{g-mor}} f_a$ where f_a is the attribute transition operator set corresponding to the translation and the scaling factors; $R \xrightarrow{F_{g-mor}} o_r$ where o_r is the attribute transition operator set that applies to the attributed vertices and hyperedges corresponding to the object rotation.

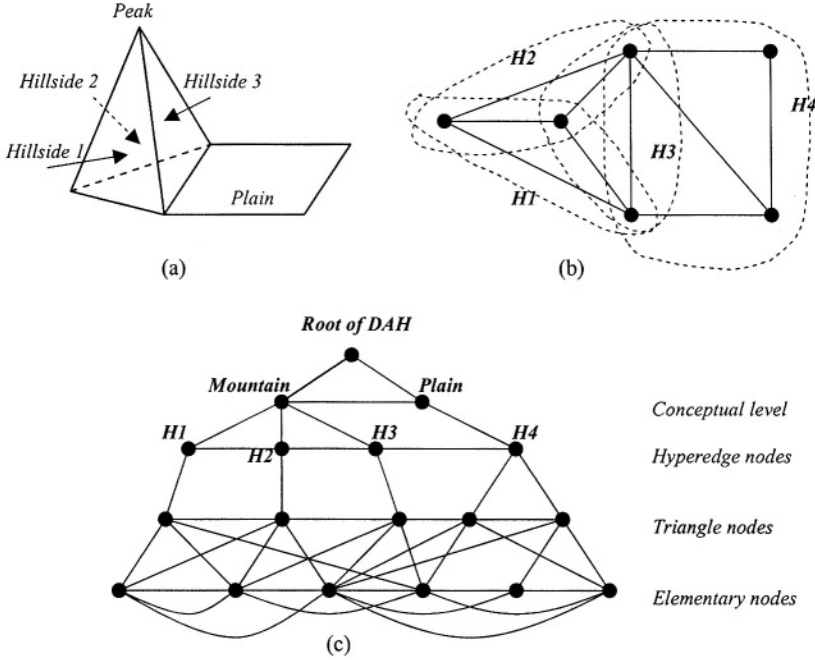


Figure 3.9. An example of the DAH representation: (a) a natural terrain with a hill and a piece of plain land; (b) the attributed hypergraph representation; (c) the DAH structure.

3.3.3 Modeling of Physical Transformations

Physical feature based representations describe deformable objects by physical parameters and state transitions. They use physics laws and constraints to govern the motions. The shape/location changes of an object are calculated by simulations in physics. It does not change the representation forms across representation levels during information abstraction. The object's physical state (in the form of physical parameters) at time t can be written as $\mathcal{S}_t$, and the state transition function is f . Given the initial state $\mathcal{S}_0$, function f calculates $\mathcal{S}_t$ where $t > 0$. For elastic objects, the form of f is one or a set of partial differential equations [2].

A 4-level DAH $H = (X, H_{base}, H_t, H_p)$ for the physical description is formed by the definition of the functor $F_p = (F_{p-obj}, F_{p-mor})$ which maps the physical configurations from triangular mesh to the AH category. Suppose that the triangular mesh for finite element analysis is in the form of its RAG

$G = (V_t, E_t)$ where the vertex set V_t corresponds to the vertex set in the mesh, and edge set E_t corresponds to the edge set in the mesh. F_p is defined as:

- 1 Elementary nodes: $v_t \xrightarrow{F_p} v_p$ where $v_t \in V_t, v_p \in X$
- 2 Elementary edges: $e_t \xrightarrow{F_p} h_{base}$ where $e_t \in E_t, h_{base} \in H_{base}$
- 3 Triangles: $t_t \xrightarrow{F_p} h_t$, t_t is a triangle in the mesh and $h_t \in H_t$
- 4 Hyperedges: $S_t \xrightarrow{F_p} h_p$ where S_t is a surface patch, $h_p \in H_p$
- 5 Operators:
 $(V_t(t_0), D(t_0, t_1, \dots, t), V_t(t) \xrightarrow{F_p} (X(t_0), (o(t_0), o(t_1), \dots, o(t)), X(t))$
 where $D(t_0, t_1, \dots, t)$ is the state transition function f with initial state at t_0 and evaluated at $t_0, t_1, \dots, t$, $V_t(t_0)$ and $V_t(t)$ are the states of the vertex set V_t at time t_0 and t respectively, $X(t_0)$ and $X(t)$ are the attributed vertex sets of the DAH at time t_0 and t respectively, $(o(t_0), o(t_1), \dots, o(t))$ is the set of AH operators that are applied to $X(t_0)$ and transform it to $X(t)$.

3.3.4 Modeling of Behavioural Transformations

In an AHR, the intrinsic data types are entities and their relations. The object's geometrical, kinematic and physical features are characterized by entities with attributes in an AHR. Hence, the "knowledge", "constraints" or "intelligence" of this object can be interpreted by high-order relations among these entities. In a graph language, these relations can be expressed by:

- Attributed vertices, edges and hyperedges that characterize geometrical, kinematic and physical features;
- Associations of the above entities in the form of high level hyperedges;
- Operators applied to the above attributed graph entities; and
- Rules defined on the compositions of the operators.

The corresponding mapping functor F_b which forms the behavioural factor representation from the above features in AH category can be constructed by process abstraction from the geometrical and physical mappings to a definite operator sequence $(o_1, o_2, \dots, o_n)$ where $o_i (1 \leq i \leq n)$ is either an attribute transition on AH (e.g., f_a or o_r) or a primary operator on AH (o_p).

3.4. Augmented Reality using AHR

AHR provides an efficient and practical methodology for object modeling. Augmented Reality is an excellent example showing its flexibility: it first employs computer vision to estimate the 3D geometry of real scenes, and then facilitates computer graphics to augment the synthesized models and present them as in VR.

3.4.1 Recovery of 3D Geometry

The recovery of 3D geometry begins with 2D imaging processing. We adopted the 2D feature extraction method describe in [7] and the camera calibration algorithm given by [5]. Correspondences among the 2D features in different calibrated images are established through the guidance of: (1) the epipolar line geometry; (2) the structural characteristics of the feature groupings, and (3) the probability of coincidence of the feature groupings among the images. Then we apply the triangulation algorithm from [8] to construct a collection of 3D features. With the 3D features (corners and lines), the most straightforward method to represent a 3D object is the use of the well-known *wire-frame* model. A wire-frame is a 3D object model consisting of the surface discontinuity edges. Considering building a triangular mesh as the bridge between descriptive shape modeling and symbolic AHR, a more generic dual of wire-frame model is the face modeling, in which the elements are the surface patches bounded by closed circuits of wire-frame edges. The collection of faces forms a complete 3D surface boundary of the object. It can be easily converted into coarse triangular meshes, and then to AH's by the steps described previously.

3.4.2 Construction of Augmented Reality using AHR

To build an augmented scene, virtual object/scene models have to be combined with real object models constructed by 3D geometry recovery. Up to now, range data have been modelled in the form of attributed hypergraphs. They will be incrementally synthesized into an integrated AHR, and then augmented with the AHRs representing the virtual objects/scenes.

Synthesis of Two Attributed Hypergraphs. The synthesis of two AHRs $G_1 = (X_1, Y_1)$ and $G_2 = (X_2, Y_2)$ into $G_r = (X_r, Y_r)$ is a two-stage process. First, hyperedges in Y_1 and Y_2 are considered. For two hyperedges $e_p \in Y_1$ and $e_q \in Y_2$, if they correspond to the surface patches on the same plane, and touch, contain or overlap with each other, they are integrated into one patch by averaging their attribute values and composing the adjacency properties. For each hyperedge in G_1 the comparison is performed to search for its counterpart in G_2 . After the first stage, there may be triangles that are identical, overlapping, one

containing the other. In such cases, the vertices and edges have to be re-calculated, otherwise, they are considered as new ones and simply copied to X_r . The adjustment of the vertices has several possibilities:

- If the two triangles to be compared are identical, copy one to X_r ;
- If one of the two triangles is found to contain (or to be contained by) the other by comparing the attribute values, then copy the vertices of the former (or the latter) to X_r and the latter (or the former) is ignored;
- If the two triangles are overlapping, then:
 - 1 Establish one-to-one correspondences between the three triangle points between the two triangles;
 - 2 Calculate the average point for each pair of matched triangle points;
 - 3 Construct an average triangle, whose points are the average points;
 - 4 Copy the three vertices of the average triangle to X_r .
- Update the edges and the hyperedges in Y_r according to the new X_r set.

Incremental AHR Synthesis. The above synthesis process considers only two AHRs. With multiple views, such synthesis can be repeated and a final model is constructed incrementally. Suppose that we have n range models in the form of AHRs: $G_1, G_2, \dots, G_n$. We can conduct the synthesis for $n - 1$ times on the consequent images, and incrementally construct a 3D AHR. The possibly redundant information in these AHRs from different views enables us to compensate for system errors, recover missing features and eliminate noise. The incremental AHR synthesis consists of the following steps:

- 1 Let $i = 1$, set AHR G_a empty;
- 2 Match the two attributed hypergraph G_i and G_{i+1} , obtain a synthesized G_{r_i} ;
- 3 Compare G_{r_i} with G_a ;
- 4 If a new vertex's geometrical attributed value is close enough to one that is already in G_a , they are considered the same vertex. The average between them replaces the old one and the score for this vertex is increased by 1;

- a) If no vertex in G_a is close to the new vertex, add it to G_a and set its score to 1;
 - b) adjust the edges and hyperedges that contain the modified vertex;
- 5 Let $i = i + 1$, if $i < n$, go to step 2
 - 6 Eliminate the vertices in G_a with scores lower than the preset threshold;
 - 7 Adjust edges and hyperedges of G_a after noisy vertex elimination.

The result of the synthesized 3D scene is an integrated AHR.

AHR Augmentation. At the representation level, the augmentation of a 3D scene is equivalent to the augmentation of the corresponding AHR by applying a proper set of AH operators. It may have qualitative changes on the topology of the hypergraph or quantitative changes on the attributes.

Virtual Objects Integration. At the AHR level, the integration of a virtual object with the 3D scene constructed from sensory data (or vice versa) is to combine two or more AH's. The operators *union* and *intersection* can be directly applied to two AHRs. Suppose that the AHR of the real scene is G_r and the AHR of the virtual part is G_v , the combined AH G is obtained by:

- 1 Calculate $G_u = G_r \cup G_v$ and $G_i = G_r \cap G_v$;
- 2 Let $G_o = G_u \setminus G_i$ (subtraction of graphs);
- 3 Align the attributes of all entities in G_i ;
- 4 Set $G = G_o \cup G_i$

The "alignment" of the attributes of two AH's depends upon the application context. Normally, it only involves translations and/or rotations by *attribute transitions* on the AH's for proper positioning. However, qualitative changes of the AHR are possible. In object fusion, the integration of the two AH's G_r and G_v has to be followed by eliminating all inner surfaces/vertices to ensure the integrity. The following steps remove the inner vertices and surfaces;

- 1 Calculate G which is the integration of G_r and G_v (with possible inner vertices and/or surfaces), and $G_i = G_r \cap G_v$;

- 2 If a vertex $v_k \in G_i$ is an inner vertex:
 - a) find a vertex $v_l \in G$ such that v_l is adjacent to v_k and the discrepancy between v_l 's attribute value Av_l to v_k 's attribute value Av_k is the smallest among all vertices adjacent to v_k ;
 - b) in G , apply *merge* operator on v_k and v_l ;
- 3 If a hyperedge $H_k \in G_i$ becomes an inner hyperedge:
 - a) find a hyperedge $H_l \in G$ such that H_l is adjacent to H_k and the discrepancy between H_l 's attribute value Ae_l to H_k 's attribute value Ae_k is the smallest among all hyperedges adjacent to H_k ;
 - b) in G , apply *join* operator on H_k and H_l ;
- 4 If there is no inner vertex or hyperedge, exit, otherwise go to step 2.

Texture and Colour Mapping. Natural looking surface textures and/or colours are important for the construction of augmented reality. A texturing/colouring algorithm is adopted from [9] to estimate the surface appearance for each 3D triangle in the mesh from the available 2D views. The surface texture (or colour) binding algorithm has the following modules: 1) 3D texture (or colour) region grouping; 2) 2D projection selection; 3) Boundary processing; and 4) Texture synthesis for occluded region.

3.5. Experiments of Modeling and Augmented Reality

An AHR based 3D modelling system has been implemented.

CAD Model Synthesis of a Simple 3D Object. The first experiment on model synthesis was carried on with a simple aluminium object called a "bridge". Twelve images were taken from different vantage points around the object under normal lighting condition. 2D visual features, such as lines, corners and ellipses are extracted. Figures 3.10 (a) through (c) show three of its twelve views with feature detection results, and Figures 3.11 (a) through (c) illustrate model synthesis results from the images acquired at the second, fifth, and the final views respectively. The incrementally synthesized model is depicted by the thick white lines, as the back-projection of the partial 3D CAD model onto the images. A complete model is then constructed (Figure 3.11 (c)).

3D Indoor Scene Reconstruction. The second experiment was to simulate the stereo vision system used in *vision based autonomous rover navigation*. A

colour CCD camera was placed at two vantage points and two images of a table were captured (as shown in Figures 3.12 (a) and (b)). The raw 2D features extracted are corners and lines (Figures 3.13 (a) and (b)). The corners of the table surface were measured manually and used as the landmark for camera calibration. The 2D corner features of the table are adequate for camera pose determination. Figure 3.14 shows a partial table model constructed from the corner/line features back-projected to the left image. Without information from other views, the model is not complete but is sufficient to build a partial AR or to be used as a landmark in other 3D reconstruction experiments for scenes containing this table.

A third experiment on 3D scene analysis involves a laboratory scene. The left and right images are shown in Figures 3.15 (a) and (b) respectively. Figure 3.16 gives the feature extraction results. Key features which are crucial to rebuild the room, such as the corners and lines, are selected to construct the partial CAD model, while detailed structures of the furniture in the corners and objects outside the door are ignored. Figure 3.17 back-projects the constructed partial CAD model to the left image.

Augmented Reality with the Laboratory Table. Figures 3.18 (a) through (f) demonstrate another AR experiment involving the table. Figure 3.18 (a) shows the triangular mesh generated from Figure 3.14. The corresponding AHR is in Figure 3.18 (b). Here, the rectangles signify the triangles and colours represent the triangles' orientations. Figure 3.18 (c) shows the reconstructed object from the AHR. The colours and textures of the table are also presented. In Figure 3.18(d), an arm chair, a table lamp and a piece of textured floor are added in the scene. Figure 3.18 (e) gives the view with the table top enlarged. The texture of the table top was then changed to marble and the illumination effect of the table lamp on all objects is simulated as shown in Figure 3.18 (f)).

Augmented Reality with the 3D Indoor Scene. As with the table, the same experiment scheme is also applied for the 3D reconstruction of the laboratory scene. Figure 3.19 (a) shows the triangular mesh generated from the face model given in Figure 3.17. The corresponding AHR is in Figure 3.19 (b). In the AHR, the green nodes connect each other by the green links to form the floor plane. It is the "navigable" area in the scene which could be used for path planning. Figure 3.19 (c) shows the reconstructed scene with textures and colours inverted from the original stereo images. Figures 3.19 (d) to (f) demonstrate three augmented views of the laboratory scene. They include: (1) the incomplete table replaced by a full table model; (2) the floor plane expanded outside of the door; and (3) the changed floor textures. For example, in Figures 3.19 (d), the floor texture is changed, graphics of a tetrahedron, a robot and a door are placed in the scene. Figure 3.19 (f) is built from the AHR shown in Figure 3.19 (b) with only the geometric properties

preserved. The surface colours are all arbitrarily assigned, and two virtual lighting sources are posed.

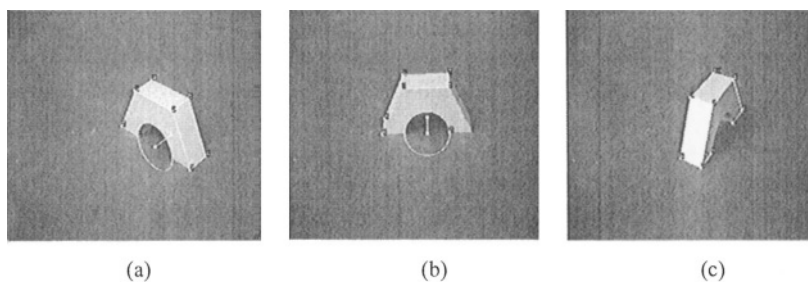


Figure 3.10. The images and feature detection results of 3 different views of the “bridge” object.

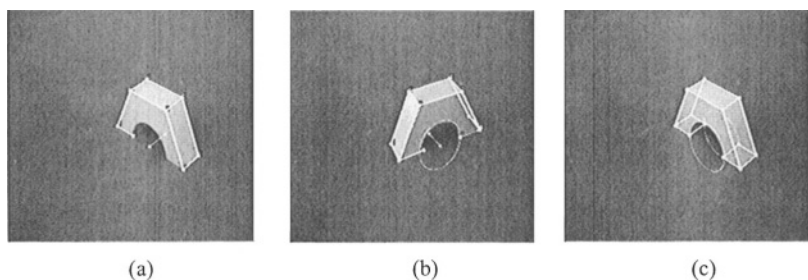


Figure 3.11. The incremental model synthesis results for the “bridge” object.

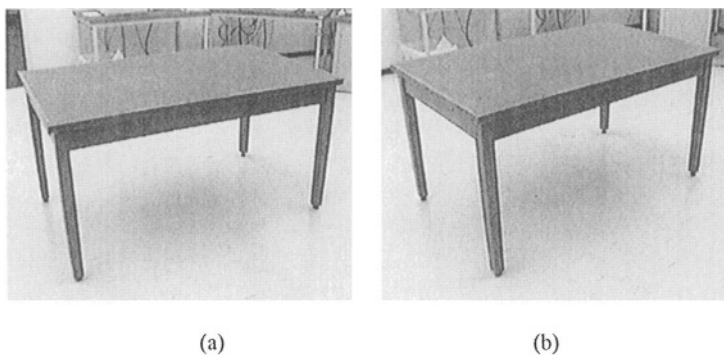


Figure 3.12. The stereo images of a table (a) left image; (b) right image.

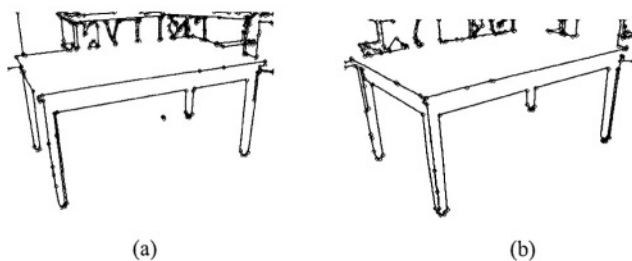


Figure 3.13. The feature extraction results from the stereo images of the table.



Figure 3.14. The partial CAD model of the table back-projected to the left image.

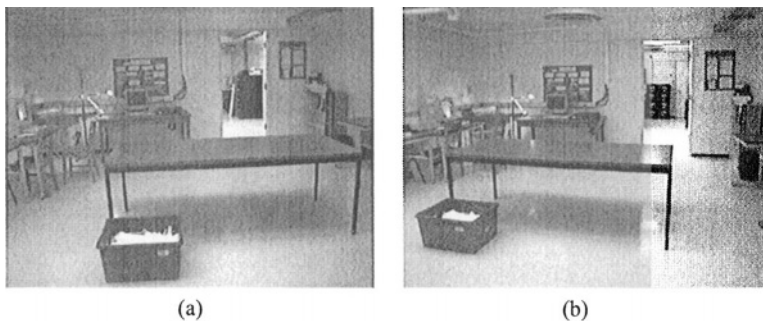


Figure 3.15. The stereo images of the laboratory scene (a) left image; (b) right image.

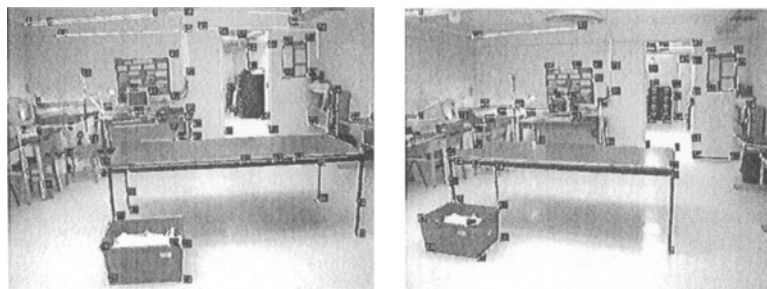


Figure 3.16. The feature extraction results on the stereo images of the laboratory scene.

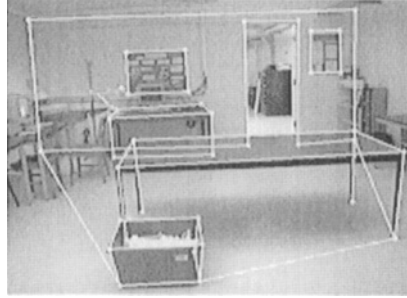


Figure 3.17. A rough CAD of the lab scene back-projected to the left image.

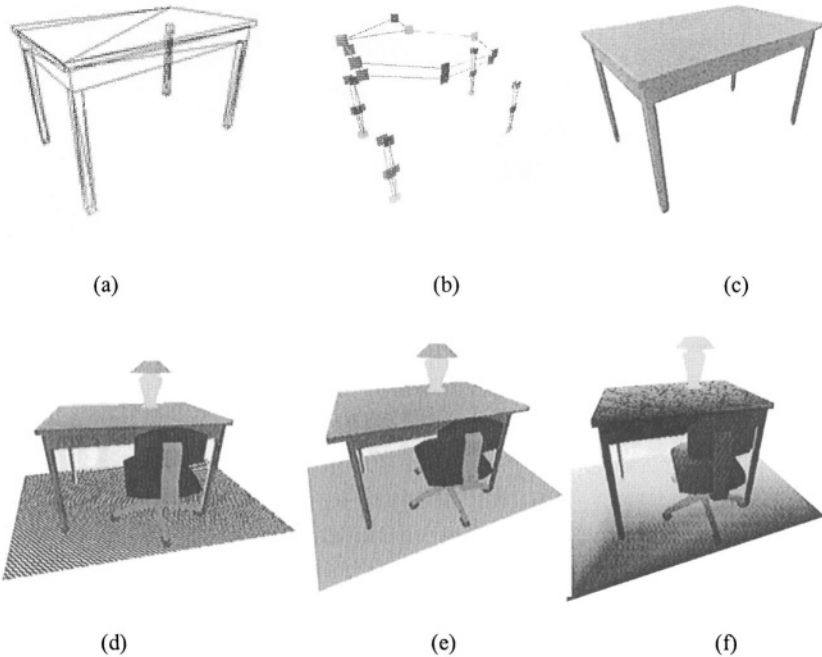


Figure 3.18. Experiments on AR on the table model using various augmentations.

3.6. Conclusions

This article presents a 3D object modeling system based on a unified *attributed hypergraph representation* (AHR). Its mathematical aspect was investigated with the theoretical support from the *category theory*. A net-like hierarchical structure called *dynamic attributed hypergraph* that supports fast structural operations was

designed. The efficacy of the methodology was illustrated by experiments on 3D model construction and augmentation applications. The theories and the applications presented are only implemented as research prototypes. The research can be extended or enforced in several areas. Optimizing the rendering part and automating the vision process are no doubt possible. Nevertheless, the proposed intelligent modeling methodology provides a very general representation scheme. It may also provide solutions in many other areas such as intelligent engineering visualization, knowledge based automatic authoring in animation, and knowledge based artificial life.

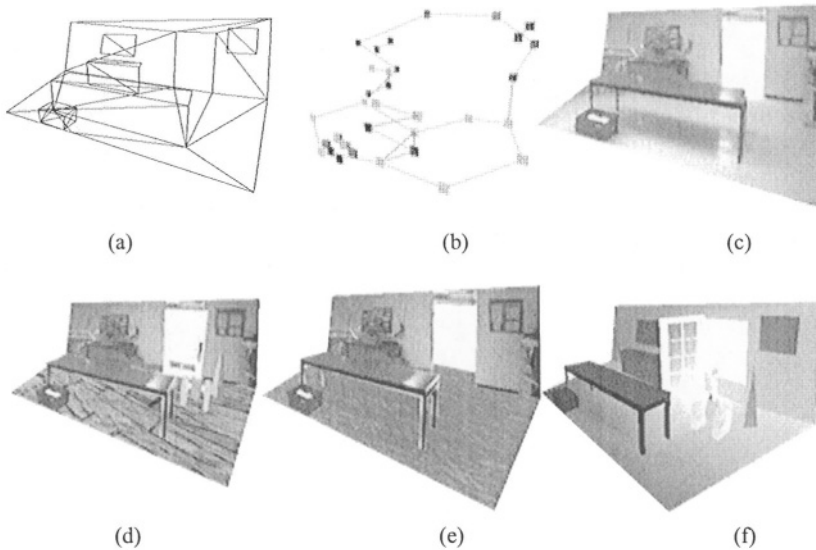


Figure 3.19. Laboratory scene experiments on AR using various augmentations.

References

- [1] A. H. Barr. "Superquadrics and angle-preserving transformations". *IEEE Computer Graphics Applications*, vol. 18, pp. 21-30, 1981.
- [2] T. S. Chua, K. H. Hay and W. N. Chin. "Optimization constraint model for dynamic animation". *Computer Graphics and Applications, Proc. of Pacific Graphics '93*, vol. 1, pp. 61-72, 1993.
- [3] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, 1993.
- [4] R. E. Fayek and A. K. C. Wong, "Triangular mesh model for natural terrain". *Proc. of SPIE: Intelligent Robots and Computer Vision*, October, 1994.

- [5] M. A. Fischler and R. C. Bolles. "Random sample consensus: A paradigm for model fitting applications to image analysis and automated cartography". *ACM Communications*, vol. 24, pp. 381-395, 1981.
- [6] L. De Floriani and E. Puppo. "Constrained delaunay triangulation for multi-resolution surface description". *Pattern Recognition*, pp. 566-569, 1988.
- [7] Q. C. Gao and A.K.C. Wong. "A curve detection approach based on perceptual organization". *Pattern Recognition*, vol. 26, no. 7, pp. 1039-1046, August, 1993.
- [8] H. C. Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections". *Nature*, vol. 293, pp. 133-135, 1981.
- [9] W. Niem and H. Broszio. "Mapping texture from multiple camera views onto 3D object models for computer animation". *Proc. of the Int. Workshop on Stereoscopic and 3D Imaging*, Santorini, Greece, September, 1995.
- [10] M. Pave. *Fundamentals of Pattern Recognition*. 2nd edition, Marcel Dekker Inc., 1993.
- [11] A. C. Shaw. "Parsing of graph-representation pictures". *Journal of ACM*, vol. 17, pp. 453-481, 1970
- [12] D. Terzopoulos and D. Metaxas. "Dynamic 3D models with local and global deformations: Deformable Superquadrics". *IEEE Trans. PAMI*, vol. 13, no. 7, pp. 703-714, 1991.
- D. Terzopoulos and K. Waters. "Analysis and synthesis of facial image sequences using physical and anatomical models". *IEEE Trans. PAMI*, vol. 15, no. 6, pp. 569-579, 1993.
- [14] A. K. C. Wong and W. Liu. "Hypergraph representation for 3-D object model synthesis and scene interpretation". *Proc. the 2nd Workshop on Sensor Fusion and Environment Modeling (ICAR)*, Oxford, UK, 1991.
- [15] A. K. C. Wong and B. A. McArthur. "Random graph representation for 3-D object models". *SPIE Milestone Series, Model-Based Vision*, vol. 72, pp. 229-238, 1991.

Chapter 4

VISIBILITY CULLING FOR INTERACTIVE DYNAMIC SCENES

George Baciú and Ki-Wan Kwok

Department of Computing, The Hong Kong Polytechnic University

Kowloon, Hong Kong

{csgeorge, cskwkwo}@comp.polyu.edu.hk

Abstract Virtual environments are rapidly growing in size and complexity. At the same time, there is a strong commercial need for rendering larger and larger scenes at interactive rates. This leads to two basic enhancements, one by increasing the performance and the size of memory in hardware in order to support large scene rendering and another in software by designing more efficient visibility algorithms. Visibility culling methods manage complexity by sending only the potentially visible primitives into the rendering pipeline. At present, occlusion culling algorithms do not handle well scenes with dynamic objects. One of the main difficulties is handling changes to the object hierarchies since the visibility information changes continuously. In this chapter, we present a fast from-region occlusion culling method that is able to compute the potential visible sets online for large dynamic outdoor scenes. The method uses an occlusion map on a dual ray-space in order to encode visibility with respect to a view cell. It utilizes new features of the advanced graphics hardware architecture to construct and maintain occlusion maps.

Keywords: From-region visibility, visibility culling, occlusion culling, interactive, real-time rendering, dynamic scene, outdoor scene, virtual environment

4.1. Introduction

Virtual environments are rapidly growing both in size and complexity. In order to render large scenes at interactive frame rates, it is important to render only the visible objects and use a suitable level of detail for objects (the size of triangles should not be less than a portion of the pixel size). The frame rate should ideally be bounded only by the screen resolution in a rendering system since the number of visible primitives is bounded. Thus, interactive

rendering can be achieved by fixing the screen resolution and using a relatively fast graphics subsystem such as the ones currently available from NVIDIA and ATI. However, in practice, visibility determination can be computationally expensive. The determination of the visible geometry from a complex 3D scene is often complex it is tightly dependent on the 3D model employed. In order to achieve interactive rendering rates, there will always be a need for faster visibility culling algorithms that determine the visible primitives for rendering.

4.1.1 Visibility Culling

Visibility culling is a technique that determines the set of visible objects in the scene. It is classified into three categories: *back-face culling*, *view-frustum culling* and *occlusion culling*. Back-face culling algorithms discard geometry that is facing away from the viewer. View-frustum culling algorithms discard geometry that is outside the view frustum. Occlusion culling algorithms try to avoid rendering objects that are occluded by some other objects in the scene. The cost of computing the exact visible set is too high [47]. Most occlusion culling algorithms in the literature computes a potentially visible set (*PVS*). The size of the PVS should be closed to that of the exact visible set.

From-point occlusion culling techniques compute visibility with respect to the current view point only. The computations are performed in every frame. So it is not feasible for interactive rendering of large scenes. *From-region* occlusion culling techniques compute visibility that is valid anywhere in a region of space. The advantage is the visibility information is valid for a couple of frames. It takes longer time to compute but the visibility can be precomputed and stored on disks. In general scenes, the typical usage is to divide the scene into a grid of viewcells. For each viewcell we compute the set of visible objects and stored them in disks. When navigating the scene, the visible set of the current viewcell is known and the visible sets of the adjacent viewcells are pre-fetched from disk. When the viewer crosses a viewcell boundary and enters another viewcell, the visible set of the new viewcell is already available. The visible sets of adjacent viewcells of this new viewcell is then pre-fetched. The drawback is it cannot handle dynamic objects as the visibility information is changing in every frame.

There are specific occlusion culling algorithms on dedicated scenes. In architectural environments, a scene is naturally partitioned into viewcells (rooms, corridors, etc). An object is visible only if the cell which the object located in is visible through a series of visible portals (e.g. doors, windows, etc). From-region visibility is achieved by maintaining sequences of visible portals for each viewcells. Therefore handling dynamic objects is possible as long as we know in which viewcells the dynamic objects are located in. Urban-scenes are 2.5 dimensions in nature. Recent methods [38, 13] are able to reduce the visibility

problem to two dimensions using dual space. Efficient algorithms can hence be constructed.

4.1.2 Occluder Fusion

In occlusion determination, the cumulative occlusion of multiple objects can be far greater than the sum of what they are able to occlude separately. Early generic from-region methods are based on a single occluder. To reduce the PVS it is necessary to detect objects that are occluded by a group of occluders. Advanced from-region methods are able to combine individual umbra into a large umbra capable of occluding larger parts of the scene (see Figure 4.1). It should be noted that occluder fusion occurs whenever individual umbrae intersect, but it can still occur if individual umbrae do not intersect.

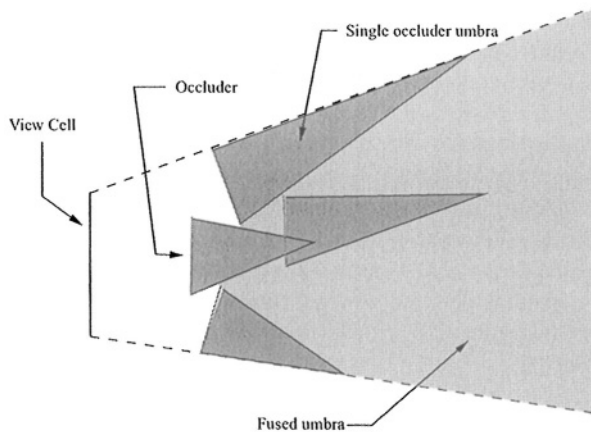


Figure 4.1. The fused umbra is larger than the union of individual umbrae.

4.1.3 Occlusion Map

When an opaque object is projected onto the screen, the projection forms an opaque region. Other objects that lie farther away from the viewer and project onto the same region will not be visible. An occlusion map is a gray-scale image that corresponds to a uniform subdivision of the screen into rectangular regions [66]. Each pixel in the occlusion map represents one of the regions, recording its opacity. Take a from-point occlusion culling example, where the screen resolution is 1024×768 and we use a 256×192 occlusion map to encode visibility. Each pixel in the occlusion map will encode the visibility of a 4×4 screen region. An occlusion map can be generated by rendering the objects

at the same resolution as the occlusion map. A higher resolution occlusion map encodes the visibilities more accurately, but induces higher computational overhead. Occlusion map is an effective way for occluder fusion. Traditional occlusion culling methods make use of occlusion maps in the object space. Recent methods are able to utilize the concept of occlusion map in a dual space of the scene. Occlusion map can be implemented using stencil buffers in conventional graphics hardware. Advance graphics hardware supports occlusion queries [49] which facilitates fast occlusion map implementations.

4.1.4 Dynamic Scenes

From-point occlusion culling algorithms can handle dynamic scenes. Visibility is computed in every frame. Therefore they are able to capture the latest states of dynamic objects. However, they are too slow for large scenes. From-region occlusion culling algorithms on architectural scenes can handle dynamic objects efficiently. If we store the *object-id* of the dynamic objects in the viewcells they are located in, dynamic objects can be culled if the viewcell they are located in is detected as invisible. However, only few occlusion culling algorithms in the literature are able to handle general dynamic scenes. One of the main difficulties is handling changes to object hierarchies. If the visibility culling algorithm uses preprocessing, this information has to be updated. Since from-region methods usually precompute a PVS, it is very hard for them to treat dynamic scenes. In particular, the whole information has to be recomputed if occluders move. Moving occludees can however be handled by bounding their motion using motion volumes [56]. However, there are no algorithms known to work well on large dynamic general scenes [17].

4.2. Overview

Leyvand et al [40] proposed a from-region visibility algorithm that computes the PVS of a static 3D outdoor scene online. It can compute the PVS of a viewcell in less than one second for a large scene. The algorithm factorizes the 4D visibility problem into horizontal and vertical components. The horizontal component is based on a ray space parameterization. The vertical component is solved by incrementally merging umbrae. The horizontal and vertical operations can be efficiently realized together by modern graphics hardware. The method encode visibility using occlusion map in the ray space. We present an extension of this method which is able to handle dynamic objects. We borrow figures from Leyvand's paper for explanations.

The scene is divided into a grid of axis-aligned square viewcells. Static objects are inserted into a kd-tree. The original position of each dynamic object is inserted into the existing nodes of the kd-tree, that is, no new kd-tree nodes

are created for the dynamic objects. The kd-tree serves two purposes: First, it allows traversing the scene in front-to-back order. Second, large parts of the scene can be culled when a large node is detected as hidden when traversing the kd-tree top-down.

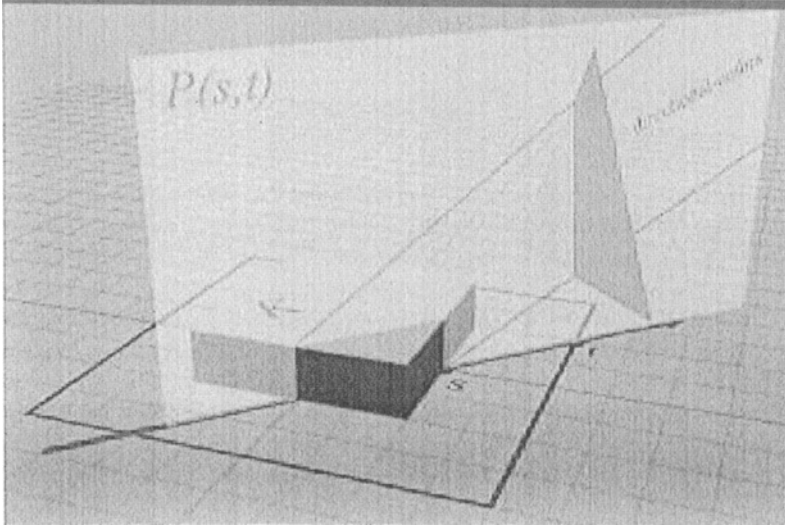


Figure 4.2. The horizontal ray direction (s, t) defines the vertical plane $P(s, t)$. The intersection between a polygon and $P(s, t)$ is a line segment. It casts a directional umbra with respect to the viewcell.

The rays originates from a given square viewcell is represented by a parameterization to be described in Section 4.3. Two parameters s, t represent a viewing direction from the plane (see Figure 4.2). We encode visibility by occlusion map on each viewing direction (s, t) . Given a viewcell, first we determine the PVS of static objects. We traverse the static objects in the kd-tree in front-to-back order. If an object is visible, we add it to the PVS and update the occlusion map accordingly. Otherwise we just discard the object.

In each frame, the state and position of the dynamic objects are updated. Each of them is deleted from the original kd-tree node and re-inserted into an existing kd-tree node corresponding to its new position. When rendering a frame, PVS of static objects together with dynamic objects located in their kd-tree nodes are sent to the rendering pipeline. Additional from-point occlusion culling techniques can be utilized here to further trim the visible set. This is useful when the expected number of objects is large or there are objects that are closed to the viewer and occupies large portion of screen spaces.

4.3. Ray Parameterization

The common duality of $\mathbb{R}^2$ is a mapping between 2D lines $y = ax + b$ in the primal space and points $(a, -b)$ in the dual space. This parameterization is unbounded and therefore simple discretization is not allowed. This section presents a bounded parameterization that does not have singularities. It parameterizes rays that cast from a given 2D square viewcell. All the rays that cast from the viewcell and intersect a triangle, form a footprint in the parameter space that can be represented by a few polygons.

4.3.1 The Parameter Space

For a given square viewcell we define two concentric squares: an inner square and an outer square. The inner square is the viewcell itself and the length of the outer square edge is chosen to be twice the length of the inner square edge. Parameters s and t ($0 \leq s, t \leq 1$) are associated with the inner and outer squares respectively (see Figure 4.3). A ray r cast from the viewcell intersects the inner square at s_r and the outer square at t_r . Thus, the parameter space is bounded and each ray maps to the parameter pair (s, t) . A ray can intersect the outer square either on a vertical edge or a horizontal edge. We choose to map the ray only to points (s, t) such that both s and t associate to parallel edges. This can still captures all the rays cast from the viewcell since each ray intersects at least one parallel pair of inner and outer edge.

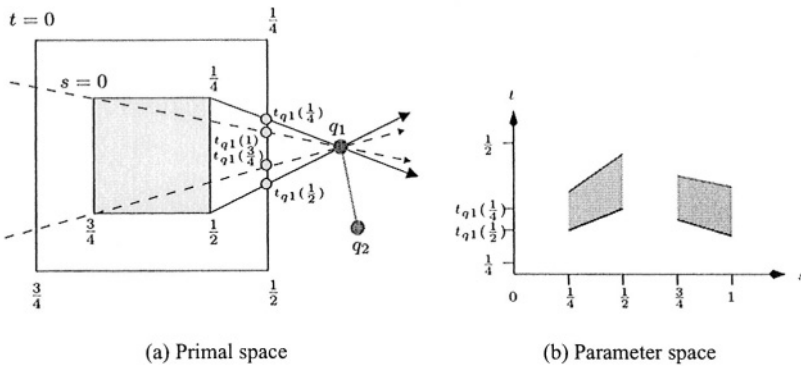


Figure 4.3. The footprint of a point is a line segment. The rays passing through a point in the primal space are mapped to line segments in the parameter space. The rays passing through a line segment in the primal space are mapped to the area bounded by the two footprints of the endpoints of that line segment.

4.3.2 The Footprint of a 2D Triangle

The *footprint* of a geometric primitive is defined as the set of all points in the parameter space that refer to rays that intersect the primitive. The following describes the shape of the footprint of a point, a segment and then a triangle.

All the rays that intersect some point q in the primal space are mapped to a set of line segments in the parameter space. In order to compute the footprint of q we need to consider the eight pairs of parallel edges of the squares. Each pair defines a line $t_q(s) = \alpha s + \beta$ in the parameter space. As the range of both s and t is bounded, the footprint of q is a line segment on the line $t_q(s)$ bounded by the domain of s and t (see Figure 4.3 (b)).

The footprint of a line segment $\overline{q_1 q_2}$ is a set of polygons in the parameter space. Each pair of parallel edges produce one polygon in the parameter space. The footprint of a 2D triangle is the union of the footprints of its edges. (see Figure 4.4).

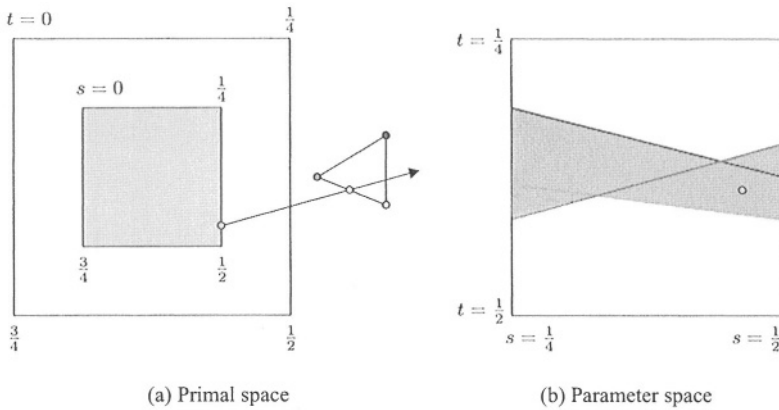


Figure 4.4. (a) The orthogonal projection of a triangle in the primal space. We use a different colour for each vertex. (b) A portion of the parameter space footprint. The line segments are in the same colour as their corresponding vertices. The footprint is divided into different regions. Each region represents rays that have the same pair of entry and exit edges.

4.4. Visibility within a Vertical Directional Plane

We traverse the cells in the kd-tree in front-to-back order and test the visibility of each cell against an occlusion map. If it is visible, we add the objects in it to the PVS and fuse their umbrae with the occlusion map. In this section, we will describe how to perform visibility queries within a vertical directional plane and the occluder fusion process.

4.4.1 Vertical Visibility Query

Let (s, t) be a point in the parameter space representing some horizontal ray, $P(s, t)$ be the vertical plane that corresponds to that direction, and the rectangle K be the intersection of the viewcell with $P(s, t)$ (see Figure 4.5). Let R be an arbitrary 3D triangle and R' is its orthogonal projection onto the ground. The line segment $B = \overline{p_1 p_2}$ is its intersection with $P(s, t)$. The line segment B casts a directional umbra with respect to K within $P(s, t)$. The directional umbra is defined by the supporting lines l_t and l_b . α_t and α_b are the supporting angles of l_t and l_b respectively. Thus, α_t and α_b can be alternatively used to define the directional umbra. We can compute the horizontal visibility component of R by parameterizing the horizontal rays that hit R' .

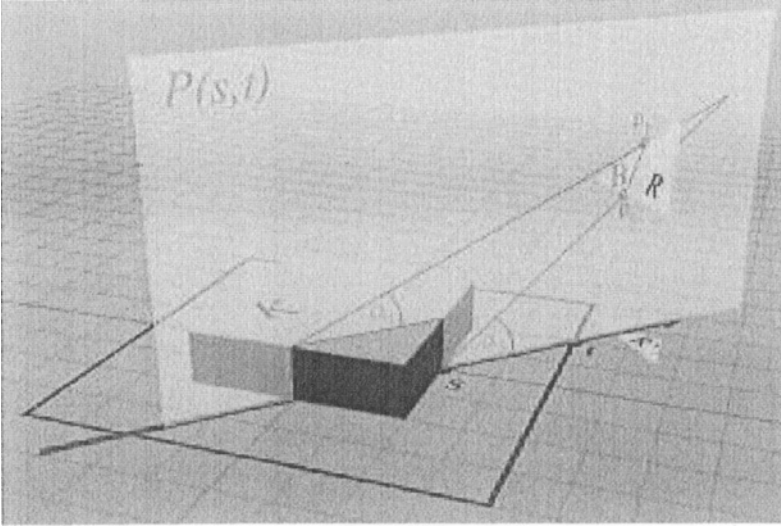


Figure 4.5. The umbra of R within the directional plane $P(s, t)$ is defined by the angles α_t and α_b .

Let Q be some other line segment within $P(s, t)$ that is behind B according to the front-to-back order with respect to the viewcell. To determine whether Q is occluded by B , we check if the umbra of B contains the umbra of Q . It happens when both endpoints of Q is contained in the umbra of B . Let β_t and β_b be the supporting angles of Q . If $\beta_t \leq \alpha_t$ and $\beta_b \geq \alpha_b$, then Q is occluded by B .

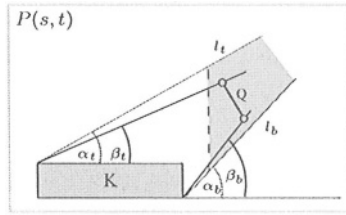


Figure 4.6. Visibility test within a vertical plane. α_t and α_b represent the accumulated umbra. If $\beta_t \leq \alpha_t$ and $\beta_b \geq \alpha_b$, then Q is occluded by the accumulated umbra.

4.4.2 Occluder Fusion

Occluder fusion in the vertical directional plane is performed by umbra aggregation from front-to-back order. If the umbra of an occluder is not completely contained in the accumulated umbra, it is fused into the accumulated umbra. The supporting lines alone do not uniquely describe the umbra in general. We need to add the separating lines to define the umbra uniquely (Figure 4.7 (a)). This also allows testing whether two umbræ intersect (Figure 4.7 (b)-(h)). In cases (b)–(d), the umbra of one segment is contained in the area between the supporting lines of the other segment. This happens when $\alpha_t < \beta_t$ and $\alpha_b > \beta_b$ (or vice versa). In this case, we discard the contained segment. In case (g) the umbræ intersect but do not contain each other, the segments are merged into a *virtual occluder* (Figure 4.7 (h)). The angles of this virtual occluder are calculated as: $\gamma_t = \max(\alpha_t, \beta_t)$, $\gamma_b = \min(\alpha_b, \beta_b)$, $\tilde{\gamma}_t = \min(\tilde{\alpha}_t, \beta_t)$, $\tilde{\gamma}_b = \max(\tilde{\alpha}_b, \beta_b)$. For the occludees behind, the occluding ability of this virtual occluder is the same as the two original segments. If their umbræ do not intersect (cases (e),(f)), their umbræ cannot be fused and their aggregated occlusion is represented by the union of their umbræ. Therefore for each vertical plane, a number of umbræ are needed to be maintained for a full occlusion description. If we just maintain one umbra, the occlusion description is more conservative and hence the PVS generated becomes larger. However, it makes hardware implementation possible. Therefore we discard the last umbra component. Leyvand claimed that for scenes with low vertical complexity, maintaining a single umbra in the occlusion map is efficient enough as it captures most of the occlusion and produces a tight PVS. He believes that typically, a small number of umbræ is enough to converge into a large augmented umbra. Processing adjacent or nearby triangles is likely to rapidly merge small umbræ into one larger umbra. Thus, an approximated ordering of the occluders is more efficient. However, in true 3D models with no preferable orientation, maintaining only one umbra is overly conservative and ineffective.

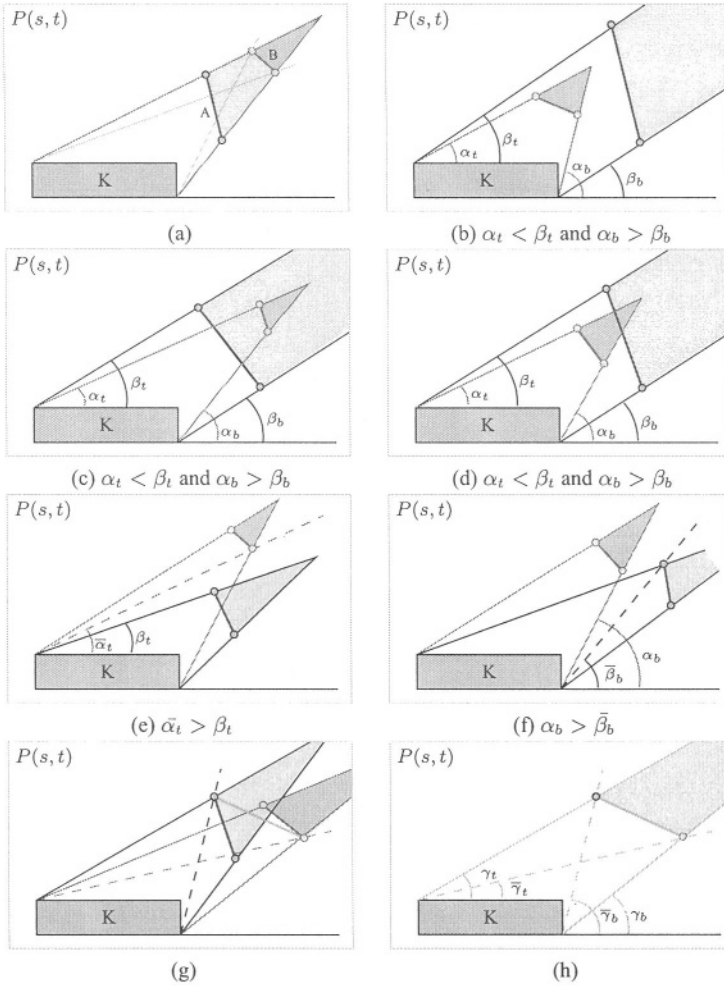


Figure 4.7. (a) The line segments A and B have the same supporting lines, but their umbrae are different. The separating lines are needed together with the supporting lines to define the umbra uniquely. (b)-(h) Different cases of umbra fusion in the vertical plane. Supporting lines are denoted by solid lines and separating lines are denoted by dashed lines. $\alpha_t, \alpha_b, \beta_t$ and β_b are supporting angles and $\tilde{\alpha}_t, \tilde{\alpha}_b, \tilde{\beta}_t, \tilde{\beta}_b$ are separating angles.

4.5. Visibility Culling on Static Objects

The parameter space is conservatively discretized as described in [63, 23, 38]. Since we are maintaining only one umbra, there are four values (angles) needed for each direction (s, t) in the occlusion map. We denote the four values as

$v_i = \{v_0 \dots v_3\}$. Each (s, t, v_i) defines a surface that can be computed using shading operations available on advanced graphics hardware. All the per- (s, t) vertical visibility queries and umbra merging operations are performed by per-pixel operations available in advanced graphics hardware. This can be achieved because directional operations are independent of each other. Thus the visibility queries and umbra merging operations can be performed in parallel across the parameter space.

4.5.1 Hardware Implementation

Readers are referred to [40] on the hardware implementation using conventional graphics hardware. We describe here the hardware implementation of testing visibility and augmenting occlusion map using the nVidia GeForce FX graphics card and nVidia's Cg shader language. It uses the available 32bit floating-point PBuffers (denoted *occlusionPB*) to store the global occlusion map and another 32bit floating-point PBuffer (denoted *tempPB*) for temporary storage. The separating and supporting angles are represented by their tangents.

To augment the occlusion map with the umbra of an occluder triangle R , we use *occlusionPB* as input texture and *tempPB* as output. We render the 2D footprint of R , which triggers fragment shader code on each (s, t) pixel that calculates the supporting and separating angles. The angles representing the current umbra is read from *occlusionPB* and compared to the calculated angles as described in Section 4.4.2. If umbra fusion occurs, the fused umbra is output to *tempPB*. After rendering R , the occlusion query extension tells whether umbra fusion had occurred. In that case, *tempPB* is set as input texture and *occlusionPB* as output. The fragment shader copies the updated umbra to *occlusionPB*.

To test the visibility of R against the occlusion map, we set *occlusionPB* as input texture and *tempPB* as output. The fragment shader only calculates the supporting angles of R and compares them with the current occlusion umbra angles read from *occlusionPB* as described in Figure 4.6, and outputs some arbitrary value to *tempPB* if R is visible. The occlusion query extension on *tempPB* tells whether R is visible.

4.6. Dynamic Scene Occlusion Culling

This section presents the complete dynamic occlusion culling algorithms. The scene is divided into a grid of axis-aligned square viewcells. Static objects are inserted into a kd-tree. Dynamic objects are inserted into existing kd-tree nodes corresponding to their initial positions. The PVS of the initial viewcell and its adjacent viewcells are computed. The infinite loop in *Main-Loop* demonstrates the dynamic occlusion culling process in every frame. The

function *UpdateDynamicObjects* updates the states of the dynamic objects. It removes each dynamic object from the old position in the kd-tree and re-insert it into a kd-tree node corresponding to its new position. When the view point enters an adjacent viewcell Q , the PVS of Q is already available. Some of the previous adjacent viewcells remain as adjacent viewcells of Q . The PVS of those do not remain as adjacent viewcells are discarded. The PVS of new adjacent viewcells are computed by the algorithm *FindPVS*. *FindPVS* should be executed in parallel with the main loop. Otherwise the application cannot be interactive. After the updating, all static objects in the PVS of the current viewcell and dynamic objects inside all potentially visible kd-tree nodes are rendered by the function *RenderScene*.

Algorithm *MainLoop()*

1. $V \leftarrow \text{InitialViewcell}$
2. *FindPVS*(V) **for** each adjacent viewcell A of V
3. *FindPVS*(A)
4. **while true**
5. *UpdateViewPoint*()
6. *UpdateDynamicObjects*()
7. **if** view point have entered an adjacent viewcell Q **then**
8. $V \leftarrow Q$
9. *DiscardOutdatedAdjPVS*()
10. **for** each adjacent viewcell A of V that is not in memory
11. *FindPVS*(A)
12. *RenderScene*(V)

Algorithm *RenderScene*(V)

Input. viewcell V

Output. image on screen

1. $N \leftarrow \phi$
2. **for** each static object $S \in V.\text{PVS}$
3. $N \leftarrow N \cup S.\text{kd-node}$
4. *RenderObject*(S)
5. **for** each kd-node $n \in N$
6. **for** each dynamic object $D \in n$
7. *RenderObject*(D)

The *FindPVS* algorithm traverse the kd-tree top-down in a front-to-back order. A kd-tree node n is tested for visibility against the occlusion map. It is discarded if it is not visible. If n is visible, individual objects within n are tested for visibility. The visible objects are added to the PVS and their associated triangles are augmented to the occlusion map.

Algorithm *FindPVS(V)*

Input. viewcell V

Output. PVS of V

1. $V.PVS \leftarrow \phi$
2. *FindPVS*($V, kd\text{-}tree.root$)

Algorithm *FindPVS(V, n)*

Input. viewcell V , kd-tree node n

Output. temporary PVS of V after traversing n

1. **if** *IsVisible*(V, n) **then**
2. **if** $n.IsLeaf()$ **then**
3. $V.PVS \leftarrow V.PVS \cup n.getTriangles()$
4. *AugmentOcclusionMap*($n.getTriangles()$)
5. **else**
6. **for each** $kdChild \in n.children$ **in front-to-back order**
7. *FindPVS*($V, kdChild$)
8. **else**
9. **return**

4.7. Conclusion

We have presented an online from-region occlusion culling algorithm for large dynamic outdoor scenes. It is an extension to Leyvand's algorithm [40] which handles static outdoor scenes of low vertical complexity efficiently. It is an important contribution to the real-time rendering field for the reasons described below. From-point methods can handle dynamic scenes, but they are not efficient for large scenes. From-region methods are suitable for large scenes. However, current from-region methods are not able to handle dynamic objects efficiently in outdoor scenes as they need to update object hierarchies and hence visibility information in every frame. Offline from-region methods pre-compute the PVS of all viewcells and store in disk. They have the advantage of fast visibility determination (just loading the PVS from disk). But they need large storage spaces and cannot handle dynamic objects since visibilities keep changing. Online from-region methods save the large storage spaces. Recent online from-region methods [38, 40] make use of a dual ray-space and take advantages of graphics hardware to solve the visibility on the ray-space. They are fast enough that the visibility culling time can be amortized over frame time. Therefore, online methods are practically more useful.

References

- [1] T. Aila. *SurRender Umbra: A Visibility Determination Framework for Dynamic Environments*. Master's thesis, Helsinki University of Technology, 2000.
- [2] J. M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, Dept. of CS, U. of North Carolina, July, 1990.
- [3] J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr. "Towards image realism with interactive update rates in complex virtual building environments". *Computer Graphics 1990 Symposium on Interactive 3D Graphics*, vol. 24, no. 2, pp. 41-50, March, 1990.
- [4] T. A. Moller and E. Haines. *Real-time Rendering*. 2nd Ed. AK Peters, 2002.
- [5] C. Andujar, C. S. Vazquez, I. Navazo, and P. Brunet. "Integrating occlusion culling and levels of details through hardly-visible sets". *Computer Graphics Forum*, vol. 19, no. 3, 2000.
- [6] D. Bartz, J. Klosowski, and D. Staneker. "k-dops as tighter bounding volumes for better occlusion performance". In *ACM SIGGRAPH Visual Proceedings 2001*.
- [7] F. Bernardini, J. T. Klosowski, and J. El-Sana. "Directional discretized occluders for accelerated occlusion culling". *Computer Graphics Forum*, vol. 19, no. 3, 2000.
- [8] J. Bittner, V. Havran, and P. Slavik. "Hierarchical visibility culling with occlusion trees". In *Proceedings of Computer Graphics International*, pp. 207-219, 1998.
- [9] J. Bittner, P. Wonka and M. Wimmer. "Visibility preprocessing in urban scenes using line space subdivision". In *Proceedings of Pacific Graphics (PG'01)*, pp. 276-284, Tokyo, Japan, October 2001.
- [10] J. Bittner. *Hierarchical Techniques for Visibility Computations*. PhD Dissertation. Department of Computer Science and Engineering, Czech Technical University in Prague, October 2002.
- [11] J. Bittner. "Efficient construction of visibility maps using approximate occlusion sweep". In *Proceedings of Spring Conference on Computer Graphics (SCCG02)*, pp. 163-171, Budmerice, Slovakia.
- [12] J. Bittner and P. Wonka. "Visibility in computer graphics". To appear in *Journal of Environment and Planning – B*, Pion Ltd., 2003. Also available as Technical Report TR-186-2-03-03 at the Insitute of Computer Graphics and Algorithms, Vienna University of Technology.
- [13] J. Bittner, J. Prikryl and P. Slavik. "Exact regional visibility using line-space partitioning". To appear in *em Computers & Graphics*, vol. 27, no. 4, 2003. Also available as Technical Report TR-186-2-01-06 at the Insitute of Computer Graphics and Algorithms, Vienna University of Technology.
- [14] Y. Chrysanthou and M. Slater. "Shadow volume BSP trees for computation of shadows in dynamic scenes". In *1995 Symposium on Interactive 3D Graphics*, P. Hanrahan & J. Winget, eds., pp. 45-50, *ACM SIGGRAPH*, ISBN 0-89791-736-7.
- [15] Y. Chrysanthou, D. Cohen-Or and D. Lischinski. "Fast approximate quantitative visibility for complex scenes". In *Proceedings of Computer Graphics International 98 (CGI98)*, pp. 23-31, IEEE, NY, Hannover, Germany.
- [16] D. Cohen-Or, G. Fibich, D. Halperin and E. Zadicario. "Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes". *Computer Graphics Forum*, vol. 17, no. 3, pp. 243-254, 1998.

- [17] D. Cohen-Or, Y. Chrysanthou, C. T. Silva and F. Durand. "A survey of visibility for walkthrough applications". To be appeared in *Transaction of Visualization and Computer Graphics*.
- [18] S. Coorg, and S. Teller. "Temporally coherent conservative visibility". In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pp. 78-87, 1996.
- [19] S. Coorg and S. Teller. "Real-time occlusion culling for models with large occluders". In *ACM, editor, Proc. Symposium on Interactive 3D Graphics (SI3D97)*, pp. 83-90, 1997.
- [20] L. Downs, T. Möller and C.H. Séquin. "Occlusion horizons for driving through urban scenes". In *Symposium on Interactive 3D Graphics*, pp. 121-124, *ACM SIGGRAPH*, 2001.
- [21] F. Durand. *3D Visibility: Analytical Study and Applications*. PhD thesis, Universite Joseph Fourier, Grenoble, France, 1999.
- [22] F. Durand, G. Drettakis and C. Puech. "Fast and accurate hierarchical radiosity using global visibility". *ACM Transactions on Graphics*, April, 1999.
- [23] F. Durand, G. Drettakis, J. Thollot and C. Puech. "Conservative visibility preprocessing using extended projections". *Proceedings of ACM SIGGRAPH 2000*, pp. 239-248, July, 2000.
- [24] D. H. Edberly. *3D Game Engine Design. A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann Publishers, 2001.
- [25] T. A. Funkhouser, C. H. S'équin, and S. J. Teller. "Management of large amounts of data in interactive building walkthroughs". *1992 Symposium on Interactive 3D Graphics*, vol. 25, no. 2, pp. 11-20, March, 1992.
- [26] T. Funkhouser and C. S'équin. "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments". In *Computer Graphics (Proc. Siggraph)*, 1993.
- [27] T.A. Funkhouser. "Database management for interactive display of large architectural models". *Graphics Interface*, pp. 1-8, May, 1996.
- [28] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen. "The lumigraph". In *Proceedings of ACM SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series*, pp. 43-54, 1996.
- [29] C. Gotsman, O. Sudarsky and J. Fayman. "Optimized occlusion culling using five-dimensional subdivision". *Computer & Graphics*, vol. 23, no. 5, pp. 645-654, 1999.
- [30] J. Grasset, O. Terraz, J.M. Hasenfratz and D. Plemenos. "Accurate scene display by using visibility maps". In *Spring Conference on Computer Graphics and its Applications*, 1999.
- [31] N. Greene, M. Kass and G. Miller. "Hierarchical Z-buffer visibility". In *Computer Graphics (Proceedings of SIGGRAPH 93)*, pp. 231-238.
- [32] H. Hey, R.F. Tobler and W. Purgathofer. "Real-Time occlusion culling with a lazy occlusion grid". In *Proceedings of EUROGRAPHICS Workshop on Rendering 2001*, pp. 217-222.
- [33] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff and H. Zhang. "Accelerated occlusion culling using shadow frustra". In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 1-10, 1997.
- [34] J. T. Klosowski and Cláudio T. Silva. "Rendering on a budget: A framework for time-critical rendering". *IEEE Visualization*, vol. 99, pp. 115-122, October, 1999.

- [35] J. T. Klosowski and Cláudio T. Silva. "The prioritized layered projection algorithm for visible set estimation". *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 108-123, April-June, 2000, ISSN 1077-2626.
- [36] J. Klosowski and C. Silva. "Efficient conservative visibility culling using the prioritized-layered projection algorithm". *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 4, 2001.
- [37] V. Koltun, Y. Chrysanthou and D. Cohen-Or. "Virtual occluders: An efficient intermediate PVS representation". *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pp. 59-70, June, 2000.
- [38] V. Koltun, D. Cohen-Or and Y. Chrysanthou. "Hardware accelerated from-region visibility using a dual ray space". *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pp. 59-70, June.
- [39] M. Levoy, and P. M. Hanrahan. "Light field rendering". In *Proceedings of ACM SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series*, pp. 31-42, 1996.
- [40] T. Leyvand, O. Sorkine and D. Cohen-Or. "Ray space factorization for from-region visibility". In *Proceedings of ACM SIGGRAPH 2003*.
- [41] B. Lloyd and P. Egbert. "Horizon occlusion culling for real-time rendering of hierarchical terrains". In *Proceedings of the conference on Visualization 02*, pp. 403-410, IEEE Press.
- [42] D. Luebke and C. Georges. "Portals and mirrors: Simple, fast evaluation of potentially visible sets". In *A. Press, editor, Proc. ACM Symposium on Interactive 3D Graphics, special issue of Computer Graphics*, pp. 105-106, Monterey, CA, April, 1995.
- [43] D. Meneveaux, E. Maisel, C. F., and K. Bouatouch. *Partitioning Complex Architectural Environments for Lighting Simulation*. Technical Report 2981, INRIA/IRISA, 1996.
- [44] D. Meneveaux, E. Maisel, and K. Bouatouch. *A New Partitioning Method for Architectural Environments*. Technical Report 1096, IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France, April, 1997.
- [45] T. Moller and E. Haines. *Real-Time Rendering*. A. K. Peters Limited, 1999.
- [46] B. Nadler, G. Fibich, S. Lev-Yehudi and D. Cohen-Or. "A qualitative and quantitative visibility analysis in urban scenes". *Computer & Graphics*, vol. 23, no. 5, pp.655-666, 1999.
- [47] S. Nirenstein, E. Blake and J. Gain. "Exact from-region visibility culling". In *Rendering Techniques 2002, Proceedings of the 13th EUROGRAPHICS Workshop on Rendering*, pp. 199-210.
- [48] NVIDIA. <http://developer.nvidia.com>.
- [49] NVIDIA. http://www.nvidia.com/dev_content/nvopengl/specs/GL_NV_occlusion_query.txt. web page.
- [50] Harry Plantinga. "Conservative visibility preprocessing for efficient walkthroughs of 3D scenes". In *Proceedings of Graphics Interface 93*, pp. 166-173, Canadian Information Processing Society, Toronto, Ontario, Canada, May, 1993.
- [51] S. Rivière. "Dynamic visibility in polygonal scenes with the visibility complex". In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 421-423, 1997.
- [52] C. Saona-vazquez, I. Navazo and P. Brunet. "The visibility octree: A data structure for 3D navigation". *Computer & Graphics*, vol. 23, no. 5, 1999.

- [53] G. Schaufler, J. Dorsey, X. Decoret and F. X. Sillion. "Conservative volumetric visibility with occluder fusion". *Proceedings of ACM SIGGRAPH 2000*, pp. 229-238, July, 2000.
- [54] N. Scott, D. Olsen and E. Gannett. "An overview of the VISUALIZE fx graphics accelerator hardware". *The Hewlett-Packard Journal*, pp. 28-34, May, 1998.
- [55] M. Slater, A. Steed and Y. Chrysanthou. *Computer Graphics and Virtual Environments*, Addison Wesley, 2002.
- [56] O. Sudarsky and C. Gotsman. "Dynamic scene occlusion culling". *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 13-29, January-March, 1999.
- [57] D. Staneker, D. Bartz AND M. Meissner. "Using occupancy maps for better occlusion query efficiency (Poster)". In *Eurographics Workshop on Rendering*, 2002.
- [58] S. J. Teller and C. S'equin. "Visibility preprocessing for interactive walkthroughs". In *Proc. Computer Graphics (SIGGRAPH 91)*, vol. 25, pp. 61-69, ACM Press, 1991.
- [59] S. Teller. *Visibility Computation in Densely Occluded Polyhedral Environments*. PhD thesis, UC Berkeley, CS department, 1992.
- [60] S. J. Teller. "Computing the antipenumbra of an area light source". *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, no. 2, 1992.
- [61] S. Teller and P. Hanrahan. "Visibility computations for global illumination algorithms". In *Proc. Computer Graphics (SIGGRAPH 93)*, vol. 27, pp. 239-246, ACM Press, July, 1993.
- [62] M. van de Panne and J. Stewart. "Efficient compression techniques for precomputed visibility". In *Proc. Eurographics Rendering Workshop*, pp. 305-316, June, 1999.
- [63] P. Wonka, and D. Schmalstieg. "Occluder shadows for fast walkthroughs of urban environments". In *Computer Graphics Forum*, vol. 18, pp. 51-60, 1999.
- [64] P. Wonka, M. Wimmer and D. Schmalstieg. "Visibility preprocessing with occluder fusion for urban walkthroughs". *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pp. 71-82, 2000.
- [65] P. Wonka, M. Wimmer and F. X. Sillion. "Instant visibility". In *Computer Graphics Forum (Proc. of Eurographics 01)*, vol. 20, no. 3, 2001.
- [66] H. Zhang, D. Manocha, T. Hudson and K. E. Hoff III. "Visibility culling using hierarchical occlusion maps". In *Proceedings of ACM SIGGRAPH 97*, pp. 7788.

This page intentionally left blank

Chapter 5

IMAGE-BASED COLLISION DETECTION *

George Baciú and Wingo Sai-Keung Wong

Department of Computing, The Hong Kong Polytechnic University

Kowloon, Hong Kong

{csgeorge,cswingo}@comp.polyu.edu.hk

Abstract Dynamic objects in virtual environments are the source of some of the largest number of interaction events. These are mostly generated from object collisions. The collision detection methods that have been developed so far are based on geometrical *object-space* interference tests. In this chapter, we first discuss the development of conventional collision detection methods that has lead to certain limitations due to their dependence on isolating geometrical features in close proximity. Then, we introduce a more comprehensive approach based on *image-space* interference tests which improves on the object-space collision detection by distributing the computational load throughout the graphics pipeline. In conjunction with efficient bounding-box strategies in the object-space, this approach can handle complex object interactions of both rigid and deformable objects of arbitrary surface complexity at interactive rates.

Keywords: Collision detection, object-space culling, surface intersections, animation

5.1. Introduction

Despite the large number of solutions proposed for handling collision detection for three-dimensional objects, recently, there has been a resurgence of work in this area. This is observed not only in interactive computer graphics but also in mechanics, robotics, computer vision, and computational geometry. It seems that much larger computer models, with a significant number of features, are appearing in a variety of applications from virtual navigation, robotics, path planning, CAD, computer games and special effects. However, the compu-

* Portions reprinted, with permission, from "Image-based techniques in a hybrid collision detector," IEEE Transactions on Visualization and Computer Graphics, Vol. 9, No. 2, Apr. 2003, pp. 254-271. Copyright ©2003 IEEE.

tational complexity associated with interference detections, proximity tests or object interactions pose major challenges to both researchers and application designers.

5.1.1 Background

In the 1980's, the simulation and animation of 3D object interactions did not require more than a few hundred polygons. From falling chains to robotic assembly simulations, vehicle dynamics and crash tests, the complexity of the models has exploded to hundreds of thousands of geometrical components and features. Now, the order of the day is clothing, soft and deformable bodies and, of course, hair, fur, grass, leaves and everything that makes nature so much more interesting. This has placed a much higher demand on automatic collision detection for complex object interactions.

For example, in Figure 5.1, a dress falling onto the model of a female body consists of more than 30,000 triangles. Two kinds of collision events can occur: inter-collision and self-collision. This animation uses a hybrid approach based on both object space triangle culling and image space interference detection. The speedup is more than 2.5 the conventional techniques.

Recently, there has been a resurgence of research work on collision detection. In many cases, we observe a compromise between complexity and the frame rate performance for interactive purposes. This work lead to new and significant ideas in object representation, spatial partitioning and culling methods. First, there are a number of new bounding box techniques [12, 13]. Second, hierarchies have improved the performance of locating interfering features [20]. Third, localization of features and distance tracking between objects have sped up the collision detection methods [12, 30]. Fourth, adaptive time steps, dependent on speed and acceleration for robust detection, have been proposed and implemented [16].

Object-based collision detection (OBCD) methods are generally dependent on the geometrical structure of objects. This structure can exhibit a high degree of complexity even for simple combinations of polyhedra. This leads to a highly complex computational task for accurately and efficiently identifying the closest features between pairs of objects in close proximity. Furthermore, most methods are limited to planar bounding surfaces. They are often costly to run either due to pre-processing or due to the run-time search for the closest features.

Tracking the motion of arbitrary objects in 3D space can ultimately overload the main processor with object-space identification and tracking of the closest features. Despite the progress in graphics acceleration hardware, it seems that the collision detection problem in object space will remain bounded by

$O((N + K)\log^2 N)$ [30], where N is the number of polyhedra and K is the number of intersecting polyhedra pairs.

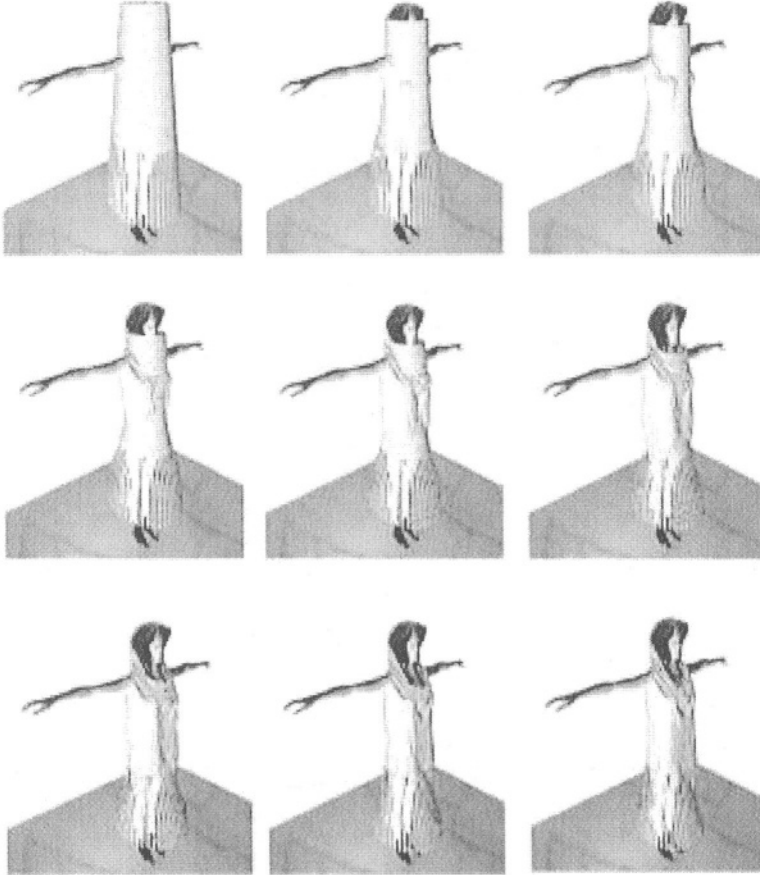


Figure 5.1. A fringed dress drapes under the effect of gravity on a female body model. This model consists of 30,609 triangles and each side has 41 fringes.

In traditional algorithms, all the interference tests are performed on the main processor while the rendering pipeline waits for a new scene to be processed. In the attempt to balance the computational load between the main processor and the rendering pipeline, a new class of collision detection algorithms are currently being developed. This is the class of *image-based collision detection (IBCD)* methods.

5.1.2 So, What is New in 3D Collision Detection?

In contrast with the object-based collision detection methods, the image-based collision detection solutions are much simpler to implement, exhibit a high degree of robustness with respect to the object geometry, and have a high potential to exploit the hardware-assisted rendering features of the current graphics boards. This trend started with the work of Shinya and Forgue [28], Rossignac [27], Myszkowski et. al [25], and Baciú et al [2]. Until recently, the image-based collision detection methods have gained little acceptance due to lack of hardware support. With the advent of better localization methods [1], the image-based collision detection methods have become competitive with the pure object-space approaches. In solid modeling, performing transparency operations on CSG models in hardware was reported by Kelley et al [19]. More recently Hoff, Keyser, Lin, Manocha and Culver have reported real-time processing of the Voronoi diagram using z-buffer hardware [15].

5.1.3 Performance Characteristics

In this chapter, we concentrate on the characteristics of image-based collision detection methods and show new performance improvements due to better 3D feature pair isolation and localization of potential collision areas. Currently, commercial hardware rendering pipelines do not support bitwise tests for checking the change of state in the stencil buffer. In the current hardware, this crucial step could only be emulated by reading blocks from the stencil buffer into the main memory and performing a content-based check for possible changes of the stencil buffer values.

In the process of testing different collision detection methods we developed a scalable and repeatable test experiment, called *the comb experiment*. This experiment emphasizes both the robustness and the performance characteristics of various algorithms. Due to its scalability, simple implementation, setup and control, this experiment can potentially serve as a standard testbed for all the available and the future collision detection solutions [1].

The hybrid object-space/image-space collision detection method adds the following advantages to the pure object-space methods: (1) it attains a better computational load balance between the general purpose processor and the rasterization engines available on most current graphics boards, and (2) it compensates for the geometrical complexity in the close proximity of colliding features by resorting to the manipulation of the view point and image-space sampling.

5.1.4 Chapter Summary

The chapter is organized as follows. In section 5.2, we outline the background of the problem stressing the difference between image-space and object-space

collision detection approaches. In section 5.3 we give a perspective of some of the most relevant and recent work on collision detection methods and classify them according to the image– (section 5.3.2) and object–based division (section 5.3.1). Starting with section 5.4, we develop the details of our image–based algorithm in incremental stages, from a brute force approach to a highly optimized version. We conclude the chapter with a hybrid algorithm that takes advantage of both the object–space culling and the image–space interference tests in order to fully utilize the graphics pipeline. This leads to a new computer architecture that supports interactive collision detection for arbitrary surfaces.

5.2. Simulation Space

We describe the dynamic simulation space in terms of a mathematical framework that accounts for all the possible objects and all the features associated with these objects in a 3D space. Thus, the collision detection algorithms can be analyzed based on the following entities. Let t be the simulation time sampled as t_s at intervals Δt_s . Then, $S = \{x, y, z, t\}$ is the simulation space. We let $O = \{o_1, o_2, \dots, o_n\}$ be the set of disjoint geometric objects in 3D space and $F = \{F^1, F^2, \dots, F^c\}$ be the set of geometric feature classes, e.g. points, lines, planes, curves, etc. Then, $F^k = \{f_1^k, f_2^k, \dots, f_{m_k}^k\}$ is the set of all the geometric features of class k , where m_k depends on the feature class k . Let $G_i^k = \{g_{i1}^k, g_{i2}^k, \dots, g_{il_i}^k\}$ be the subset of the geometric features of class k found in object o_i , where l_i depends on the feature class k . Finally, if we let $V = \{v_1, v_2, \dots, v_n\}$ be the set of bounding volumes such that v_i encloses object o_i , i.e. $o_i \subset v_i$, then, the interactions between all the geometric objects in S at time samples a and b can be described by the algorithm in Figure 5.2. The generic collision test is described in Figure 5.2. Instances of the geometry definitions of the object set O and bounding volume set V , respectively, are defined by O_s and V_s . These objects are sampled at time t_s in the simulation space S . Similarly, O_c and V_c are instances of the geometry definitions of the object set O and bounding volume set V , respectively. These are sampled at time t_c in the simulation space S . The exact time of collision between two objects is represented by t_c . This is done on the sampled interval $(t_s - \Delta t_s, t_s)$ as the pairs of objects come into the proximity of their bounding volumes. The sampling of the simulation space is the major cause for computational bottlenecks.

As Hubbard [16] points out, in addition to the sampling problems in both time and space, the existing algorithms suffer from the computational complexity associated with the $O(N^2)$ tests performed in lines 5-18 of the COL function, Figure 5.2.

How are these problems currently solved? The time sampling problem has been addressed in detail by Hubbard [16, 17] who developed an adaptive time

```

1 proc SIM( $O, V, a, b$ )  $\equiv$ 
2    $\Delta t_s \leftarrow \text{initializeTimeStep}(a, b)$ ;
3   for  $t_s \leftarrow a$  to  $b$  step  $\Delta t_s$  do
4      $O_s \leftarrow \text{computeGeometry}(O, t_s)$ 
5      $V_s \leftarrow \text{computeVolumes}(O, V, t_s)$ 
6     if COL( $\Delta t_s, t_s, O_s, V_s$ )
7        $O_s \leftarrow \text{responseObjects}(t_s, O_s)$ 
8        $V_s \leftarrow \text{responseVolumes}(t_s, O_s, V_s)$ 
9     fi
10     $\Delta t_s \leftarrow \text{adaptTimeStep}(t_s, \Delta t_s)$ ;
11    render( $O_s$ )
12  od
13 .

```

```

1 proc COL( $dt, t, O, V$ )  $\equiv$ 
2    $\Delta t_c \leftarrow \text{decreaseTimeStep}(dt)$ ;
3   for  $t_c \leftarrow (t - dt)$  to  $t$  step  $\Delta t_c$  do
4      $V_c \leftarrow \text{computeVolumes}(O, V, t_c)$ 
5     for  $i \leftarrow 1$  to  $n$  do
6        $v_i \leftarrow \text{getVolume}(V_c, i, t_c)$ 
7       for  $j \leftarrow 1$  to  $n$  do
8          $v_j \leftarrow \text{getVolume}(V_c, j, t_c)$ 
9         if  $(i \neq j)$  and  $(v_i \cap v_j \neq \phi)$ 
10           $o_i \leftarrow \text{getObject}(O, i, t_c)$ 
11           $o_j \leftarrow \text{getObject}(O, j, t_c)$ 
12          if  $o_i \cap o_j \neq \phi$ 
13             $t_s \leftarrow t_c$ ;
14            return{TRUE,  $t_s, i, j$ };
15          fi
16        fi
17      od
18    od
19  od
20  return{FALSE,  $t_s, 0, 0$ };
21 .

```

Figure 5.2. Object interaction simulation.

step algorithm based on an envelope of computed velocity and acceleration bounds from the current object velocity and acceleration. This significant contribution allows us to determine adaptively the time steps Δt_s and Δt_c for robust collision detection tests. However, intersections are determined in the object space by using bounding spheres as preliminary proximity tests. Such adaptive time step algorithms have the advantage of being more flexible than the 4D time-space algorithms that compute the intersections between volume sweeps

in both time and space [5] since the complete motion trajectories are not always known in advance. Similar adaptive bounds have been proposed by Mirtich and Canny [23] and Foisy et al. [9].

5.3. Object Space vs. Image Space Collision Detection

Solutions to the collision detection problem can be classified according to the type of sampling being performed: 1. the space domain, 2. the time domain, or both. Temporal sampling can be further taken as either *global* or *local*. Within each temporal classification, the spatial sampling for collision detection can be classified into (1) *object-space interference test* and (2) *image-space interference test*.

5.3.1 Object-space Collision Detection

Object-based interference tests include a variety of techniques from topological analysis to computational geometry. An interference structure is usually built either locally or globally in a precomputation phase. Data structures, such as BSP trees [31], octrees [24, 32], voxel grid [10], or hierarchical oriented bounding boxes [13] are built for the spatial localization of regions of potential interference.

Studies in the *object-space interference* include local algorithms [5, 6, 11, 24, 14, 21, 29, 32, 33, 10, 3, 8, 16, 26, 12, 13] as well as global space-time analysis [4]. The global techniques are generally more robust. However, these techniques require complete knowledge of the space-time trajectories. Therefore, they are memory bound and computationally prohibitive. Furthermore, in an dynamic scene with autonomous moving objects the full motion trajectories are not generally known in advance. A similar situation arises in an interactive navigation through a virtual environment. For interactive rendering rates the local methods are more suitable. In the last few years, significant contributions in this area have been packaged in a series of general purpose collision detection software systems such as RAPID [12, 13], I-COLLIDE [8], Q-COLLIDE [7], V-COLLIDE [18], and V-Clip [22]. Most of this work is based on the seminal work of Lin and Canny [21] who proposed an efficient computation of the closest features between pairs of objects.

The *separating axis theorem* [12] has made the elimination of noncolliding objects fast and practical. Now, it is possible to efficiently bound objects within more tightly fit **O**riented **B**ounded **B**oxes (**OBB**) than **A**xis **A**ligned **B**ounded **B**oxes (**AABB**) [13]. This dramatically improves the performance for collision tests. By exploiting the temporal and geometric coherence one can reduce the all pairs testing of arbitrary polyhedral objects from $O(N^2)$ to $O(N + S)$ where N is the number of objects in the scene and S is the number of pairwise

overlaps [8] when the coherence is preserved. This result is extremely useful when the relative orientation between objects does not change and collisions may occur due to translational motion alone. However, in many applications objects change their relative orientation triggering a recomputation of the closest feature sets.

The *separating vector technique* has been shown to improve the bounding box overlap tests, Chung and Wang [7]. In this work, the authors also reported that the closest feature determination may add computational overhead when objects rotate and the feature sets need to be dynamically changed frequently.

The intrinsic complexity of object-space collision detection methods prompted us to further explore the rendering process and devise more efficient collision detection methods. An obvious direction has been the exploration of the *image-space* collision detection methods.

5.3.2 Image-space Collision Detection

The area of image-space collision detection has recently opened new avenues for improving the performance bounds [25]. The image-space interference test is based on projecting the object geometry onto the image plane and performing the analysis in a dimensionally reduced space with the depth map maintained in an image buffer, such as the Z-buffer.

Little has been reported on the image-space interference techniques. The pioneering work of Shinya and Forque [28] and Rossignac [27] lead to a simpler and more efficient algorithm for interference detection of interacting complex solid objects. The premise of this work is based on an enhancement in the graphics hardware support. This requires checking the change in the stencil buffer state from one frame to the next at the hardware level. If this hardware assisted test was available, the algorithm proposed by Myszkowski et al [25] would be a very practical solution to the interference problem between complex solid objects. However, such hardware enhancements are not currently available.

As our previous experiments show [2], without the hardware-assisted stencil buffer check, the collision detection becomes prohibitively slow due to the very large number of redundant memory accesses required to simulate the change in the stencil buffer state. However, we have found that it is still practical to use the conventional stencil buffer and Z-buffer for collision detection provided that we reduce the region for testing the stencil buffer state [2].

5.4. Ray Casting

The interference of two convex objects A and B in 3D space can be analyzed from the point of view of basic ray casting. The projection of the sampled 3D space onto the image plane gives a sequence of images.

The problem can then be reduced to a one-dimensional interference analysis based on the principle that a potential interference occurs if

$$R(A) \cap R(B) \neq \emptyset \text{ and } I(A) \cap I(B) \neq \emptyset \quad (5.1)$$

where $R(A)$ and $R(B)$ are the regions occupied by object A and B , respectively, on the plane orthogonal to the ray. $I(A)$ and $I(B)$ are the corresponding intervals of A and B occupied along the ray. As shown in Figure 5.3, there are nine possible cases which determine the interference between two objects.

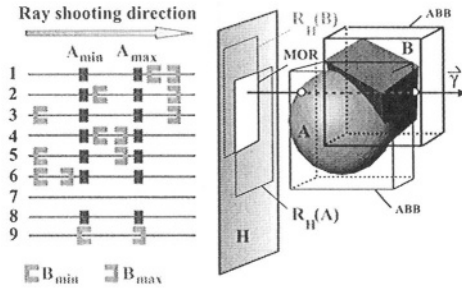


Figure 5.3. Possible interference cases along a ray; (1) object A is in front of object B ; (2-5) objects A and B overlap; (6) object B is in front of object A .

Two convex objects do not collide if and only if one of the cases 1, 6, 7, 8 or 9, occurs for every ray cast into the scene. In general, it is infeasible to cast an infinite number of rays into the scene. We can, however, reduce the ray sampling space as follows. First, we select a ray direction $\vec{\gamma}$ for all rays. We define a plane H with normal $\vec{\gamma}$. We place H at a relatively distant position from the objects in the scene. Let $R_H(A)$ and $R_H(B)$ be the regions covered by the projections of the axis-aligned bounding boxes (AABBs) of A and B on H . The regions $R_H(A)$ and $R_H(B)$ will be rectangular. Now, we only need to consider the rays cast from the minimum overlapping region $MOR = R_H(A) \cap R_H(B)$ into the half-space containing objects A and B along the direction $\vec{\gamma}$. The subset of rays cast from the MOR is chosen so that we can trade-off the accuracy of the collision detection with the computation of the interference test based on the intersection of the rays with objects in the scene. In order to do this, we subdivide the MOR into grid cells, Figure 5.4. From

each cell, we cast a single ray. The number of rays cast will be the same as the number of grid cells in *MOR* .

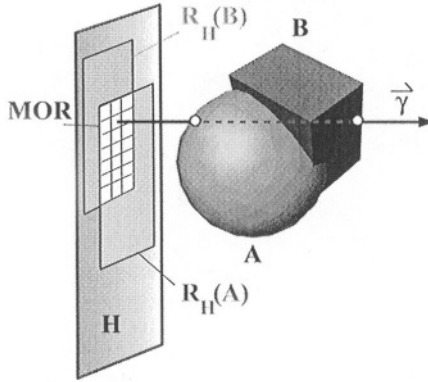


Figure 5.4. Subdividing the MOR and casting rays.

In Figure 5.4, each grid cell represents a pixel. The process of subdividing the *MOR* and casting rays can be construed as defining a viewport region (VPR) and applying an orthographic projection on the objects inside an orthogonal viewing volume (OVV).

We observe that it is not necessary to compute all the intersection points between the ray and two convex objects that potentially collide. For a convex object there will be two such intersection points at Z_{min} and Z_{max} . The projected Z -interval of the ray going through a convex object is given by a pair (Z_{min}, Z_{max}) of depth values. If the Z -intervals of two objects do not overlap, the objects are disjoint along the ray.

No collision between two objects will occur if for every pixel covered by the two objects the corresponding intervals do not overlap. In the rasterization domain, the computation of this test is influenced by: (1) the depth buffer resolution, (2) the viewport size and resolution, and (3) the viewing direction $\vec{\gamma}$.

The viewport acts as a spatial resolution filter that magnifies or reduces the view onto the possible region of collision. An optimal viewport size depends naturally on the domain of application and the density of the features per unit volume. Increasing the size of the viewport increases the resolution of the possible region of collision but it will also require more z -tests (more pixels) and a higher number of stencil buffer entries to be checked. Ideally, the viewport should be kept as small as possible while not compromising the coverage of the collision region.

Based on the factors of influence listed above, we develop a set of rules to determine whether two objects collide or not. These are shown in Figure 5.3. We only consider the pixels which are overlapped by both objects. For example, case (1) is equivalent to a successful test

$$Z_{max}^A < Z_{min}^B \quad (5.2)$$

where Z_{max}^A and Z_{min}^B are the the maximum and the minimum depth values of A and B, respectively, at a particular pixel. We can develop similar conditional tests for other cases if we know the intervals (Z_{min}, Z_{max}) for each pixel.

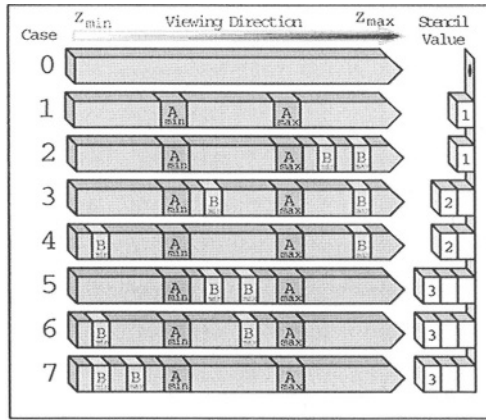


Figure 5.5. Possible Z-interval cases and the corresponding stencil buffer values.

The stencil buffer (SB) is a mask buffer that covers the entire image plane. We use it to restrict the drawing of geometric objects to small portions of the frame buffer. The Z-buffer algorithm is modified so as to embed the stencil buffer into the rendering pipeline. The possible values in the stencil buffer depend on the case of the z-interval overlap between two objects, A and B, as shown in Figure 5.5.

5.5. Rendering Passes

Unfortunately, in the rasterization system, we only have one depth value at any given time of rendering. That is, we know all the values of Z_{max}^A of object A for each pixel after A is rendered by the system. However, we have no knowledge of the exact values of Z_{min}^A , Z_{min}^B , and Z_{max}^B unless we set up the configuration of the system and render the objects again. However, if we do this, we will lose the previous depth values. This is due to the fact that the

system has only one depth buffer. If the test in Eq. (5.2) fails, we need to perform a second rendering pass.

In order to overcome this difficulty, we count the number of successful passes for the test

$$Z_{max}^A > Z^B \quad (5.3)$$

where Z^B is the depth value for B. The count is stored in a 2-D array SC. The dimension of SC is exactly the same as the dimension of VPR. There is one-to-one correspondence between the elements in SC and the pixels in VPR. Let $SC(p)$ (where $p \in VPR$) be the counter associated with a pixel p . If $SC(p) == 0$, then B is entirely behind A at pixel p , or B does not overlap p . If we obtain the same result for all pixels in VPR, then there is no collision and we terminate the process.

The possible values of $SC(p)$ are 0, 1, and 2. We summarize the value of $SC(p)$ and the interference status for each case in Table 5.1. Each case is illustrated in Figure 5.3.

Table 5.1. The pixel mask value SC(p) after the first rendering pass.

Case	$SC(p)$	Collision status	Second Pass
1:	0	False	No
2:	1	True	No
3:	1	True	No
4:	2	*True	Yes
5:	2	*True	Yes
6:	2	*False	Yes

From Table 5.1, there is a clearance between objects A and B at p when $SC(p) == 0$. When $SC(p) == 1$, objects A and B collide with each other. However, if $SC(p) == 2$, we do not know whether A and B collide or not. This happens only in cases 4, 5, and 6. We cannot distinguish these three cases by only applying the counting test (5.3). This problem can be solved by reversing the rendering order of A and B. The symbols A and B are exchanged in these tests. Then, case 4 becomes case 3, case 5 becomes case 2, and case 6 becomes case 1. Since cases 1, 2 and 3 can be resolved without any further processing, we can conclude that we need at most two rendering passes to determine the interference between objects A and B.

Please note, that the stencil buffer counts above are correct if and only if the hardware has the property that no point is rendered twice. This is specifically necessary for boundary points. On the hardware that we have tested the algorithm, specifically SGI Octane, Onyx2 and nVidia GeForce cards this property seems to hold specifically when the OpenGL setting of "not-equal" is

used when writing into the z-buffer. This situation can occur when multiple cone-like objects intersect at a single point, the apex.

5.6. Interference Region

The location of each object can be determined by its geometric transformation matrix M^X with respect to the world coordinate system. We set the viewing volume to determine which parts of the objects are to be projected onto the viewport, as well as the projection transformation. This is set by defining a global projection matrix. During rasterization, the viewport will contain the region of possible intersection between the two objects. This is akin to *zooming into* the region of interference in order to resolve a possible collision event.

The minimum rectangular area which covers the interference region and is projected onto the viewport $(X_{min}, X_{max}, Y_{min}, Y_{max})$ is computed by reading the block of depth values z associated with this region. If we know the pixel coordinates (x, y) in the viewport and its depth (z) , we can compute the coordinates (x', y', z') of its corresponding point in the world coordinate system. This method can be extended to testing for collision events between *simple concave polyhedra*. A *simple concave polyhedron* is an object that can be intersected by a straight line in at most four points on its boundaries. For example, a torus is a simple concave object.

5.7. Optimal MOR's

An optimal algorithm for finding the MOR between two objects can be developed based on the integration of object-space culling and image-space intersection computations in order to optimize the process at all levels and balance the computational load between the dynamic scene synthesis and hardware rendering.

Two observations can be in order here. The first observation is that the MOR is too loose to cover the actual contact region. The contact region often occupies a small percentage of the MOR. This is due to the fact that the an initial AABB is used to compute the new AABB' for each object. The new AABB' does not bound the object tightly. The second observation is that often all the faces of both objects are passed to the renderer. Some of the faces which are outside the sweeping volume of MOR along $\vec{\gamma}$ do not have any contribution for the interference test. In order to both minimize the faces rendered and provide an optimal projection plane, we must select a direction $\vec{\gamma}$ such that the ratio of the contact region to MOR is as high as possible, i.e. close to one, and the number of faces inside the sweeping volume of MOR is small. Then, there are three tasks to be performed. The first task is to construct a tight AABB' quickly

without using the initial AABB. The second task is to locate the local faces which are lying within the sweeping volume of MOR only. This requires an efficient data structure. The third task is to design a data structure to store the faces. This data structure is used to locate all the faces which are overlapped with an infinite sweeping volume along some axis $\vec{\gamma}$.

5.7.1 Locating the MOR's

In order to locate the MOR of two objects A and B in close proximity, we must first find a directional vector $\vec{\gamma}$.

Initially, we can compute the AABBs for each object in its own local frame. Let $AABB_c = (x, y, z, 1)$ be the center of AABB. First, we choose $\vec{v} = AABB_c^B - AABB_c^A$. Then, we rotate $\vec{v}$ to align it with the x-axis and select the z-axis as $\vec{\gamma}$. The reason for choosing $\vec{v} = AABB_c^B - AABB_c^A$ is the following. Consider two intersecting disks A and B in the plane. Obviously, $AABB_c^A$ and $AABB_c^B$ are the centers of A and B, respectively. We rotate $(AABB_c^B - AABB_c^A)$ so as to align it to the x-axis. Let $AABB_c^A = (x^A, y^A, z^A, 1)$, $AABB_c^B = (x^B, y^B, z^B, 1)$, and $\vec{v} = (x, y, z, 1)$, where $(x, y, z) = (x^B, y^B, z^B) - (x^A, y^A, z^A)$. We may choose the following matrix M^v to transform $\vec{v}$:

$$M^v = \begin{pmatrix} \frac{y}{D_1} & -\frac{x}{D_1} & 0 & 0 \\ \frac{xz}{D_1 D_2} & \frac{yz}{D_1 D_2} & -\frac{D_1}{D_2} & 0 \\ \frac{x}{D_2} & \frac{y}{D_2} & \frac{z}{D_2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5.4)$$

where $D_1 = \sqrt{x^2 + y^2}$ and $D_2 = \sqrt{x^2 + y^2 + z^2}$.

We compute the tight $AABB'^A$ and $AABB'^B$ for A and B. If the objects just touch each other, we can obtain the MOR such that its area is the smallest possible area among all the other MOR' along any coordinate system within the plane. The following theorem characterizes the smallest MOR region that can be practically achieved. The proof of this theorem is given in Appendix A.

Theorem 1 *Let A and B be two intersecting disks with radii r_1 and r_2 , respectively. Assume that $r_1 \geq r_2$, A_c is at the origin and B_c is in the first quarter of the plane. The distance between the centers is $R = (r_1 + r_2) - \delta$, where δ is a small positive real number. Then the smallest MOR occurs when B_c lies on the x-axis (or on the y-axis by symmetry).*

Since the area of MOR is the smallest and the colliding region area does not change in all cases in 2D, the ratio of the 2D image of the colliding region area

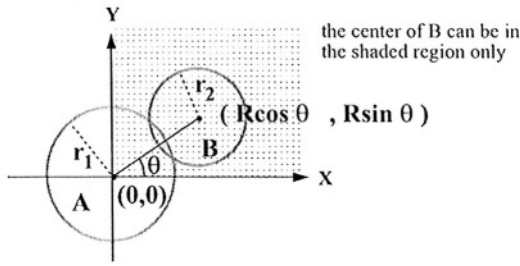


Figure 5.6. Locating a MOR: two disks A and B intersect each other; R is the distance between the centers of the disks.

to the area of MOR is the highest among all cases. Moreover, the sweeping volume of $\overline{MOR}$ is the smallest. This has a high probability to cover fewer faces of the objects than any other MOR' chosen. This is valid if the objects just penetrate each other.

We can also show that if $r_1 \geq (1 + \sqrt{2})r_2$ then the area of MOR is the smallest without employing any other constraints. The bound of δ can be removed. The proof of this statement is given in Appendix A.

Our next task is to find a tight AABB for each object.

5.7.2 Searching for Local Faces

In this section, we present a method to identify the local faces, which overlap with the infinite sweeping volume of MOR along the z-axis in the world space. We employ a local search strategy. The sweeping volume of MOR along the z-axis in the world space is transformed into the local frame of the object. We test for interference between the faces of a polytope and the sweeping volume. The selected faces that are passed into the rasterization system of the rendering pipeline, are those which overlap with the sweeping MOR volume. We can group the faces into clusters. If the bounding boxes of the clusters overlap with the volume, all the faces in the clusters are selected.

In our implementation, we use a binary partition scheme to group faces into clusters. We first compute the three-dimensional AABB of each object locally. Then, we choose a cutting plane C parallel to the x-y plane and passing through $AABB_c$ (the centroid of the AABB). This plane divides the object into two sets A_1 and A_2 , such that

$$FaceSet(A_1) \cup FaceSet(A_2) == FaceSet(O)$$

where O is the entire object and $FaceSet(O)$ represents the set of all faces of O . All the faces that lie entirely inside the half space of C^+ , belong to A_1 . Those faces which lie entirely inside the half-space of C^- , belong to A_2 . The

faces, which are located partially inside C^+ and C^- , are split into triangles. We assign the split triangles to C^+ or C^- , Figure 5.7. Then, we compute the AABBs and $AABB_c$ for A_1 and A_2 , respectively. We choose another cutting plane C' parallel to the x-z plane. This subdivision scheme is applied to A_1 and A_2 recursively. The cutting plane is selected in the order x-y, x-z and then y-z. This cycle is repeated at each iteration of the subdivision and a tree structure is constructed.

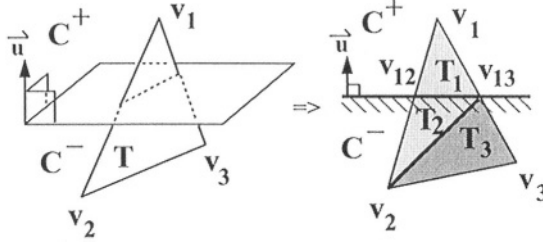


Figure 5.7. Triangle T is split into T_1 , T_2 and T_3 . T_1 lies inside C^+ while T_2 and T_3 lie inside C^- .

If a face lies partially in C^+ and C^- , the face is cut into smaller triangles so that the triangles will entirely lie inside C^+ or C^- . In Figure 5.7, a triangle T has 3 vertices v_1 , v_2 and v_3 . Vertex v_1 is inside C^+ while v_2 and v_3 are inside C^- . We find the intersection points of the plane C and two edges, say $\overline{v_1v_2}$ and $\overline{v_1v_3}$. Let $v_{12} = C \cap \overline{v_1v_2}$ and $v_{13} = C \cap \overline{v_1v_3}$. We construct three triangles T_1 , T_2 and T_3 , where $T_1 = (v_1, v_{12}, v_{13})$, $T_2 = (v_{12}, v_2, v_3)$ and $T_3 = (v_{13}, v_2, v_3)$. Now, we split triangle T into three pieces so that each piece is entirely inside either C^+ or C^- .

5.7.3 A Hybrid Collision Detection Approach

In this section, we present a hybrid object-space and image-based collision detection algorithm, **HYCODE**, **HY**brid **C**ollision **D**etection, Figure 5.8. In this algorithm, we use the separating vector algorithm as the front-end of our collision detection algorithm. We set the maximum number of iterations in the separating vector algorithm loop to five. From the result stated in [7], more than 90% of the non-colliding pairs can be eliminated in five iterations. This is consistent with the results obtained from our experiments. The operation cost of the separating vector algorithm is comparatively low. If no separating vectors are found within five iterations, we invoke our image-based algorithm to further test for interference. Once the collision event is detected, the colliding region can be easily computed.

```

1  proc HYCODE(A,B)  $\equiv$ 
2    collide = FALSE
3    if (ExistSeparatingPlane(A,B)==FALSE)
4       $\vec{v} = ABB_c^B - ABB_c^A$ 
5       $M \leftarrow$  rotation matrix that aligns  $\vec{v}$  to x-axis
6       $M'^A = M M^A$  //  $M^A$  : Transformation of A
7       $M'^B = M M^B$  //  $M^B$  : Transformation of B
8       $ABB^A = \text{ComputeTightABB}(A, M'^A)$ 
9       $ABB^B = \text{ComputeTightABB}(B, M'^B)$ 
10      $MOR' = \text{ComputeMOR}(ABB^A, ABB^B)$ 
11      $MOR^A = \text{InfiniteSweepVolume}(MOR, M'^A)$ 
12      $MOR^B = \text{InfiniteSweepVolume}(MOR, M'^B)$ 
13      $FaceList^A = \text{LocateLocalFaces}(A, MOR^A)$ 
14      $FaceList^B = \text{LocateLocalFaces}(B, MOR^B)$ 
15     collide=
16     Collide(  $FaceList^A, FaceList^B, M'^A, M'^B, MOR'$  )
17   fi
18   return(collide)
19 .

```

Figure 5.8. The HYCODE Algorithm.

5.7.4 Speedup due to MOR

Since HYCODE operates both in the object-space and on projection sampling at the image-space level, it benefits from both optimizations due to culling in the object-space as well as from image-based sampling optimizations. Thus, isolating the MOR achieves two purposes: (1) reduce the number of primitives to be considered in the collision tests, and (2) enhance the possible region of interference. In the table below we have taken a few model samples to show the speedup achieved by collision tests performed with MOR vs tests performed without MOR. Even for models with relatively low triangle count, the speedup ranges from 1.125 to 3.9 times.

Table 5.2. Speedup due to object-space culling and isolating the MOR.

No. of triangles	Time (msec) of collision tests	Time (msec) of collision tests	Speedup
	without MOR	with MOR	
800	3.6	3.4	1.125
9800	13.9	4.6	3.022
20000	25.4	6.5	3.908

HYCODE has two essential bottlenecks: (1) rendering the object surfaces, and (2) reading or inspecting the state of the stencil buffer in a software emulation mode.

In order to solve the first problem, we may employ a caching scheme in order to avoid re-rendering candidate objects. If the available buffers (Z and stencil buffers) are large enough, the objects can be prerendered, their depth values and their pixel masks can be stored in the image buffers. If the buffers are limited, we can subdivide the buffers into several parts. By exploiting the object space coherence and temporal coherence, we can cache the object pairs which are required for the rendering passes, in the next iteration. For a solution to the second problem, we need the hardware implementation for checking the status of the stencil buffer.

Acknowledgments

The work described in this chapter was supported in part by the Hong Kong Research Grants Council/CERG grants HKUST 6089/99E and PolyU B-Q566.

References

- [1] G. Baciú and S.K. Wong. "Image-based techniques in a hybrid collision detector". *IEEE Trans. on Visualization and Computer Graphics (TVCG)*, vol. 9, no. 2, pp. 254-271, April, 2003.
- [2] G. Baciú and S.K. Wong. "Rendering in object interference detection on conventional graphics workstations". In *Proceedings of the Pacific Graphics '97*, pp. 51-58. IEEE, Oct, 1997.
- [3] S. Bandi and D. Thalmann. "An adaptive spatial subdivision of the object space for fast collision of animated rigid bodies". In *Eurographics 95*, pp. 259-270, Maastricht, August, 1995.
- [4] S. Cameron. "Collision detection by four-dimensional intersection testing". *IEEE Trans. on Robotics and Automation*, vol. 6, no. 3, pp. 291-301, June, 1986.
- [5] J. F. Canny. "Collision detection for moving polyhedra". *IEEE Trans. PAMI*, vol. 8, pp. 200-209, March, 1986.
- [6] B. Chazelle and D. P. Dobkin. "Intersections of convex objects in two and three dimensions". *Journal of the ACM*, vol. 34, pp. 1-27, 1987.
- [7] K. Chung and W. Wang. "Quick collision detection of polytopes in virtual environments". In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 125-132, ACM, July, 1996.
- [8] J. Cohen, M. Lin, D. Manocha and M. Ponamgi. "I-COLLIDE: An interactive and exact collision detection system for large-scale environments". In *Proc. of ACM Interactive 3D Graphics Conference*, pp. 189-196, ACM, 1995.

- [9] A. Foisy, V. Hayward and S. Aubry. "The use of awareness in collision prediction". In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 338-343, IEEE, 1990.
- [10] A. Garcia-Alonso, N. Serrano and J. Flaquer. "Solving the collision detection problem". *IEEE Computer Graphics and Applications*, vol. 13, no. 3, pp. 36-43, 1994.
- [11] E. G. Gilbert, D. W. Johnson and S. S. Keerthi. "A fast procedure for computing the distance between objects in three-dimensional space". *Journal of Robotics and Automation*, vol. 4, pp. 193-203, 1988.
- [12] S. Gottschalk. "Separating axis theorem". Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill, 1996.
- [13] S. Gottschalk, M. Lin and D. Manocha. "OBBTree: A hierarchical structure for rapid interference detection. In *Computer Graphics (SIGGRAPH'96)*, pp. 171-180, ACM, Aug 1996.
- [14] B. V. Herzen, A. H. Barr and H. R. Zats. "Geometric collision for time-dependent parametric surfaces". *Computer Graphics (SIGGRAPH'90)*, vol. 24, no. 4, pp. 39-48, 1990.
- [15] K. E. Hoff, J. Keyser, M. Lin, D. Manocha and T. Culver. "Fast computation of generalized voronoi diagrams using graphics hardware". *Computer Graphics (SIGGRAPH'99)*, pp. 277-286, 1999.
- [16] P. M. Hubbard. "Collision detection for interactive graphics applications". *IEEE Transactions of Visualization and Computer Graphics*, vol. 1, no. 3, pp. 218-228, 1995.
- [17] P. M. Hubbard. "Approximating polyhedra with spheres for time-critical collision detection". *ACM Transactions on Graphics*, vol. 15, no. 3, pp. 179-210, 1996.
- [18] T. Hudson, M. Lin, J. Cohen, S. Gottschalk and D. Manocha. "V-COLLIDE: Accelerated collision detection for VRML". In *Proc. of VRML'97*, ACM, 1997.
- [19] M. Kelley, K. Gould, B. Pease, S. Winner and A. Yen. "Hardware accelerated rendering of CSG and transparency". *Proceedings of the 21st annual conference on Computer graphics*, pp. 177-184, 1994.
- [20] J. T. Klosowski, M. Held, S. B. J. Mitchell, H. Sowizral and K. Zikan. "Efficient collision detection using bounding volume hierarchies of k-DOPs". *IEEE Trans. on Visualization and Computer Graphics*, vol. 4, no. 1, Jan, 1998.
- [21] M. C. Lin and J. F. Canny. "Efficient collision detection for animation". In *Third Eurographics Workshop on Animation and Simulation*, Cambridge, September, 1992.
- [22] B. Mirtich. "V-Clip: Fast and robust polyhedral collision detection". *ACM Trans. on Computer Graphics*, vol. 17, no. 3, pp. 177-208, Jul, 1998.
- [23] B. Mirtich and J. Canny. "Impulse-based simulation of rigid bodies". In *Proceedings of 1995 Symposium on Interactive 3D Graphics*, pp. 181-188, 1995.
- [24] M. Moore and J. P. Wilhelms. "Collision detection and response for computer animation". In *Computer Graphics (SIGGRAPH'88)*, vol. 22, no. 4, pp. 289-298, ACM, 1988.
- [25] K. Myszkowski, O. G. Okunev and T. L. Kunii. "Fast collision detection between computer solids using rasterizing" graphics hardware". *The Visual Computer*, vol. 11, pp. 497-511, 1995.
- [26] M. Ponamgi, D. Manocha and M. C. Lin. "Incremental algorithms for collision detection between general solid models". In *Proceedings of ACM/SIGGRAPH Symposium on Solid Modeling*, pp. 293-304, ACM, 1995.

- [27] J. Rossignac, A. Megahed and B. O. Schneider. "Interactive inspection of solids: cross-section and interferences". *Computer Graphics (SIGGRAPH'92)*, vol. 26, no. 2, pp. 353-360, July, 1992.
- [28] M. Shinya and M. Forgue. "Interference detection through rasterization". *Journal of Visualization and Computer Animation*, vol. 2, pp. 131-134, 1991.
- [29] J. Snyder, A. R. Woodbury, K. Fleischer, B. Currin and A. H. Barr. "Interval methods for multi-point collisions between time" dependent curved surfaces". In *Computer Graphics (SIGGRAPH'93)*, pp. 321-334, ACM, Aug, 1993.
- [30] S. Suri, P. M. Hubbard and J. F. Hughes. "Collision detection in aspect and scale bounded polyhedra". In *Proc. of Ninth ACM-SIAM Symp. Discrete Algorithms*, ACM, Jan, 1998.
- [31] W. C. Thibault and B. F. Naylor. "Set operations on polyhedra using binary space partitioning trees". *Computer Graphics (SIGGRAPH'87)*, vol. 21, no. 4, pp. 153-162, July, 1987.
- [32] Y. Yang and N. Thalmann. "An improved algorithm for collision detection in cloth animation with human body". In *First Pacific Conference on Computer Graphics and Applications*, pp. 237-251, 1993.
- [33] M. J. Zyda, D. R. Pratt, D. Osborne and J. G. Monahan. "NPSNET: Real-time collision detection and response". *The Journal of Visualization and Computer Animation*, vol. 4, pp. 13-24, 1993.

Chapter 6

FOURIER PROCESSING IN THE GRAPHICS PIPELINE

Edward Angel ¹ and Kenneth Moreland ²

1. Computer Science Department, University of New Mexico, Albuquerque, NM 87131, USA

2. Sandia National Laboratories, University of New Mexico, Albuquerque, NM 87185, USA
angel@cs.unm.edu, kmorel@sandia.gov

Abstract The latest generation of graphics hardware has new capabilities that give commodity cards the ability to carry out algorithms that formerly were only possible on standard CPUs or on special purpose parallel hardware. The ability to carry out floating point operations with large arrays on Graphics Processing Units (GPUs) makes them attractive as image processors. In this article, we discuss the implementation of the Fast Fourier Transform on a GPU and demonstrate some applications.

Keywords: Fast Fourier Transform, GPU, graphics pipeline, image processing, signal processing

6.1. Introduction

Over the past 20 years, there has been a convergence of graphics and imaging systems. While formerly users would have separate systems, we no longer make this distinction. From both the hardware and the software perspectives, we have seen the merging of Computer Generated Imaging (CGI) and image processing software. For example, the OpenGL API supports image processing through its widely-used imaging extensions. The capabilities of CGI products such as MayaTM and LightwaveTM include imaging operations.

Nevertheless, if we look more closely at the hardware, we can see that until recently, there were serious limitations in what was really possible, especially in real time. Lack of floating-point support in the graphics pipeline and the inability to control the graphics pipeline were serious problems.

Graphics hardware has evolved to the point that we now have programmable pipelines and floating-point frame buffers. A consequence of these developments is that we can now do true image processing as part of the graphics pipeline. In this paper, we shall show how to use fragment programming to do image processing via two dimensional Fast Fourier Transforms (FFTs). We first presented these results in \cite{GH}. Here our focus will be somewhat different. Because most graphics practitioners are unfamiliar with signal processing, we shall start with an introduction to convolution and the use of the FFT for filtering. Then we will discuss the graphics pipeline. We will introduce the idea of fragment programming. We show how to implement FFT filtering with fragment programming. Finally, we will show some applications.

6.2. Convolution

A (digital) image is an $N \times M$ array of picture elements, or pixels, arranged on a rectangular grid. The basic idea behind standard image processing is that we can form a new image in which the value of each pixel is some mathematical function of the pixels values in the original image. Suppose that we have an $N \times M$ image.

$$\mathbf{f} = [f_{ij}], \quad 0 \leq i \leq N-1, 0 \leq j \leq M-1, \quad (6.1)$$

where f_{ij} is value of the pixel at located (i, j) . When we process $\mathbf{f}$, we form a new image $\mathbf{g}$ by combining the values of the pixels in $\mathbf{f}$ in some manner, or abstractly

$$\mathbf{g} = \Psi(\mathbf{f}), \quad (6.2)$$

for some function Ψ .

This formulation is obviously too general to be of much practical use. If the transformation is linear, then each g_{ij} is a linear combination of the values of $\mathbf{f}$. If the transformation is **spatially invariant**, then the function used to evaluate each processed pixel is not dependent on where the pixel is located in the image. Thus, each pixel is processed in the same manner. It is easy to show that if a transformation is both linear and spatially invariant, we can write the filtering operation as

$$g_{ij} = \sum_k \sum_l f_{kl} h_{i-k, j-l} \quad (6.3)$$

where the set of coefficients $\{h_{kl}\}$ characterize the particular transformation. At this point, we need not worry about the indices for the sums but in practice the values

of h_{kl} go to zero as the magnitudes of k and l increase¹. The filter is characterized by the array

$$\mathbf{h} = [h_{ij}], \quad (6.4)$$

and $\mathbf{g}$ is said to be the convolution of $\mathbf{f}$ and $\mathbf{h}$, usually written as

$$\mathbf{g} = \mathbf{f} * \mathbf{g} \quad (6.5)$$

Note that if the original image is an impulse at (0,0),

$$f_{ij} = \begin{cases} 1 & \text{if } i = 0, j = 0 \\ 0 & \text{otherwise,} \end{cases} \quad (6.6)$$

then

$$\mathbf{g} = \mathbf{h} \quad (6.7)$$

which leads to $\mathbf{h}$ being known as the **impulse response** of the filter.

Usually, the impulse response has a small area. That is

$$h_{ij} = 0 \quad \text{if } i^2 + j^2 > m \quad (6.8)$$

for some small integer m . In this case, the convolution can be computed in $O(NMm)$ operations rather than the $O(N^2M^2)$ required for the general case. For many imaging operations, such as edge enhancement, antialiasing, and smoothing, a small value such as $m=3$ is sufficient. The small array of nonzero elements is called the **convolution kernel**.

6.3. Hardware Implementation

Graphics systems have supported some instances of linear filtering. For example, OpenGL allows linear filtering of textures. However, only 2×2 filters are supported on most implementations and even this filtering operation incurs a performance penalty.

¹ For the mathematics of convolution to work properly, we must assume all functions are periodic with periods N and M . However, because filter functions tend to zero, we can embed the problem within a periodic framework and worry only about artifacts at the edges

We can also use OpenGL's accumulation buffer to support filtering with larger kernels, but the cost is proportional to the number of elements in the kernel. Depending on the hardware, there is often an additional significant performance penalty because of the data movement between processor memory and the accumulation buffer. Some systems support off-screen buffers for which there are OpenGL extensions that are often more efficient than the accumulation buffer.

Note that graphics hardware is also limited by the precision of the buffers that make up the frame buffer. Until recently, these buffers, with the exception of the (slow) accumulation buffer, were restricted to eight bits per color component, a limitation that made general filtering impractical. With the latest generation of hardware, the frame buffer works in floating point which solves some of these problems. However, we still need to worry about the number of floating point operations for large convolution kernels.

6.4. The Fourier Transform

The major computational advantage using the Fourier Transform is that it reduces the $O(NM)^2$ complexity of the convolution operation to $O(NM)$. For discrete data, we actually compute the Discrete Fourier Transform. Consider a one dimensional sequence of N complex-valued points, $\{f_i\}$ for $i=0,1,\dots,N-1$. Its DFT is given by the sequence

$$F_j = \sum_{i=0}^{N-1} f_i W_N^{-ij}, j = 0, \dots, N-1 \quad (6.9)$$

where W is the complex N th root of unity ($i^2 = -1$)

$$W_N = e^{i2\pi/N} = \cos 2\pi/N + i \sin 2\pi/N \quad (6.10)$$

The inverse DFT is given by

$$f_i = \frac{1}{N} \sum_{j=0}^{N-1} F_j W_N^{ij}, j = 0, \dots, N-1 \quad (6.11)$$

The discrete convolution of two N -point sequences $\{f_i\}$ and $\{h_i\}$ is given by

$$g_i = \frac{1}{N} \sum_{j=0}^{N-1} h_j f_{i-j} \quad (6.12)$$

where the indices are computed periodically.² We can show that Fourier transform coefficients of $\{g_i\}$ are given by [9].

$$G_i = H_j F_j \quad (6.13)$$

Thus, convolution in the original domain which is an $O(N^2)$ operation is an $O(N)$ operation in the Fourier or frequency domain. Of course, we have to add in the work to go back and forth between the domains, but we will address that issue with the FFT algorithm.

6.4.1 The Two-dimensional DFT

For a two dimensional sequence or array f_{ij} , the DFT is another two dimensional sequence

$$F_{kl} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_{ij} W_N^{-ik} W_M^{-jl} \quad (6.14)$$

If we rearrange this equation as

$$F_{kl} = \sum_{i=0}^{N-1} W_N^{-ik} \sum_{j=0}^{M-1} W_M^{-jl} f_{ij} \quad (6.15)$$

we see that the two-dimensional DFT is actually a sequence of one-dimensional DFTs on the rows of the matrix $\mathbf{f} = [f_{ij}]$, followed by a sequence of one-dimensional DFTs on the columns (or visa versa).

Figure 6.1 shows a color image of a 1024 x 1024 rendered Utah teapot. Figure 6.2 shows the magnitude of the FFT as an RGB image formed from the magnitudes of each of the FFTs of the color components of the original image.

6.4.2 The Fast Fourier Transformation

If we simply compute the DFT by its definition, the one-dimensional transform requires $O(N^2)$ operations and the two-dimensional transform requires $O(NM^2 + N^2M)$. Hence, for filtering an image, there would be no reason to use the DFT unless we could find a faster way to take the forward and backward transformations.

² Although most functions are not periodic, we can use periodic convolution by padding functions with zeros.



Figure 6.1. Utah teapot.

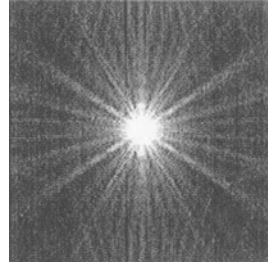


Figure 6.2. FFT of Utah teapot.

If N is a power of 2 ($N=2^y$), the Fast Fourier Transform algorithm [3] reduces the $O(N^2)$ operations for the one-dimensional transform to $O(N \log N)$. Using the same reasoning as before, if N and M are powers of 2, we can take the two-dimensional DFT in $O(NM(\log M + \log N))$ operations.

Although we do not need the derivation of the algorithm in order to use it, the derivation will give us hints on how to implement the algorithm on a graphics processor. The FFT is a divide and conquer algorithm. If we start with a sequence of $N=2^y$ elements $\{f_i\}$, $i=0, \dots, N-1$, we can divide it into two sequences, one of the odd elements $\{f_i^o\} = \{f_{2i+1}\}$, $i=0, \dots, N/2-1$ and a second of the even elements $\{f_i^e\} = \{f_{2i}\}$, $i=0, \dots, N/2-1$. Each of these sequences has its own $N/2$ dimensional DFTs, $\{F_j^o\}$ and $\{F_j^e\}$. We can then rewrite the definition of the DFT in terms of these half-size DFTs as

$$F_j = \sum_{i=0}^{N/2} f_j^o W_{N/2}^{ij} + W_N^j \sum_{i=0}^{N/2-1} f_i^e W_{N/2}^{ij} \quad (6.16)$$

$$= F_j^o + W_N^j F_j^e \quad (6.17)$$

Thus, the FFT of an N point sequence can be decomposed into two DFTs of $N/2$ point sequences. Applying this result recursively, the original DFT can be decomposed into 4 DFTs of $N/4$ points and so on until we get $N/2$ DFTs of 2 points. Each of these 2-point DFTs looks as in Figure 6.3 and requires 2 (complex) multiplications. This fundamental calculation is known as the basic butterfly. There is a complex multiplication along the directed edges followed by a complex sum of the results. All values are computed in complex pairs.

As we work our way back up the recursion, building up the DFT of the original N -point sequence, we can reduce all calculations to basic butterfly computations. Thus, the final calculation is $O(N \log N)$ or $O(\gamma N)$. The algorithm can be done in place, but because the pairs of points change in each calculation, there is some

tricky indexing in the implementation [8]. Figure 6.4 shows 3 stages of basic butterfly calculations used to compute the 8-point DFT.

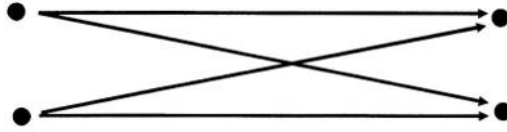


Figure 6.3. Basic butterfly.

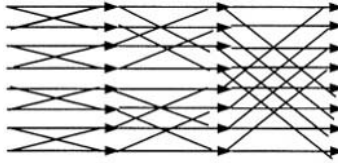


Figure 6.4. Data flow in 8-point FFT

6.4.3 Processing Real-valued Images

The Fourier Transform is a complex number to complex number transformation. However, with images we are usually working with only real numbers³. Thus even though the DFT of an $N \times M$ image has $2NM$ real numbers, only NM of them can be independent. We can use the conjugate symmetry property of the transform of real functions [11]

$$F_{ij} = F_{N-1, M-j}^* \quad (6.18)$$

Except for a few special values that are real numbers or at the edges, each element in the top half of the array has a matching conjugate in the bottom of the array, as in shown in Figure 6.5. Hence, we can save half the storage. However, what is even more interesting is that for real data we can show that we can use an $N/2$ transform to take the DFT of a sequence of N real numbers. We can also use an N -point transform to take the transform of two N -point real sequences simultaneously.

Consider two N -point real sequences $\{f_i\}$ and $\{g_i\}$. We can form the complex N -point sequence

$$h_i = f_i + \mathbf{i}g_i \quad (6.19)$$

³ For color images, we take the DFT of each color component.

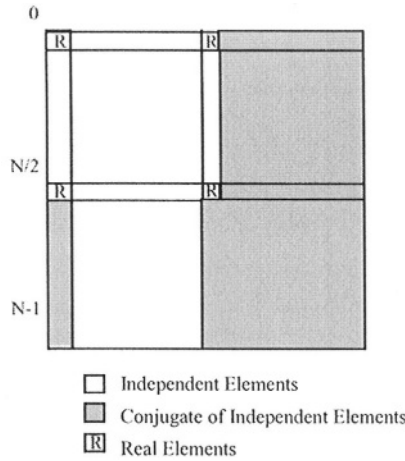


Figure 6.5. Symmetry in 2D DFT of a real image.

and find its DFT $\{H_j\}$ with the FFT. We can then compute the real and imaginary parts of the Fourier transform sequences $\{F_j\}$ and $\{G_j\}$ by

$$\Re(F_j) = \frac{1}{2} (\Re(H_j) + \Re(H_{N-j})), \quad (6.20)$$

$$\Im(F_j) = \frac{1}{2} (\Im(H_j) + \Im(H_{N-j})), \quad (6.21)$$

$$\Re(G_j) = \frac{1}{2} (\Im(H_j) - \Im(H_{N-j})), \quad (6.22)$$

$$\Im(G_j) = \frac{1}{2} (\Re(H_{N-j}) - \Re(H_j)). \quad (6.23)$$

Note that if we want to use this result to take the transform of an N -point real sequence with an $N/2$ -point transform, we have considerable flexibility in how we form $\{h_i\}$. For example, we could use the odd and even values of the original sequence to form two subsequences or we could use the first and second halves of the original sequence.

Putting together all these results, we can see that if we want to take the FFT of an $N \times M$ image and N and M are powers of 2, we can do it with $N/2$ and $M/2$

dimensional transforms. All calculations are basic butterflies. In terms of a program, there are $\log N + \log M$ stages. Probably, the trickiest part of the implementation is the indexing [8]. The data are reorganized by the butterfly calculations and by our desire to exploit the symmetry of the transform of real-valued data.

6.4.4 Filtering

Returning to filtering, we can process an image as in Figure 6.6 using a forward FFT, an element by element multiplication of coefficients, and an inverse FFT. The complexity for $N \times M$ images is $O(NM (\log N + \log M))$ operations.

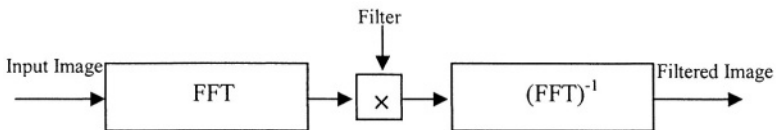


Figure 6.6. FFT filtering.

In terms of a graphics application, the most obvious way to do a convolution would be to use the CPU on the host machine and then send the image to the graphics processor and frame buffer with an OpenGL *glDrawPixels* function. The main bottleneck would then be the transfer of image data between the CPU and the graphics card, an operation that usually requires that the data be reformatted. If the data were already in the graphics processor, we would have to do two transfers between processor memory and the frame buffer (or texture memory) with a *glReadPixels* and a *glDrawPixels*. These transfers can require a significant amount of time. However, with recent hardware advances, we can use a fragment program to do the entire filtering operation on the graphics card.

6.5. Vertex and Fragment Programming

Virtually all realtime graphics systems use a pipeline architecture. A simplified version is shown in Figure 6.7. The front end is a vertex processor that works in a geometric three- or four-dimensional space. The back end works with bits that form the image in the frame buffer [1]. The rasterizer provides the link between them. The rasterizer must assemble the vertices into geometric objects and generate the bits that form the pixels in the frame buffer that are displayed. The outputs of the rasterizer are **fragments**⁴. Fragments determine the pixels that are

⁴ Most modern architectures support fragment processing after the rasterizer.

placed in the color buffers. Because most fragments are the size of a pixel, the terms *pixel* and *fragment* are often used interchangeably. However, a fragment along the edge of a polygon might be smaller than a pixel in size. A given fragment may not be visible. If a fragment is visible, it contributes to the color of a pixel in the frame buffer. Thus, if two vertices define a line segment, the rasterizer must generate the colors or shades between the vertices taking into account the type of line and its width. For polygons, the rasterizer determines shades for fragments in the interior of the polygons. The appeal of this architecture is that the various calculations are pipelined for efficiency and each block is supported directly by the hardware.

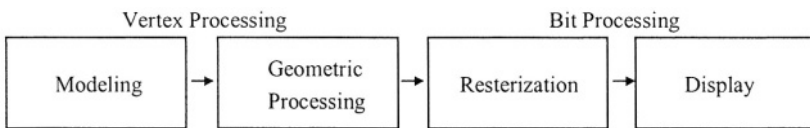


Figure 6.7. Graphics pipeline.

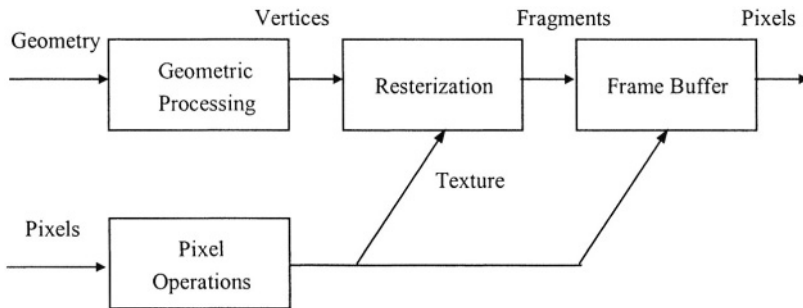


Figure 6.8. Expanded graphics pipeline.

Figure 6.8 shows more detail in the pipeline and represents what we would see in most mid- and high-end graphics cards. There is a separate pixel pipeline that allows an application program to place pixels directly in the frame buffer, work with textures, and perform some simple imaging operations. In general, the geometric pipeline includes Phong shading that is computed for each vertex. The geometric and pixel pipelines merge during rasterization.

Until recently, the pipeline was fixed. Some parameters could be altered from an application program and features such as lighting and texture mapping could be turned on and off. However, what was not possible was changing how each vertex or fragment was processed within the pipeline. Thus, we could not use

sophisticated shaders such as those that could be defined with the RenderMan [10] shading language. Although some pixel operations were possible, they often required the use of slow buffers, such as the accumulation buffer, or transferring pixels back to processor memory, a slow operation, or multiple rendering passes.

Although there were earlier research architectures that supported a flexible pipeline, the nVidia G Force 3 provided vertex programming in the commodity market. Now there is a variety of programmable graphics cards in the commodity market that allow for both vertex and fragment (or pixel) programming. We can look at these architectures as in Figure 6.9. As vertices are generated by the geometry, a vertex program can alter their colors, locations, or any other property associated with a vertex. We can write a program that will replace the Phong shader with another shader. Note that altering the attributes of a vertex need not affect any operation further down the pipeline. Thus, if we replace the Phong shader, the normal behavior of the rasterizer will still be to interpolate vertex colors over the polygons defined by the vertices.

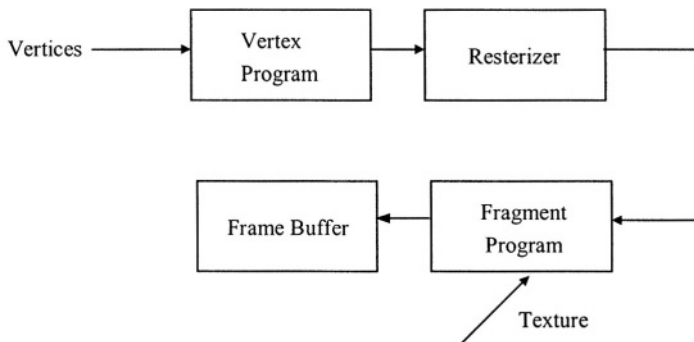


Figure 6.9. Programmable graphics pipeline.

Fragment (or pixel) programs work in much the same manner. As fragments are generated by the rasterizer, we can write an application program to change their colors or even to reinterpret color and texture data. Such a program can use data in texture maps and calculations within the code to determine new fragment colors. For example, fragment shaders can implement algorithms such as bump mapping within the GPU. We shall use fragment programs to carry out the FFT.

The latest generation of graphics processors has added another key feature: floating point frame and texture buffers. These are especially important for fragment programs. Older processors allowed for only one byte per color component. This limitation prevented applications from doing accurate compositing operations due to overflows. With colors and textures stored in

standard floating-point form, not only can we avoid using slow alternatives such as the accumulation buffer but we can now think of using color and texture buffers in new ways.

Programmable pipelines can be thought of in various ways. The most natural way is as stream processors, a view that comes out of the pipeline nature of the architecture. Another is as SIMD processors that execute the same instructions on all vertices or fragments in parallel. From this perspective, we can see that a vertex (or fragment) program can be seen as performing the same instruction in parallel on all vertices (or fragments). Whichever view we choose, these pipelines should be able to carry out numerical calculations at a high rate, especially those that involve arrays that can be placed in texture memory.

6.5.1 APIs

Initially, the control of programmable pipelines was done by low-level programming for the particular graphics processing unit (GPU). This process had all the standard deficiencies of assembly-language programming. Over the past few years, a few alternatives have been developed.

One was to use OpenGL's flexible extension mechanism to provide a C interface to particular GPUs. Although this approach allows the application programmer to control a GPU, the approach lacks generality, one of the expected benefits of a good API. Note that although the extensions allowed the application programmer to program the cards in C, in reality the programmer is downloading an assembly language program.

More recently, there have been two proposals for more general interfaces, Cg and the OpenGL Shading Language. Both seek to give the application programmer control of the GPU through a C-like interface. We have done our work with Cg [5] [7]. Alternatively, we could have used DirectX9 for our applications as it supports programmable shaders.

6.6. Using the GPU for the FFT

Returning to the algorithm, we can see that implementing it on a GPU is actually a very straight-forward task. We can pass the input data to the GPU as a texture map. However, we need geometric primitives to flow down the pipeline and cause the execution of fragment or vertex programs. In principle, we could form a single quadrilateral parallel to the projection plane that would be associated with the texture map and size it so that each fragment corresponds to a point in the original data set. As fragments are generated, they would cause the FFT to be computed by the fragment program.

However, there are a few details that make the process a bit more complex. Actually, we would need two quadrilaterals and two frame buffers because we

have to compute the real and imaginary parts of the FFT. Each would have its own fragment program. But that method would not take advantage of the symmetry of the transform of real-valued data. We must also take into account that the FFT is a multipass algorithm which takes $\log N + \log M$ iterations.

We solve the first problem by packing the computation and storage into a single frame buffer using the methods described in Section 6.4.3. Because we pack both real and imaginary values into one frame buffer, we often must load separate fragment programs generating the real and imaginary components and draw quadrilaterals over subregions on the screen pertaining to the respective values. We can solve the second problem by rendering into a texture rather than into the frame buffer. Thus each iteration reads from one buffer and writes into another. For the details, see [8].

6.7. Examples

The example in Figure 6.1 was computed on a nVidia FX5200 Ultra and a Quadro FX 3000. Figure 6.10 and 6.11 show standard low- and high-pass filters, respectively.



Figure 6.10. Low-pass filtering of teapot image.



Figure 6.11. High-pass filtering of teapot image.

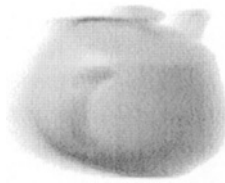


Figure 6.12. Motion blur of teapot image.

These examples are not particularly interesting because they could be carried out easily in the spatial domain with a small area convolution. Figure 6.12 is a bit more interesting. Here we see a motion blur on the teapot.

Usually we would form such an image by averaging a sequence of images with the teapot moved in each. Although this method would be correct and the FFT method being done on the images rather the object, nevertheless the visible appearance is remarkably good.

However, if the FFT is to be useful for graphics and imaging applications, we should look at applications that could not be done efficiently in the spatial domain, in particular applications with large convolution kernels. One example is texture generation [4]. Many textures exhibit a quasiperiodic behavior and can be characterized by distinct frequencies in the spectral domain that are spread over a wide range. One way to generate a family of such textures is to drive a filter with noise. Figures 6.13 and 6.14 were generated with a fixed magnitude spectrum and random phase, a process that was done entirely within the GPU.

6.8. Conclusions

At this point, the speed of our unoptimized implementation on an early Nvidia FX card is a little slower than running an optimised software implementation of the FFT on a fast Pentium. If we count operations, the GPU was carrying out operations at a rate of over 5 Gflops on the Quadro FX. For 512×512 images, we could achieve filtering at a 4 Hz rate.

However, the performance improvements in graphics cards are occurring at such a rate that this statement will not be true for long. Indeed, it might not be true as you read this chapter. More important is that in real applications, if we have an application that can benefit from transform domain processing on computer-generated images, we can avoid the bottleneck of reading images back into processor memory.

If we take a more generalized view of what we have done here, we can argue that GPUs are well suited for a wide variety of linear algebraic and imaging

operations. Of particular interest are multi-resolution methods that are presently of great interest in the graphics community. For some examples, see [6] and [2].

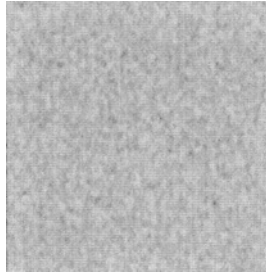


Figure 6.13. Texture created in the frequency domain.

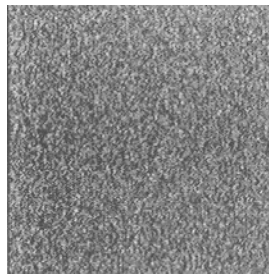


Figure 6.14. A Second texture created in the frequency domain.

Acknowledgment

This work was carried out at the University of New Mexico with the support of the National Science Foundation under Grant CDA-9503064; the Los Alamos Computer Science Institute; the DOE Mathematics, Information, and Computer Science Office; and Sandia National Laboratories. Sandia is a multi-program laboratory operated by the Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. We would also like to acknowledge the help of Ye Cong of the Computer Science Department at UNM for doing some of the examples and the NVidia Corporation for the donation of an the GPUs.

References

- [1] Angel. *Interactive Computer Graphics*, Third Edition, Addison-Wesley, 2003.
- [2] J. Bolz, I. Farmer, E. Grinspun and P. Schroöder. "Sparse matrix solvers on the GPU: Conjugate gradients and mltigrid." In *Computer Graphics*, ACM SIGGRAPH 2003, pp. 917-924, July, 2003.
- [3] J. W. Cooley and J. W. Tukey. "An algorithm for the machine calculation of complex fourier series". In *Mathematics of Computation*, vol. 19, no. 90, pp. 297-301, 1965.
- [4] S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin and S. Worley. *Texturing and Modeling: A Procedural Approach*, Second Edition, Academic Press Professional, 1998.
- [5] R. Fernando and M.J. Kilgard. *The CG Tutorial*, Addison Wesley, 2003.
- [6] J. Krüger and R. Westermann. "Linear algebra operations for GPU implementation of numerical algorithms". *Computer Graphics*, ACM SIGGRAPH 2003, pp. 908-916, July, 2003.
- [7] W. R. Mark, R. S. Glanville, K. Akeley and M. J. Kilgard. "CG: A system for programming graphics hardware in a C-like language". *Computer Graphics*, ACM SIGGRAPH 2003, pp. 896-907, July, 2003.
- [8] K. Moreland and E. Angel. "Computing the FFT on a GPU". In *Graphics Hardware*, pp. 112-119, July, 2003.
- [9] A. V. Oppenheim and R. W. Schaffer. *Digital Signal Processing*, Prentice-Hall, 1975.
- [10] S. Upstil. *The RenderMan Companion*, Addison-Wesley, 1989.
- [11] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes in C++*, Second Edition, Cambridge University Press, 2002.

Chapter 7

TRANSFORMATION IMAGE INTO GRAPHICS

Zonghua Zhang ^{1, 2}, Xiang Peng ^{1, 3} and David Zhang ⁴

1. State Key Laboratory of Precision Measurement Technology and Instruments, Tianjin University, China

2. Biomotion Lab, Department of Psychology, Ruhr-University-Bochum, Germany

3. Institute of Optoelectronics, Shenzhen University, China

4. Department of Computing, The Hong Kong Polytechnic University, Hong Kong

zhzhangju@hotmail.com, xpeng@tju.edu.cn, csdzhang@comp.polyu.edu.hk

Abstract This chapter focuses on the transformation of range images (2.5D) into three-dimensional (3D) graphical models. Beginning with an overview of state-of-the-art range image acquisition methods, it provides an introduction to the construction of complete 3D graphical models using multi-view range images registration and integration. The description of these transformation processes emphasizes a 'snapshot' three-dimensional imaging and modeling (3DIM) approach based on full-field fringe projection and phase mapping technology. We also introduce a novel colour texture acquisition method and discuss popular approaches to the geometric description of 3D graphical models resulting from range image registration and integration. In our conclusion we discuss directions for future work.

Keywords: Range image, graphical model, computer graphics, machine vision, 3D imaging sensor, fringe projection, phase mapping, colour texture, registration, visualized registration, Iterative Closest Point (ICP) algorithm, integration.

7.1. Introduction

With the rapid development of advanced display hardware and geometric representation software, photo-realistic rendering techniques have become an important feature of computer graphics, providing high-quality real world 3D shapes, often with characteristics such as intensity, colour, or texture. Where the objects and environments represented in many applications exist in the real world – whether they are virtual museums or human beings – people have higher

expectations of the quality of their imaging and modeling. However, because such objects or scenes are complex, it is challenging, time consuming, and sometimes even impossible to get 3D geometrical shapes and appearances using traditional computer graphics techniques. This difficulty can be alleviated using 3D imaging, a complementary technique to computer vision.

Computer vision produces 3D graphical models from digital images or image sequences. Because it originates from the area of Artificial Intelligence (AI), computer vision has been regarded as image understanding. It is widely believed that computer vision is much harder to carry out than computer graphics because it requires 3D objects or graphical models to be derived from 2D images or image sequences.

Recent years have seen a number of attempts to model the 3D geometry of objects and scenes from images or photographs. These attempts start with one or more 2D images and end up with a complete object representation, that is, the transformation of an image into a 3D graphical model. In this chapter, we will present an overview of the techniques involved in this transformation. We will also discuss a 3D imaging and modeling approach based on full-field fringe projection and spatial phase mapping.

7.1.1 Identification of the Issue

An image is a photo or a projection of a real-world object or scene in 3D space. Images can be simply described using gray or colored dots, which are called pixels. Graphics, however, are drawings produced using geometric primitives, such as points, lines, and circles. To obtain realistic representations, computer graphics researchers often use mathematical equations to generate complex graphics and simulate lighting conditions in 3D space. Represented in pixels without geometric features, a graphic is reduced to the status of an image, while the analysis of an image can reveal its original graphic or 3D geometric models. In this sense, the relationship of computer vision and computer graphics is inverse problem.

But these definitions of images and graphics are general. This chapter defines the image as a range image and graphics as a 3D graphical model. In a range image, every pixel corresponds to the relative height at a location of an object surface, rather than gray or colored dots. The 3D graphical model realistically represents the shape and appearance of an object.

7.1.2 Major Steps from Image to Graphics

A complete scan of an object or scene requires multiple range images from different viewpoints. This is realized by rotating the object or by combining an imaging sensor with a track device to move around the object. The 3D graphical model can then, by fusing all the range images, be placed in a common coordinate system centring in the object or scene.

The transformation of an image or image sequence into a 3D graphical model usually takes place in three steps: 1) Range image acquisition, also known as a 3D imaging technique. A range image describes the 3D relative height of an object's surface observed from one viewpoint in a local coordinate system. 2) Registration, in which range images are aligned so that they are in either a common or a world coordinate system. 3) Integration, in which a complete and non-redundant 3D graphical model is constructed by merging the overlapped data formed in registration.

7.1.3 Organization of this Chapter

The remainder of this chapter is organized as follows. Section 7.2 presents an overview of a range image acquisition and modeling methodology that involves three different but correlated techniques: range image acquisition, range image registration, and range image integration. Section 7.3 describes in detail a 'snapshot' 3D imaging and modeling system based on full-field fringe projection and spatial phase mapping. We also present a novel method for extracting the colour texture information and the geometric shape from one frame-modulated fringe image. Further, we present a method for roughly registering partial-view range images using all possible information, for example, shape and texture, and then using an ICP algorithm to improve the accuracy of the final registration. In the third step, range image integration, we present a method for fusing multi-view range images that is based on the Primary-Stitched-Line (PSL) and Secondary-Stitched-Line (SSL) concepts. Section 7.4 presents some experimental results to validate our system while in Section 7.5 we summarize the main points involved in image-based modeling and point out some open problems for future work.

7.2. Overviews

The creation of 3D graphical models of complex objects or environments is becoming an increasingly important topic in the field of computer vision as well as in industrial applications [1]. Two major forces are driving this development: the rapid development of microelectronics leading to the greater availability of lower cost imaging devices such as digital cameras, and of processors and light sources such as computers, and lasers. A second impetus is the huge potential market for 3D products, reverse engineering and computer animation being typical examples.

The following subsections present a step-by-step overview of the state-of-the-art of range acquisition techniques, following the procedure from range image to geometric model.

7.2.1 Range Image Acquisition

A range image offers a direct way to produce a 2.5D shape description of an object surface or of an environment. To the present, various types of range image acquisition techniques have been developed. According to the measurement principle, range image acquisition techniques can be put into two broad categories: passive methods and active methods. Passive methods do not interact with the object, whereas active methods do, making contact with the object or projecting some kind of energy onto the object's surface. A well-known passive approach is stereovision. Stereovision effects can be achieved in two ways: by moving an optical sensor to a known relative position in the scene or by using two or more optical sensors previously fixed in known positions [2]. In order to obtain 3D coordinates of a given point from n given projections (one from each sensor), one needs to determine the same point among these projections, but because of a lack of features or because of measurement errors it is usually difficult to determine the point-correspondence. This so-called point-correspondence problem imposes severe limitations on stereovision in practical applications. Other passive approaches include shape-from-shading in single images [3], and optical flow methods in video streams [4]. Although they require little special-purpose hardware, passive approaches do not yield the highly dense and accurate range images that many applications require, while active methods require more complicated structure designs, but offer dense and accurate range data (range images).

Active 3D imaging methods can be classified into two groups: optical and non-optical (Figure 7.1) however, because optical sensors have the advantages of non-contact operation mode and damage-free data acquisition, we here introduce only the optical approaches. A more comprehensive coverage of the optical methods can be found in Ref [5].

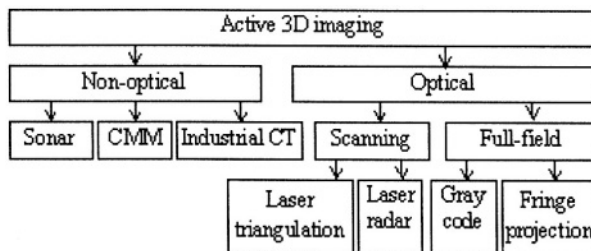


Figure 7.1. Taxonomy of active 3D imaging methods.

Optical approaches can be further divided into scanning techniques and full-field imaging techniques. Scanning techniques are represented by laser

triangulation and laser radars. Laser triangulation uses either a laser point or a strip to illuminate the surface and obtain the 3D contour of the object based on the triangulation principle [6], as illustrated in Figure 7.2. The laser radar technique is based on the measurement of the travel time or phase of either a pulsed or modulated laser [7,8]. Lange et al. [7] described a real-time range camera that could quickly sample modulated optical waves. Mengel et al. [8] presented a novel approach for direct range image acquisition based on a CMOS image sensor. All these scanning techniques required either one-dimensional or two-dimensional scanning to cover the entire surface of the object. This makes the systems more complicated and measurement more time-consuming.

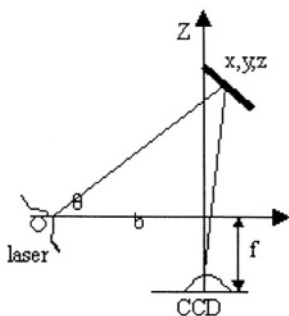


Figure 7.2. Principle of triangulation.

Full-field optical methods are becoming increasingly popular because they do not require scanning for single view acquisition, measure quickly, and can process data automatically. Full-field methods use multiple stripes or patterns rather than a single laser spot or stripe. One approach uses the sequential projection of a binary coded pattern. Multiple frames of the coded pattern image are captured to encode a pixel on the CCD with its corresponding range [9]. Fringe projection is another popular typical full-field imaging technique [10~13]. A camera captures an image of a fringe pattern projected onto a target surface. If the surface is flat, the fringe lines will appear straight. If the surface is not flat, however, any surface features will alter the appearance of the fringe pattern, which will no longer appear as a series of straight lines. Instead, each line of the fringe pattern will be curved, bent, or otherwise deformed by the shape of the surface region onto which it is projected. If the exact position, orientation, and angle of projection of the original fringe is known and each line on the surface is uniquely identifiable, then well-known triangulation formulas may be applied to the fringe pattern to calculate the profile of the surface.

A fringe pattern can be generated by projecting a grating or laser interference. In order to improve the measurement accuracy and to calibrate the system effectively, software-produced fringe patterns have been developed. One uses an LCD (Liquid Crystal Display) to generate the fringe pattern [14]. This method is simple and flexible as it creates the digital fringe pattern in the computer. However, because of the low image brightness and contrast of LCD panels, the quality of the fringe images is poor and this limits accuracy when extracting surface 3D shape information. The other method is produced by DMD (Digital Micromirror Device) [15]. Compared to the LCD, the DMD has following major advantages: high fringe brightness and contrast, high image quality and spatial repeatability. Recently, DMD has applied to acquire optically sectioned images by using the fringe-projection technique and the phase-shift method [16].

To acquire the colour texture information corresponding to the range image, several approaches have also been investigated. A detailed summary of this is provided in the Introduction of [17].

7.2.2 Registration

The final 3D graphical model representation of an object's surface is affected by two main kinds of error [18]: the acquisition error and the registration error. Acquisition error is caused by the inaccuracy of hardware such as lenses, CCD cameras, or structured lights. Limitations on the accuracy of acquisition make it even more important that the process of registration be accurate in the integration process. The accurate registration of a set of range images is of major concern in the design and implementation of a 3D graphical model. In fact, registration of the range images turns out to be the most difficult of the three steps.

The goal of registration is to find the relative position and orientation of each view with respect to other views. Existing registration algorithms can be mainly categorized as either surface matching or feature matching. Surface matching algorithms start with an approximate registration, and then iteratively refine that registration by gradually reducing the error between overlapping areas in the range images. Among existing algorithms, the iterative closest point (ICP) algorithm proposed by Besl et al [19] has been proved to be the most appropriate method for accurately registering a set of range images. ICP starts with two range images and an initial guess for their relative transform and iteratively refines the transform by repeatedly generating pairs of corresponding points on the two views and minimizing an error metric. The initial registration may be generated by a variety of methods, such as spin-image surface signatures [20], computing the principal axes of the scans [21], exhaustive searching for corresponding points [22], interactively selecting three or more matching features on overlapping pairs of range images [23], constraint searching by principal curvature [24], a frequency domain technique by using the Fourier transform [25], or user input. With respect

to the iterative procedure, many modifications have been introduced on the basic ICP concept. Chen et al [26] independently developed an approach similar to ICP which minimizes the sum of the squared distances between scene points and a local planar approximation of the model. Dorai et al [27] advanced the ICP algorithm by designing a verification mechanism to check the validity of the corresponding control points found by the new algorithm, employing only valid corresponding control point pairs to update the transformation matrix. Sharp et al [28] investigated the actual corresponding point pairs via a weighted linear combination of the positional and feature distances. Kverh et al [29] suggested a new refinement algorithm based on segmented data. This algorithm performs well in terms of accuracy and computational time, but cannot be used for objects with arbitrary free-form surfaces.

Advances in registration methods continue to be proposed [30~32]. Chung et al [30] proposed a registration algorithm to estimate the transformation parameters among multiple-range views by making use of the principal axes formed by three eigenvectors of the weighted covariance matrix of 3D coordinates of data points. Yamany et al [31] presented a novel surface-based registration method based on genetic algorithms (GA) that is able to register complex surfaces much faster than other techniques. In the process of finding closest point, the computational time is reduced by applying the grid closest point (GCP) technique. Jocinen et al [32] proposed a method for the simultaneous registration of multiple 3D profile maps without knowing correspondences.

In feature matching algorithms, correspondences between a number of features in the range images are first established and then, based on the correspondences, a transformation function is determined to register the range images. Thirion [33] and Bhanu [34] respectively used curvature extrema and planar patches to register range images. Guézic et al [35] applied characteristic ridge curve and extremal points. Feature-based algorithms rely on the existence of predefined features. If such features do not exist in the images, registration will be impossible. The accuracy of image registration by feature-matching techniques depends on the accuracy of the available features.

A method described by Higuchi, et al [36] uses a combination of surface-matching and feature-matching techniques to register range images. This method determines the curvature at each point in a range image and maps them to a unit sphere. The sphere is then partitioned into equal cells, and the array of cells is matched to another array of cells obtained from a second range image. This matching process involves rotating one sphere over the other and at each rotational step determining the total difference between the curvatures of the two images. The rotational step producing the smallest total difference is then used to determine the rotational difference between the range images.

7.2.3 Integration

Based on the accurate registration of multiple range images, different techniques have been developed for integrating or merging range images [18,37~43]. Methods for integrating range images can be classified as either structured or unstructured. Unstructured integration presumes that one has a procedure that creates a polygonal surface from an arbitrary collection of points in 3D space. A major advantage of this algorithm is the absence of prior assumptions about connectivity of points. Where there are no range images or contours to provide connectivity cues, this algorithm is the only recourse. In this case integration is completed by collecting all the range points from multiple images and presenting them to the polygonal reconstruction procedure. Hoppe et al [37] used graph traversal techniques to help construct a signed distance function from a collection of unorganized points. An isosurface extraction technique produces a polygon mesh from this distance function. Bajaj et al [38] explored alpha-shapes to construct a signed distance function to which they fit implicit polynomials. Although unorganized points algorithms are widely applicable, they discard useful information such as surface normal and reliability estimates. As a result, these algorithms are well behaved in smooth regions of surfaces, but they are not always robust in regions of high curvature or in the presence of systematic range distortions and outliers.

Structured integration methods make use of information about how each point in a range image is obtained, such as using error bounds on a point's position or adjacency information between points within one range image. Rutishauser et al [39] use a highly local approach and an explicit sensor error model to arrive at a surface description with explicit boundary information. A key feature of Rutishauser's algorithm is the ability to update its result by an additional new image. Thus, it is possible to gradually improve the surface description in regions of high noise. The drawback is that authors only considered two independent range images. Soucy et al [18] integrate a surface model using a set of triangulations modeling each canonical subset of the Venn diagram of the set of range images. Each canonic subset represents the overlap between a subset of the 2.5D range images. The connection of these local models by constrained Delaunay triangulation yields a non-redundant surface triangulation describing all surface elements sampled by the set of range images. Turk, et al [40] provide a method that zippers together adjacent meshes to form a continuous surface that correctly captured the topology of the object by accurately aligning the meshes. In this method, authors first integrate the multiple range images as whole description of the object then perform geometry averaging. These structured algorithms typically perform better than unorganized point algorithms, but in areas of high curvature they can still fail catastrophically.

Several new algorithms have been proposed for integrating structured data to generate implicit functions. Hilton, et al [41] have introduced an algorithm that constructs a single continuous implicit surface representation that is the zero-set of a scalar field function. Object models are reconstructed from both multiple view 2.5D range images and hand-held sensor range data. The authors improve geometric fusion algorithm and make it capable of reconstructing 3D object models from noisy hand-held sensor range data. The implicit surface representation allows reconstruction of unknown objects of arbitrary topology and geometry. Correct integration of overlapping surface measurements in the presence of noise is achieved using geometric constraints based on measurement uncertainty. Fusion of overlapping measurements is performed using operations in 3D space only. This avoids the local 2D projection which were required for many previous methods but which resulted in limitations on the object surface geometry. Curless et al [42] have presented a volumetric method for integrating range images using a discrete implicit surface representation. This algorithm possesses the following properties: incremental updating, representation of directional uncertainty, the ability to fill gaps in the reconstruction, and robustness in the presence of outliers. A run-length encoded volumetric data structure is used to achieve computationally efficient fusion and reduce storage costs. This in turn allows the acquisition and integration of a large number of range images. However, the discrete implicit surface representation does not enable reliable geometric fusion for complex geometry such as regions of high curvature or thin surface sections. Masuda et al [43] proposed a method in which integration and registration are alternately iterated based on the signed distance field (SDF) of an object's surface. The surface shapes are first integrated by averaging the data SDF with an assumption that they are roughly pre-registered. Then, each shape is registered to the integrated shape by estimating the optimal transformation. Integration and registration are alternately iterated until the input shapes are properly registered to the integrated shape. Weighting values are controlled to reject outliers derived from measurement errors and wrong correspondences. The proposed method does not suffer from cumulative registration errors because all data are registered to the integrated shape.

7.3. An Example System Based on Fringe Projection

Instead of using laser scanning, this example imaging system utilizes full-field fringe projection and spatial phase mapping technology using a white light source. It is able to provide "snapshot" acquisition from a free-form object surface of original fringe images along with parallel extraction of the geometric shape and colour texture. Briefly, the system operates as follows. Three-dimensional topographic information of the free-form surfaces is optically encoded in the phase

distribution of a deformed projected fringe pattern. The Fourier transformation method is used to demodulate the phase information. By selecting the first-order spatial spectrum one can retrieve the phase and colour texture information. A procedure using phase-to-depth conversion is used to recover the object's height. Figure 7.3 summarizes the whole process of digital demodulation of the encoded fringe patterns.

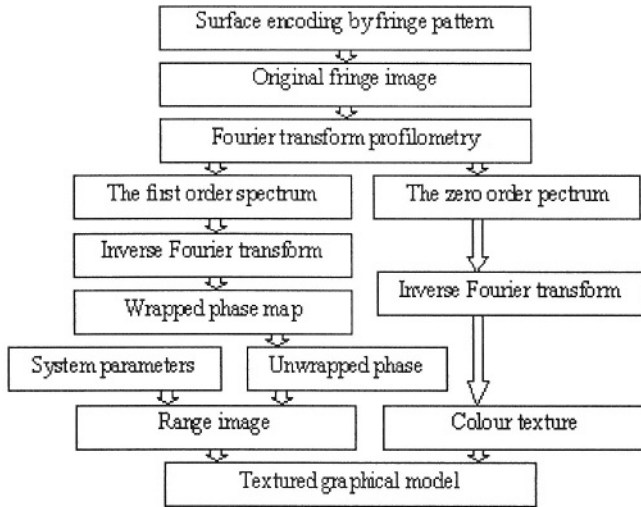


Figure 7.3. Schematic diagram of 3D imaging system.

A schematic diagram of 3D imaging system is shown in Figure 7.4. The configuration of the system is composed of an imaging unit and a projecting unit. We construct the imaging unit using a colour CCD (charge-coupled device) camera with a zoom lens and a colour frame grabber. When captured by the CCD camera and digitized by the frame grabber, the original fringe image can be saved in a hard disk. The projecting unit produces the sinusoidal fringe pattern. Figure 7.5 shows a portable 3D optical digitizer developed in our laboratory.

7.3.1 Imaging Principle

The surface encoding process is accomplished by projecting a sinusoidal fringe pattern onto the test object's surface. The intensity distribution of the deformed fringe pattern observed through the camera can be expressed as:

$$I(x, y) = a(x, y) + b(x, y) \cos \left[\frac{2\pi}{P_0} x + \Delta\phi(x, y) \right], \quad (7.1)$$

where $\Delta\phi(x, y)$ is a phase term corresponding to the object's surface height; $a(x, y)$ and $b(x, y)$ respectively representing the background intensity (denoting the colour texture information) and the fringes contrast.

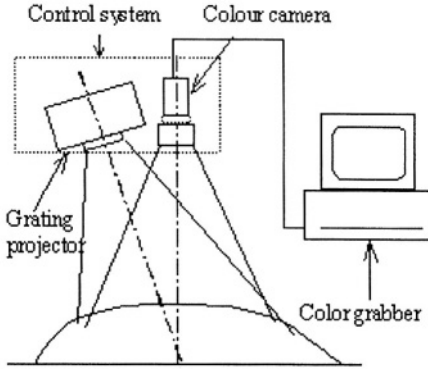


Figure 7.4. Schematic diagram of setup.

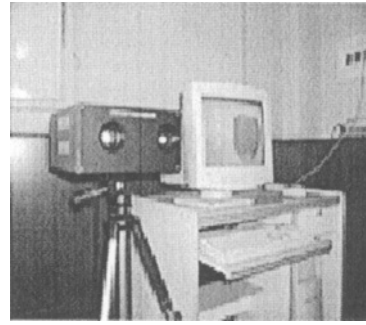


Figure 7.5. Optical digitizer.

More details about the basic principle of this kind of optical configuration for measuring the shape of 3D objects' surface can be found elsewhere, for example [44]. Here we give a brief description how to obtain the colour texture information, with more details in [17].

When a modulated fringe pattern is captured by the colour CCD camera and digitized by the colour frame grabber, the intensity image, $I(x, y)$ in Eq. (7.1), is divided into three basic-colour components: $I_r(x, y)$, $I_g(x, y)$ and $I_b(x, y)$, which denote red, green and blue, respectively. We take 2D Fourier transform of the three components and obtain the spatial spectrum of the fringe pattern. The zero-order spectrum represents the texture information of the object's surface. Using the low pass filter, we can isolate the zero-order spectrum. Taking the inverse Fourier transform of the zero-order spectrum, we can obtain the texture information of the object's surface from the deformed original fringe image.

Taking into account the three parts of the original fringe image $I_r(x, y)$, $I_g(x, y)$ and $I_b(x, y)$ according to the method mentioned above, we can obtain their texture components corresponding to red, green and blue, i.e., $T_r(x, y)$, $T_g(x, y)$, and $T_b(x, y)$. In a colour texture image, each pixel is

composed of three components: red, green and blue which can be defined by combining the corresponding position elements in $T_r(x, y)$, $T_g(x, y)$, and $T_b(x, y)$. Then, the colour texture image is extracted from the original modulated fringe image.

Because in our proposed method the colour texture and the range image are captured from exactly the same viewpoint, the corresponding parameters are known in advance. Each triangle in the range image is associated with the texture corresponding to the texture image. Each triangle is projected onto its 2D reference frame and the 2D coordinates of each point are found in the 2D texture. Consequently, it is possible to achieve a textured 3D range image.

7.3.2 Visualized Registration

As mentioned before, to integrate multiple range images from different views, we must know the transformation parameters of those views, including rotation and translation parameters. Our registration algorithm belongs to a class of surface matching.

7.3.2.1 Initial Registration

As the ICP algorithm performs a local minimization, a good initial estimate is important to ensure reliable results. Here, we explore a new method that uses a visual approach to obtain the initial registration. Rather than interactively selecting three or more matching features on overlapping pairs of range images [23], we use all of the information, including the mesh, shading and texture (even colour texture information) corresponding to the range image, to interactively adjust the relative position. The transformation parameters (three rotations and three translations) are displayed immediately. The details of operation rule and proof are represented in [45].

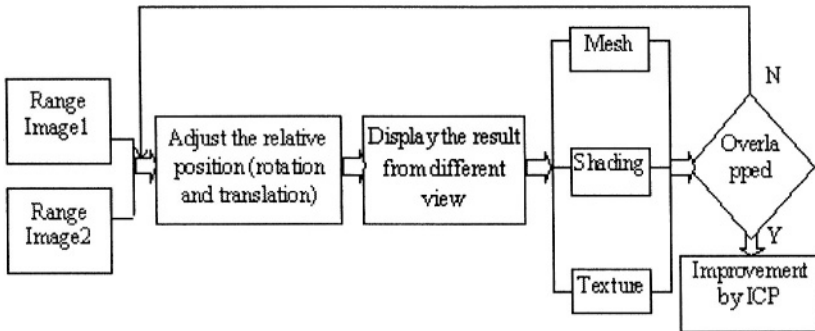


Figure 7.6. Flow chart of visualized adjustment.

Keeping one range image still, we interactively adjust the other range image using a mouse or a keyboard. While the corresponding points or features in the two range images overlap on the screen, the relative position is fixed. Then, the two range images are rotated and translated altogether. After the corresponding segments are always overlapped from different views, the initial registration parameters are obtained. When there are many sample points in the range images, it is difficult to distinguish the corresponding points. However, the texture image can be used to highlight the corresponding segments. Due to the large amount of information in the texture image, it is easy to retrieve the initial transformation parameters. The whole procedure is shown in Figure 7.6.

7.3.2.2 Improved Registration

These parameters are composed of initial entries of the ICP method transformation matrix. An improved registration procedure is described with pseudo-code shown in Figure 7.7, where T is a criteria characterized by precision threshold (here T 's value is 0.0001). The ME denotes the mean square error of the corresponding-point-pairs in the two range images. The parameter n ensures that the loop will finish even if the desired precision bound T is not reached. N is the maximum number of ICP iterations.

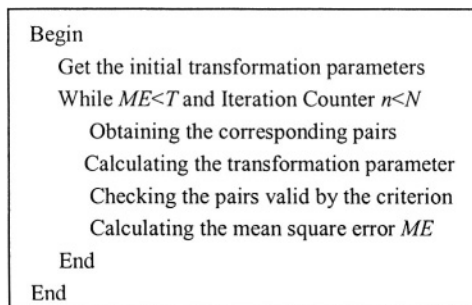


Figure 7.7. Skeleton of the variation of ICP.

7.3.3 Mesh-based Integration

After obtaining registration parameters, the range images are fused to produce a seamless 3D graphical model. In this paper, we present a new integration technique for multiple range images based on the concept of Primary-Stitched-Line (PSL) and Secondary-Stitched-Line (SSL). Before the integration, it is necessary to pre-process of the range images in two stages: (1) delete invalid data points and (2) impose a weight on each valid data point according to the distance

from the centre of the range image. Range image integration takes place in four steps: (1) Definitions: PSL and SSL; (2) Constructing PSL; (3) Constructing SSL; (4) Triangulating the Points between PSL and SSL. More details can be found in Ref. [46].

After range image integration, we get a non-redundant geometric fusion of a pair of range images. Where there are more than two range images, we add additional range image to the fused range image in order to form a new pair of range images. This process is repeated till all the range images are merged completely. Finally, we get a single and non-redundant graphical model in 3D space.

7.4. Experimental Results

Figure 7.8 shows the results of experiments using a plaster model. (a) is original fringe image; (b) shows the phase map proportional to the shape; (c) represents extracted texture; (d)~(e) describe the range data by mesh, shading, and colour textured with a certain pitching and rotation angle.

Figure 7.9 indicates the results for merging two range images, where (a)~(b) show two range images from two different views; (c)~(d) represent PSL and SSL; and (e)~(f) give the results before and after locally triangulating the points between PSL and SSL.

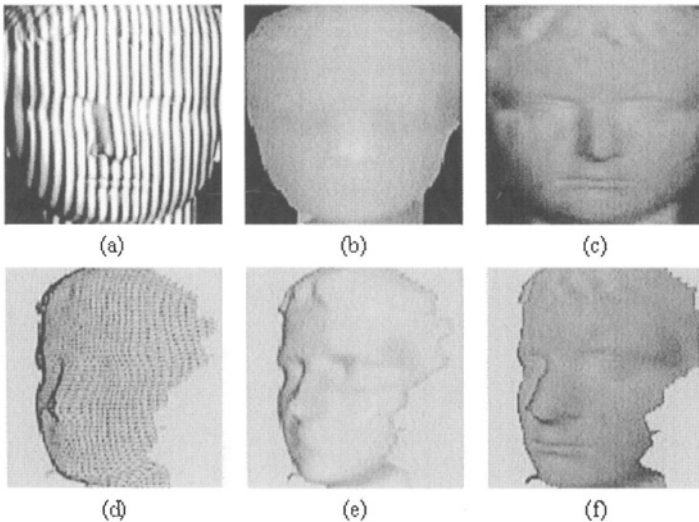


Figure 7.8. Extraction range and texture and their display.

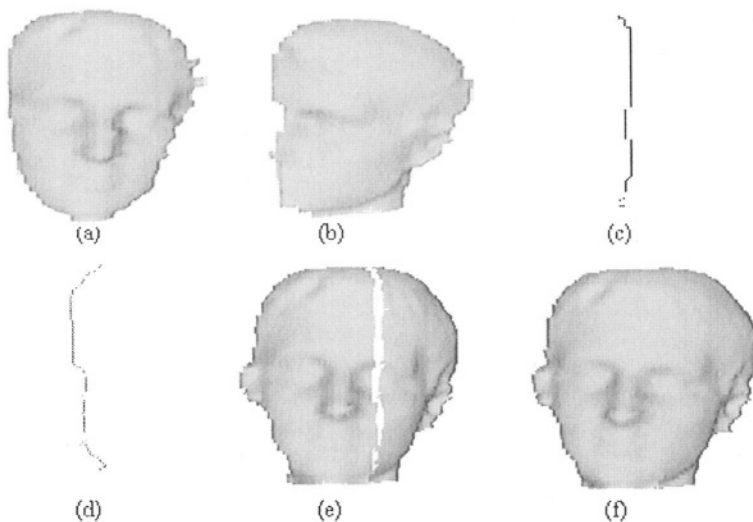


Figure 7.9. Procedure of range image integration.

Figure 7.10 represents two complete geometric models after performing the registration and integration of eight range images. The numbers of points in the merged model are 27,580 and 34,600; but the numbers of triangles in the merged model are 47,852 and 45,032. The total computational times are 87 and 139 seconds, respectively. The experimental results show that it is possible to quickly and efficiently build up a 3D graphical model of a real world object using an introduced system from multiple range images.

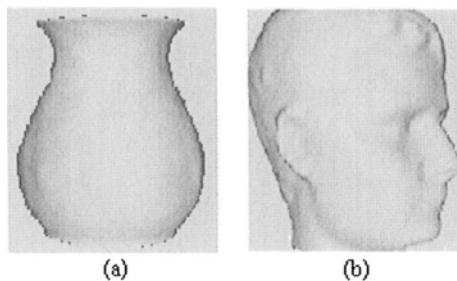


Figure 7.10. 3D graphical model from multi range images.

7.5. Conclusion Remarks and Future Work

In this chapter, we have briefly reviewed some approaches to the transformation of images into 3D graphical models and have introduced a 'snapshot' three-dimensional imaging and modeling (3DIM) system based on full-field fringe projection and phase mapping technology. We have described how to obtain a 3D graphical model of an object from multiple range images and how to get colour texture by using an introduced 3DIM system. Since 1997, the International Conference on 3D imaging and modelling has been held in Canada every two years [47]. Nowadays, the technology of 3DIM has been widely applied in many industrial areas. Blais recently, provided an overview of the development of commercial 3D imaging systems in the past 20 years [48].

We have come to the following conclusions:

- Various approaches and techniques involved in the transformation procedure have been discussed, including acquisition, registration, and integration of range images.
- Optical full-field imaging technique is becoming a more popular way to acquire the range image of an object surface, along with the colour texture information.
- A practical 3DIM system is presented to explain the three transformation steps in detail. The colour texture and the 3D geometrical shape can be extracted from the same fringe image. This eliminates alignment problems and the need for extra cameras.

Future work will include the development of a global optimization technique for multiple range image registration and a more efficient range image integration algorithm. Further research will also be undertaken into the fusing of the texture images corresponding to range images [49]. Another area of focus will be the robust reconstruction of 3D graphical models sufficiently accurate to deal with large classes of objects that are moving and deforming over time [50].

Acknowledgments

The financial support from National Science Foundation of China under the contract numbers 69875013 and 60275012 are gratefully acknowledged.

References

- [1] M. Petrov, A. Talapov and T. Robertson. "Optical 3D digitizers: bringing life to the virtual world". *IEEE Computer Graphics and Application*, vol. 18, no. 5-6, pp. 28-37, 1998.
- [2] N. Ayache and F. Lustman. "Trinocular stereovision for robotics". *IEEE Trans. on PAMI*, vol. 13, no. 1, pp. 73-84, 1991.
- [3] R. Zhang, P. Tsai, J. Cryer and M. Shah. "Shape from shading: a survey". *IEEE Trans. on PAMI*, vol. 21, no. 8, pp. 690-706, 1999.
- [4] Y. Xiong and S. Shafer. "Dense structure from a dense optical flow sequence". *Computer Vision and Image Understanding*, vol. 69, no. 2, pp. 222-245, 1998.
- [5] F. Chen, G. M. Brown and M. Song. "Overview of three-dimensional shape measurement using optical methods". *Optical. Eng.*, vol. 39, no. 1, pp. 10-22, 2000.
- [6] P. Saint-Marc, J. L. Jezouin and G. Medioni. "A versatile PC-based range finding system". *IEEE Transaction on Robotics and Automation*, vol. 7, no. 2, pp. 250-256, 1991.
- [7] R. Lange and P. Seitz. "Solid-state time-of-flight range camera". *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390-397, 2001.
- [8] P. Mengel, G. Doemens and L. List. "Fast range imaging by CMOS sensor array through multiple double short time integration (MDSI)". *International Conference on Image Processing*, vol. 2, pp. 169-172, 2001.
- [9] G. Sansoni, M. Carocci and R. Rodella. "3D vision based on the combination of gray code and phase shift light projection". *Appl. Opt.*, vol. 38, no. 31, pp. 6565-6573, 1999.
- [10] G. S. Spagnolo, G. Guattari and C. Sapia. "Contouring of artwork surface by fringe projection and FFT analysis". *Optics and Lasers in Engineering*, vol. 33, no. 2, pp. 141-156, 2000.
- [11] H. Zhang, M. J. Lalor and D. R. Burton. "Spatiotemporal phase unwrapping for the measurement of discontinuous objects in dynamic fringe-projection phase-shifting profilometry". *Appl. Opt.*, vol. 38, no. 16, pp. 3534-3541, 1999.
- [12] Y. Y. Hung, L. Lin and H. M. Shang. "Practical three-dimensional computer vision techniques for full-field surface measurement". *Optical. Eng.*, vol. 39, no. 1, pp. 143-149, 2000.
- [13] C. Quan, X. Y. He and C. F. Wang. "Shape measurement of small objects using LCD fringe projection with phase shifting". *Opt. Commun.*, vol. 189, no. 1-3, pp. 21-29, 2001.
- [14] C. Quan, C. Tay, X. Kang, X. He and H. Shang. "Shape measurement by use of liquid-crystal display fringe projection with two-step phase shifting". *Applied Optics*, vol. 42, no. 13, pp. 2329-2335, 2003.
- [15] F. Bitte, G. Dussler and T. Pfeifer. "3D micro-inspection goes DMD". *Optics and Lasers in Engineering*, vol. 36, no. 2, pp. 155-167, 2001.
- [16] T. Fukano and A. Miyawaki. "Whole-field fluorescence microscope with digital micromirror device: imaging of biological samples". *Applied Optics*, vol. 42, no. 19, pp. 4119-4124, 2003.

- [17] Z. H. Zhang, David Zhang, X. Peng and X. T. Hu. "Color texture extraction from fringe image based on full-field projection". *Optical. Eng.*, vol. 42, no. 7, pp. 1935-1939, 2003.
- [18] M. Soucy and D. Laurendeau. "A general surface approach to the integration of a set of range views". *IEEE Trans. on PAMI*, vol. 17, no. 4, pp.344-358, 1995.
- [19] P. J. Besl and N. D. McKay. "A method for registration of 3-D shapes". *IEEE Trans. on PAMI*, vol. 14, no. 2, pp. 239-256, 1992.
- [20] E. Johnson and M. Hebert. "Surface registration and integration of textured 3D data". *Image and Vision Computing*, vol. 17, no. 2, pp. 135-147, 1999.
- [21] C. Dorai, G. Wang and A. K. Jain. "Optimal registration of object views using range data". *IEEE Trans. on PAMI*, vol. 19, no. 10, pp. 1131-1138, 1997.
- [22] S. Chen, Y. P. Hung and J. B. Cheng. "RANSAC-based DARCES: a new approach to fast automatic registration of partially overlapping range images". *IEEE Trans. on PAMI*, vol.21, no. 11, pp. 1229-1234, 1999.
- [23] F. Bernardini, H. Rushmeier and I. Martin. "Building a digital model of michelangelo's florentine pieta". *IEEE Computer Graphics & Application*, vol. 2, no. 1, pp. 59-67, 2002
- [24] C. Chua and R. Jarvis. "3D free-form surface registration and object recognition". *International Journal Computer Vision*, vol. 17, no. 1, pp. 77-99, 1996.
- [25] L. Lucchese, G. Doretto, and G.M. Cortelazzo. "A frequency domain technique for 3D view registration". *IEEE Trans. on PAMI*, vol. 24, no. 11, pp. 1468-1484, 2002.
- [26] Y. Chen and G. Medioni. "Objects modelling by registration of multiple range images". *Image and Vision Computing*, vol. 10, no. 3, pp. 145-155, 1992.
- [27] C. Dorai, G. Wang and A. K. Jain. "Registration and integration of multiple object views for 3D model construction". *IEEE Trans. on PAMI*, vol. 20, no. 1, pp. 83-89, 1998.
- [28] G. C. Sharp, S. W. Lee and D. K. Wehe. "ICP registration using invariant features". *IEEE Trans. on PAMI*, vol. 24, no. 1, pp. 90-102, 2002.
- [29] B. Kverh, and A. L. Ljubljana. "A new refinement method for registration of range images based on segmented data". *Computing*, vol. 68, no. 1, pp. 81-96, 2002.
- [30] D. H. Chung, I. D. Yun and S. U. Lee. "Registration of multiple-range views using the reverse-calibration technique". *Pattern Recognition*, vol. 31, no. 4, pp. 457-464, 1998.
- [31] S. M. Yamany, M. N. Ahmed and A. A. Farag. "A new genetic-based technique for matching 3-D curves and surfaces". *Pattern Recognition*, vol. 32, no. 11, pp. 1817-1820, 1999.
- [32] O. Jocinen. "Area-based matching for simultaneous registration of multiple 3-D profile maps". *Computer Vision and Image Understanding*, vol. 71, no. 3, pp. 431-447, 1998.
- [33] J. Thirion. "New feature points based on geometric invariants for 3D image registration". *International Journal of Computer Vision*, vol. 18, no. 2, pp. 121-137, 1996.
- [34] B. Bhanu. "Representation and shape matching of 3D objects". *IEEE Trans. on PAMI*, vol. 6, no. 3, pp. 145-157, 1984.

- [35] A. Guézic, X. Pennec, and N. Ayache. "Medical image registration using geometric hashing". *IEEE Computational Science & Engineering*, vol. 4, no. 4, pp. 29-41, 1997.
- [36] K. Higuchi, M. Hebert and K. Ikeuchi. "Building 3-D models from unregistered range images". *Graphical Models and Image Processing*, vol. 57, no.4, pp. 315-333, 1995.
- [37] H. Hoppe, T. Deroose and T. Duchamp. "Surface reconstruction from unorganized points". *Computer Graphics*, vol. 26, no. 2, pp. 71-78, 1992.
- [38] C. L. Bajaj, F. Bernardini and G. Xu. "Automatic reconstruction of surfaces and scalar fields from 3D scans". *Proc. SIGGRAPH*, pp. 109-118, 1995. Los Angeles, CA.
- [39] M. Rutishauser, M. Stricker and M. Trobina. "Merging range images of arbitrarily shaped objects". *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 573-580, 1994.
- [40] G. Turk and M. Levoy. "Zippered polygon meshes from range images". *Proc. SIGGRAPH*, pp. 311-318, 1994. Orlando, Florida.
- [41] A. Hilton, A. J. Stoddart and J. Illingworth. "Implicit surface-based geometric fusion". *Computer Vision and Image Understanding*, vol. 69, no. 3, pp. 273-291, 1998.
- [42] B. Curless and M. Levoy. "A volumetric method for building complex models from range images". *Proc. SIGGRAPH*, pp. 303-312, 1996.
- [43] T. Masuda. "Registration and integration of multiple range images by matching signed distance fields for object shape modeling". *Computer Vision and Image Understanding*, vol. 87, no. 1-3, pp. 51-65, 2002.
- [44] H. J. Tiziani. "Optical methods for precision measurements". *Opt. Quantum Electron.*, vol. 21, pp. 253-281, 1989.
- [45] X. Peng, Z. H. Zhang and H. J. Tiziani. "3-D imaging and modeling—Part I: acquisition and registration". *Optik*, vol. 113, no. 10, pp. 448-452, 2002.
- [46] X. Peng, Z. H. Zhang and H. J. Tiziani. "3-D imaging and modeling—Part II: integration and duplication". *Optik*, vol. 113, no. 10, pp. 453-458, 2002.
- [47] Alberta, Canada, <http://www.3dimconference.org>.
- [48] F. Blais. "A review of 20 years of range sensors development". *Proceedings of SPIE-IS&T Electronic Imaging, SPIE*, vol. 5013, pp. 62-76, Jan. 2003. California, USA.
- [49] A. Agathos and R. Fisher. "Colour texture fusion of multiple range images". *The 4th International Conference on 3-D Digital Imaging and Modelling*, Oct. 2003, Canada.
- [50] H. Yang, T. Han and J. Lee. "Reconstruction of dynamic 3D scene based on view hull and view morphing". *The Journal of Visualization and Computer Animation*, vol. 14, no. 3, pp. 139-148, 2003.

This page intentionally left blank

Chapter 8

AN INTRODUCTION TO IMAGE-BASED RENDERING

Heung Yeung Shum ¹, Yin Li ² and Sing Bing Kang ¹

1. *Microsoft Research Asia, Beijing, P.R. China*

2. *Hong Kong University of Science and Technology*

3. *Microsoft Research, Redmond, WA 98052*

{hshum,sbkang}@microsoft.com, liyin@ust.hk

Abstract

In this chapter, we review the techniques for image-based rendering. Unlike traditional 3D computer graphics in which 3D geometry of the scene is known, image-based rendering (IBR) techniques render novel views directly from input images. IBR techniques can be classified into three categories according to how much geometric information is used: rendering without geometry, rendering with implicit geometry (i.e., correspondence), and rendering with explicit geometry (either with approximate or accurate geometry). We discuss the characteristics of these categories and their representative techniques.

IBR techniques demonstrate a surprising diverse range in their extent of use of images and geometry in representing 3D scenes. We explore the issues in trading off the use of images and geometry by revisiting plenoptic sampling analysis and the notions of view dependency and geometric proxies. Finally, we highlight a practical IBR technique called *pop-up light field*. It models a sparse light field using a set of *coherent layers*, which incorporates both color and matting information, and renders in real time and free of aliasing.

Keywords: Image-based modeling, image-based rendering, image-based representations

8.1. Introduction

Image-based modeling and rendering techniques have received a lot of attention as a powerful alternative to traditional geometry-based techniques for image synthesis. These techniques use images rather than geometry as primitives for rendering novel views. Previous surveys related to image-based rendering (IBR) have suggested characterizing a technique based on how image-centric

or geometry-centric it is. This has resulted in the image-geometry continuum of image-based representations [35, 26].

For didactic purposes, we classify the various rendering techniques (and their associated representations) into three categories, namely rendering with no geometry, rendering with implicit geometry, and rendering with explicit geometry. These categories, depicted in Figure 8.1, should actually be viewed as a continuum rather than absolute discrete ones, since there are techniques that defy strict categorization.

At one end of the rendering spectrum, traditional texture mapping relies on very accurate geometric models but only a few images. In an image-based rendering system with depth maps, such as 3D warping [40], and layered-depth images (LDI) [55], LDI tree [11], etc., the model consists of a set of images of a scene and their associated depth maps. The surface light field [63] is another geometry-based IBR representation which uses images and Cyberware scanned range data. When depth is available for every point in an image, the image can be rendered from any nearby point of view by projecting the pixels of the image to their proper 3D locations and re-projecting them onto a new picture. For many synthetic environments or objects, depth is available. However, obtaining depth information from real images is hard even with state-of-art vision algorithms.

Some image-based rendering systems do not require explicit geometric models. Rather, they require feature correspondence between images. For example, view interpolation techniques [12] generate novel views by interpolating optical flow between corresponding points. On the other hand, view morphing [54] results in-between camera matrices along the line of two original camera centers, based on point correspondences. Computer vision techniques are usually used to generate such correspondences.

At the other extreme, light field rendering uses many images but does not require any geometric information or correspondence. Light field rendering [37] produces a new image of a scene by appropriately filtering and interpolating a pre-acquired set of samples. The Lumigraph [20] is similar to light field rendering but it uses approximated geometry to compensate for non-uniform sampling in order to improve rendering performance. Unlike light field and Lumigraph where cameras are placed on a two-dimensional grid, the Concentric Mosaics representation [57] reduces the amount of data by capturing a sequence of images along a circle path. In addition, it uses a very primitive form of a geometric impostor, whose radial distance is a function of the panning angle. (A geometric impostor is basically a 3D shape used in IBR techniques to improve appearance prediction by depth correction. It is also known as geometric proxy.)

Because light field rendering does not rely on any geometric impostors, it has a tendency to rely on oversampling to counter undesirable aliasing effects in output display. Oversampling means more intensive data acquisition, more storage, and higher redundancy.

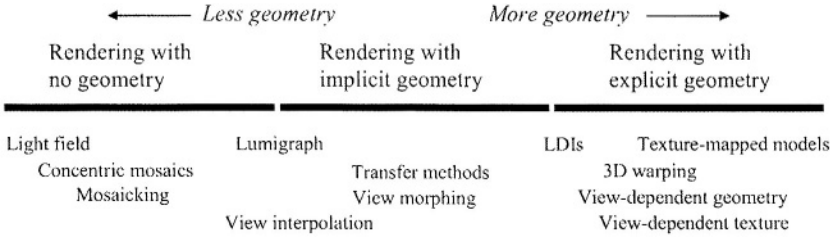


Figure 8.1. Categories used in this chapter, with representative members.

What is the minimum number of images necessary to enable anti-aliased rendering? This fundamental issue needs to be addressed so as to avoid undersampling or unnecessary sampling. Sampling analysis in image-based rendering, however, is a difficult problem because it involves unraveling the relationship among three elements: the depth and texture information of the scene, the number of sample images, and the rendering resolution. Chai *et al.* showed in their plenoptic sampling analysis [9] that the minimum sampling rate is determined by the depth variation of the scene. In addition, they showed that there is a trade-off between the number of sample images and the amount of geometry (in the form of per-pixel depth) for anti-aliased rendering.

The remainder of this paper is organized as follows. Three categories of image-based rendering systems, with no, implicit, and explicit geometric information, are respectively presented in Sections 8.2, 8.3, and 8.4. The trade-offs between the use of geometry and images for IBR are weighted in Section 8.5. A layered representation is discussed in detail in section 8.6. We also discuss compact representation and efficient rendering techniques in Section 8.7, and provide concluding remarks in Section 8.7.3.

8.2. Rendering with No Geometry

In this section, we describe representative techniques for rendering with unknown scene geometry. These techniques rely on the characterization of the plenoptic function.

8.2.1 Plenoptic Modeling

The original 7D plenoptic function [1] is defined as the intensity of light rays passing through the camera center at every 3D location (V_x, V_y, V_z) at every possible angle (θ, ϕ) , for every wavelength λ , at every time t , i.e.,

$$P_7 = P(V_x, V_y, V_z, \theta, \phi, \lambda, t). \quad (8.1)$$

Table 8.1. A taxonomy of plenoptic functions.

Dim.	Year	View space	Name
7	1991	free	Plenoptic function
5	1995	free	Plenoptic modeling
4	1996	bounding box	Lightfield/ Lumigraph
3	1999	bounding circle	Concentric Mosaics
2	1994	fixed point	Cylindrical/Spherical panorama

Adelson and Bergen [1] considered one of the tasks of early vision as extracting a compact and useful description of the plenoptic function's local properties (e.g., low order derivatives). It has also been shown by Wong *et al.* [62] that light source directions can be incorporated into the plenoptic function for illumination control. By removing two variables, time t (therefore static environment) and light wavelength λ , McMillan and Bishop [44] introduced the notion of plenoptic modeling with the 5D complete plenoptic function,

$$P_5 = P(V_x, V_y, V_z, \theta, \phi). \quad (8.2)$$

The simplest plenoptic function is a 2D panorama (cylindrical [13] or spherical [61]) when the viewpoint is fixed,

$$P_2 = P(\theta, \phi). \quad (8.3)$$

A regular rectilinear image with a limited field of view can be regarded as an incomplete plenoptic sample at a fixed viewpoint.

Image-based rendering, or IBR, can be viewed as a set of techniques to reconstruct a continuous representation of the plenoptic function from observed discrete samples. The issues of sampling the plenoptic function and reconstructing a continuous function from discrete samples are important research topics in IBR. As a preview, a taxonomy of plenoptic functions is shown in Table 8.1.

The cylindrical panoramas used in [44] are two-dimensional samples of the plenoptic function in two viewing directions. The two viewing directions for each panorama are panning and tilting about its center. This restriction can be relaxed if geometric information about the scene is known. In [44], stereo techniques are applied on multiple cylindrical panoramas in order to extract disparity (or inverse depth) distributions. These distributions can then be used to predict appearance (i.e., plenoptic function) at arbitrary locations. Similar work on regular stereo pairs can be found in [33], where correspondences constrained along epipolar geometry are directly used for view transfer.

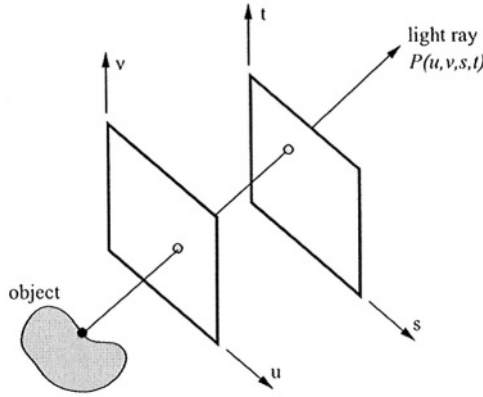


Figure 8.2. Representation of a light field.

8.2.2 Light Field and Lumigraph

It was observed in both light field rendering [37] and Lumigraph [20] systems that as long as we stay outside the convex hull (or simply a bounding box) of an object. The reverse is also true if camera views are restricted inside a convex hull. we can simplify the 5D complete plenoptic function to a 4D light field plenoptic function,

$$P_4 = P(u, v, s, t), \quad (8.4)$$

where (u, v) and (s, t) are parameters of two planes of the bounding box, as shown in Figure 8.2. Note that these two planes need not be parallel. There is also an implicit and important assumption that the strength of a light ray does not change along its path. For a complete description of the plenoptic function for the bounding box, six sets of such two-planes would be needed. More restricted versions of Lumigraph have also been developed by Sloan *et al.* [59] and Katayama *et al* [31]. Here, the camera motion is restricted to a straight line.

The principles of light field rendering and Lumigraph are the same, except that the Lumigraph has the additional (approximate) object geometry for better compression and appearance prediction. In the light field system, a capturing rig is designed to obtain uniformly sampled images. To reduce aliasing effect, the light field is pre-filtered before rendering. A vector quantization scheme is used to reduce the amount of data used in light field rendering, while achieving random access and selective decoding. On the other hand, the Lumigraph can be constructed from a set of images taken from arbitrarily placed viewpoints. A re-binning process (in this case, resampling to a regular grid using a hierarchical interpolation scheme) is therefore required. Geometric information is used to

guide the choices of the basis functions. Because of the use of geometric information, the sampling density can be reduced. Note that we place the Lumigraph in the category of "no geometry" because it is primarily image-based, with geometry playing a secondary (optional) role.

The $P_4 = P(u, v, s, t)$ two-plane parameterization is just one of many for light fields. Other types of light fields include spherical or isotropic light fields [22, 7], sphere-plane light fields [7], and hemispherically arranged light fields with geometry [38]. The issue of uniformly sampling the light field was investigated by Camahort [6]. He introduced an isotropic parameterization he calls the direction-and-point parameterization (DPP), and showed that while no parameterization is view-independent, only the DPP introduces a single bias.

Buehler *et al.* [5] extended the light field concept through a technique that uses geometric proxies (if available), handles unstructured input, and blends textures based on relative angular position, resolution, and field-of-view. They achieve real-time rendering by interpolating the blending field using a sparse set of locations.

8.2.3 Concentric Mosaics

Obviously, the more constraints we have on the camera location (V_x, V_y, V_z), the simpler the plenoptic function becomes. If we want to capture all viewpoints, we need a complete 5D plenoptic function. As soon as we stay in a convex hull (or conversely viewing from a convex hull) free of occluders, we have a 4D light field. If we do not translate at all, we have a 2D panorama. An interesting 3D parameterization of the plenoptic function, called Concentric Mosaics (CMs) [57], was proposed by Shum and He; here, the sampling camera motion is constrained along concentric circles on a plane.

By constraining camera motion to planar concentric circles, CMs can be created by compositing slit images taken at different locations of each circle. CMs index all input image rays naturally in 3 parameters: radius, rotation angle, and vertical elevation. Novel views are rendered by combining the appropriate captured rays in an efficient manner at rendering time. Although vertical distortions exist in the rendered images, they can be alleviated by depth correction. CMs have good space and computational efficiency. Compared with a light field or Lumigraph, CMs have much smaller file size because only a 3D plenoptic function is constructed.

Most importantly, CMs are very easy to capture. Capturing CMs is as easy as capturing a traditional panorama except that CMs require more images. By simply spinning an off-centered camera on a rotary table, Shum and He [57] were able to construct CMs for a real scene in about 10 minutes. Like panoramas, CMs do not require the difficult modeling process of recovering geometric and photometric scene models. Yet CMs provide a much richer user experience

by allowing the user to move freely in a circular region and observe significant parallax and lighting changes. (Parallax refers to the apparent relative change in object location within a scene due to a change in the camera viewpoint.) The ease of capturing makes CMs very attractive for many virtual reality applications.

Rendering of a lobby scene from captured CMs is shown in Figure 8.3. A rebinned CM at the rotation center is shown in Figure 8.3 (a), while two rebinned CMs taken at exactly opposite directions are shown in Figure 8.3 (b) and (c), respectively. It has also been shown in [48] that such two mosaics taken from a single rotating camera can simulate a stereo panorama. In Figure 8.3 (d), strong parallax can be seen between the plant and the poster in the rendered images. More specifically, in the left image, the poster is partially obscured by the plant, while the poster and the plant do not visually overlap in the right image. This is a significant visual cue that the camera viewpoint has shifted.

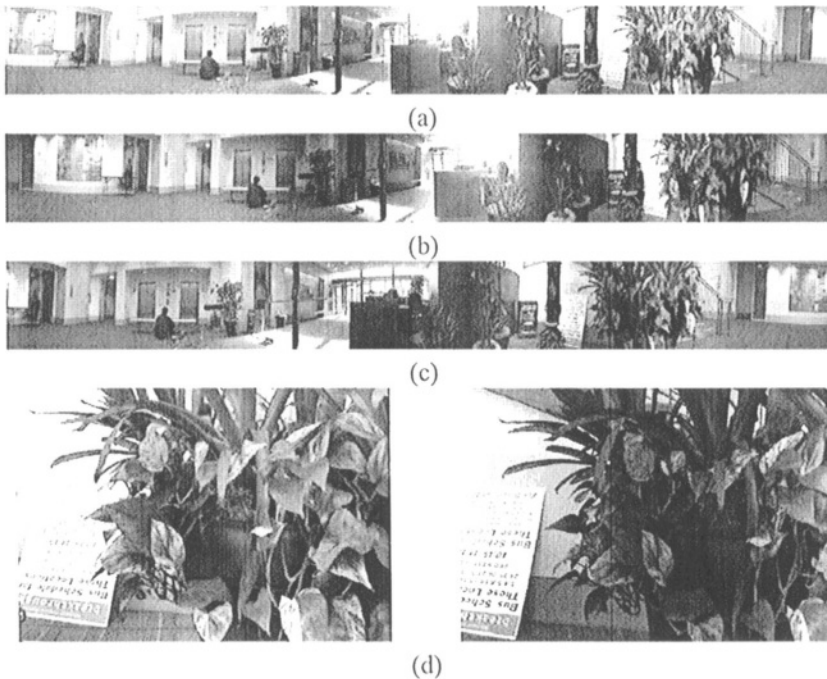


Figure 8.3. Rendering a lobby: rebinned concentric mosaic (a) at the rotation center; (b) at the outermost circle; (c) at the outermost circle but looking at the opposite direction of (b); (d) parallax change between the plant and the poster.

8.2.4 Image Mosaicing

A complete plenoptic function at a fixed viewpoint can be constructed from incomplete samples. Specifically, a panoramic mosaic is constructed by registering multiple regular images. For example, if the camera focal length is known and fixed, one can project each image to its cylindrical map and the relationship between the cylindrical images becomes a simple translation. For arbitrary camera rotation, one can first register the images by recovering the camera movement, before converting to a final cylindrical/spherical map.

Many systems have been built to construct cylindrical and spherical panoramas by stitching multiple images together, e.g., [39, 60, 13, 44, 61] among others. When the camera motion is very small, it is possible to put together only small stripes from registered images, i.e., slit images (e.g., [68, 49]), to form a large panoramic mosaic. Capturing panoramas is even easier if omnidirectional cameras (e.g., [46, 45]) or fisheye lens [64] are used.

Szeliski and Shum [61] presented a complete system for constructing *panoramic image mosaics* from sequences of images. Their mosaic representation associates a transformation matrix with each input image, rather than explicitly projecting all of the images onto a common surface, such as a cylinder. In particular, to construct a full view panorama, a *rotational mosaic* representation associates a rotation matrix (and optionally a focal length) with each input image. A *patch-based alignment* algorithm is developed to quickly align two images given motion models. Techniques for estimating and refining camera focal lengths are also presented.

In order to reduce accumulated registration errors, global alignment through block adjustment is applied to the whole sequence of images, which results in an optimally registered image mosaic. To compensate for small amounts of motion parallax introduced by translations of the camera and other unmodeled distortions, a local alignment (*deghosting*) technique [58] warps each image-based on the results of pairwise local image registrations. Combining both global and local alignment significantly improves the quality of image mosaics, thereby enabling the creation of full view panoramic mosaics with hand-held cameras.

A tessellated spherical map of the full view panorama is shown in Figure 8.4. Three panoramic image sequences of a building lobby were taken with the camera on a tripod tilted at three different angles. 22 images were taken for the middle sequence, 22 images for the upper sequence, and 10 images for the top sequence. The camera motion covers more than two thirds of the viewing sphere, including the top.

Apart from blending images to directly produce wider fields of view, one can use the multiple images to generate higher resolution panoramas as well (e.g., using maximum likelihood algorithms [23] or learnt image models [8]).

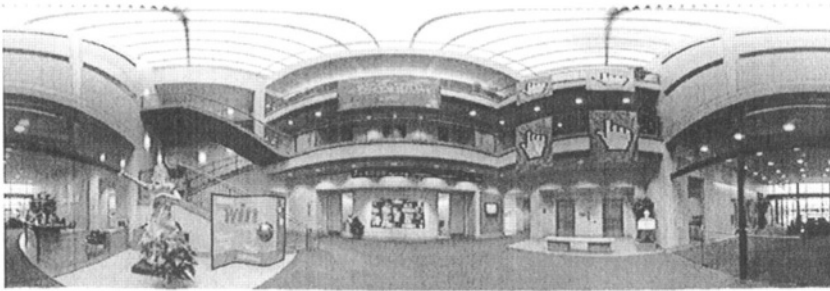


Figure 8.4. Tessellated spherical panorama covering the north pole (constructed from 54 images).

8.3. Rendering with Implicit Geometry

There is a class of techniques that relies on positional correspondences across a small number of images to render new views. This class has the term *implicit* to express the fact that geometry is not directly available; 3D information is computed only using the usual projection calculations. New views are computed based on direct manipulation of these positional correspondences, which are usually point features.

The approaches under this class are view interpolation, view morphing, and transfer methods. View interpolation uses general dense optic flow to directly generate intermediate views. The intermediate view may not necessarily be geometrically correct. View morphing is a specialized version of view interpolation, except that the interpolated views are always geometrically correct. The geometric correctness is ensured because of the linear camera motion. Transfer methods also produce geometrically correct views, except that the camera viewpoints can be arbitrarily positioned.

8.3.1 View Interpolation

Chen and Williams' view interpolation method [12] is capable of reconstructing arbitrary viewpoints given two input images and dense optical flow between them. This method works well when two input views are close by, so that visibility ambiguity does not pose a serious problem. Otherwise, flow fields have to be constrained so as to prevent foldovers. In addition, when two views are far apart, the overlapping parts of two images may become too small. Chen and Williams' approach works particularly well when all the input images share a common gaze direction, and the output images are restricted to have a gaze angle less than 90° .

Establishing flow fields for view interpolation can be difficult, in particular for real images. Computer vision techniques such as feature correspondence

or stereo must be employed. For synthetic images, flow fields can be obtained from the known depth values.

8.3.2 View Morphing

From two input images, Seitz and Dyer's view morphing technique [54] reconstructs any viewpoint on the line linking two optical centers of the original cameras. Intermediate views are exactly linear combinations of two views only if the camera motion associated with the intermediate views are perpendicular to the camera viewing direction. If the two input images are not parallel, a pre-warp stage can be employed to rectify two input images so that corresponding scan lines are parallel. Accordingly, a post-warp stage can be used to un-rectify the intermediate images. Scharstein [53] extends this framework to camera motion in a plane. He assumes, however, that the camera parameters are known.

In a more recent work, Aliaga and Carlbom [2] describe an interactive virtual walkthrough system that uses a large network of omnidirectional images taken within a 2D plane. To construct a view, the system uses the closest set of images, warps them using precomputed corresponding features, and blends the results.

8.3.3 Transfer Methods

Transfer methods (a term used within the photogrammetric community) are characterized by the use of a relatively small number of images with the application of geometric constraints (either recovered at some stage or known *a priori*) to reproject image pixels appropriately at a given virtual camera viewpoint. The geometric constraints can be of the form of known depth values at each pixel, *epipolar constraints* between pairs of images, or *trifocal/trilinear tensors* that link correspondences between triplets of images. The view interpolation and view morphing methods above are actually specific instances of transfer methods.

Laveau and Faugeras [34] use a collection of images called reference views and the principle of the fundamental matrix to produce virtual views. The new viewpoint, which is chosen by interactively choosing the positions of four control image points, is computed using a reverse mapping or raytracing process. For every pixel in the new target image, a search is performed to locate the pair of image correspondences in two reference views. The search is facilitated by using the epipolar constraints and the computed dense correspondences (also known as image disparities) between the two reference views.

Note that if the camera is only weakly calibrated, the recovered viewpoint will be that of a projective structure (see [19] for more details). This is because there is a class of 3D projections and structures that will result in exactly the same reference images. Since angles and areas are not preserved, the resulting

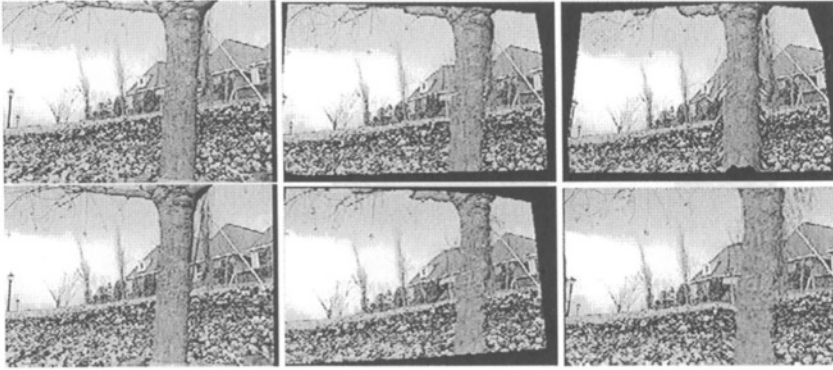


Figure 8.5. Example of visualizing using the trilinear tensor. The left-most two images are the reference images, with the rest synthesized at arbitrary viewpoints.

viewpoint may appear warped. Knowing the internal parameters of the camera removes this problem.

If a trilinear tensor, which is a $3 \times 3 \times 3$ matrix, is known for a set of three images, then given a pair of point correspondences in two of these images, a third corresponding point can be directly computed in the third image without resorting to any projection computation. This idea has been used to generate novel views from either two or three reference images [3].

The idea of generating novel views from two or three reference images is rather straightforward. First, the "reference" trilinear tensor is computed from the point correspondences between the reference images. In the case of only two reference images, one of the images is replicated and regarded as the "third" image. If the camera intrinsic parameters are known, then a new trilinear tensor can be computed from the known pose change with respect to the third camera location. The new view can subsequently be generated using the point correspondences from the first two images and the new trilinear tensor. A set of novel views created using this approach can be seen in Figure 8.5.

8.4. Rendering with Explicit Geometry

In this class of techniques, the representation has direct 3D information encoded in it, either in the form of depth along known lines-of-sight, or 3D coordinates. The more traditional 3D texture-mapped model belongs to this category (not described here, since its rendering uses the conventional graphics pipeline).

In this category, we have 3D warping, Layered Depth Image (LDI) rendering, and view-dependent texture mapping. 3D warping is applied to depth per-

pixel representations such as sprites. LDIs are extensions of depth per-pixel representations, since they can encode multiple depths along a given ray. View-dependent texture mapping refers to mapping multiple texture maps to the same 3D surface and averaging their colors based on current viewpoint relative to the sampled viewpoints.

8.4.1 3D Warping

When the depth information is available for every point in one or more images, 3D warping techniques (e.g., [43]) can be used to render nearly all viewpoints. An image can be rendered from any nearby point of view by projecting the pixels of the original image to their proper 3D locations and re-projecting them onto the new picture. The most significant problem in 3D warping is how to deal with holes generated in the warped image. Holes are due to the difference of sampling resolution between the input and output images, and the disocclusion where part of the scene is seen by the output image but not by the input images. To fill in holes, the most commonly used method is to map a pixel in the input image to several pixels size in the output image. This process is called *splatting*.

Relief Texture. To improve the rendering speed of 3D warping, the warping process can be factored into a relatively simple pre-warping step and a traditional texture mapping step. The texture mapping step can be performed by standard graphics hardware. This is the idea behind relief texture, a rendering technique proposed by Oliveira and Bishop [47]. A similar factoring approach has been proposed by Shade *et al.* in a two-step algorithm [55], where the depth is first forward warped before the pixel is backward mapped onto the output image.

Multiple-center-of-projection Images. The 3D warping techniques can be applied not only to the traditional perspective images, but also multi-perspective images as well. For example, Rademacher and Bishop [52] proposed to render novel views by warping multiple-center-of-projection images, or MCOP images.

8.4.2 Layered Depth Image Rendering

To deal with the disocclusion artifacts in 3D warping, Shade *et al.* proposed Layered Depth Image, or LDI [55], to store not only what is visible in the input image, but also what is behind the visible surface. In their paper, the LDI is constructed either using stereo on a sequence of images with known camera motion (to extract multiple overlapping layers) or directly from synthetic environments with known geometries. In an LDI, each pixel in the input image

contains a list of depth and color values where the ray from the pixel intersects with the environment.

Though an LDI has the simplicity of warping a single image, it does not consider the issue of sampling density. Chang *et al.* [11] proposed LDI trees so that the sampling rates of the reference images are preserved by adaptively selecting an LDI in the LDI tree for each pixel. While rendering the LDI tree, only the level of LDI tree that is the comparable to the sampling rate of the output image need to be traversed.

8.4.3 View-dependent Texture Mapping

Texture maps are widely used in computer graphics for generating photo-realistic environments. Texture-mapped models can be created using a CAD modeler for a synthetic environment. For real environments, these models can be generated using a 3D scanner or applying computer vision techniques to captured images. Unfortunately, vision techniques are not robust enough to recover accurate 3D models. In addition, it is difficult to capture visual effects such as highlights, reflections, and transparency using a single texture-mapped model.

To obtain these visual effects of a reconstructed architectural environment, Debevec *et al.* [17] used view-dependent texture mapping to render new views by warping and compositing several input images of an environment. This is the same as conventional texture mapping, except that multiple textures from different sampled viewpoints are warped to the same surface and averaged, with weights computed based on proximity of the current viewpoint to the sampled viewpoints. A three-step view-dependent texture mapping method was also proposed later by Debevec *et al.* [16] to further reduce the computational cost and to have smoother blending. This method employs visibility preprocessing, polygon-view maps, and projective texture mapping. More recently, Buehler *et al.* [5] apply a more principled way of blending textures based on relative angular position, resolution, and field-of-view.

8.5. Trade-off between Images and Geometry

Rendering with no geometry is expensive in terms of acquiring and storing the database. On the other hand, using explicit geometry, while more compact, may compromise output visual quality. So, an important question is, what is the right mix of image sampling size and quality of geometric information required to satisfy a mix of quality, compactness, and speed? Part of that question may be answered by analyzing the nature of plenoptic sampling.

8.5.1 Plenoptic Sampling Analysis

Many image-based rendering systems, especially light field rendering [37, 20, 57], have a tendency to rely on oversampling to counter undesirable aliasing effects in output display. Oversampling means more intensive data acquisition, more storage, and more redundancy. Sampling analysis in image-based rendering is a difficult problem because it involves unraveling the relationship among three tightly related elements: the depth and texture information of the scene, the number of sample images, and the rendering resolution, as shown in Figure 8.6. The presence of non-rigid effects (such as highlights, inter-reflection and translucency) significantly complicates this analysis, and is typically ignored. Non-rigid effects would very likely result in higher image sampling requirements than those predicted by analyses that ignore such effects. Chai *et al.* [9] recently studied the issue of *plenoptic sampling*. More specifically, they were interested in determining the number of image samples (e.g., from a 4D light field) and the amount of geometric and textural information needed to generate a continuous representation of the plenoptic function. The following two problems are studied under plenoptic sampling: (1) Finding the minimum sampling rate for light field rendering, and (2) finding the minimum sampling curve in the joint image and geometry space.

Chai *et al.* formulate the question of sampling analysis as a high dimensional signal processing problem. Rather than attempting to obtain a closed-form general solution to the 4D light field spectral analysis, they only analyze the bounds of the spectral support of the light field signals. A key observation in this paper is that the spectral support of a light field signal is bounded by only the

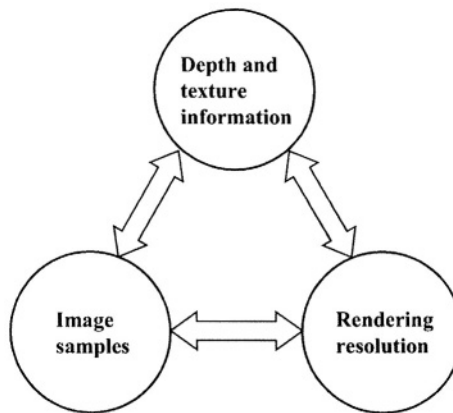


Figure 8.6. Plenoptic sampling. Quantitative analysis of the relationships among three key elements: depth and texture information, number of input images, and rendering resolution.

minimum and maximum depths, irrespective of how complicated the spectral support might be because of depth variations in the scene. Given the minimum and maximum depths, a reconstruction filter with an optimal and constant depth can be designed to achieve anti-aliased light field rendering.

The minimum sampling rate of light field rendering is obtained by compacting the replicas of the spectral support of the sampled light field within the smallest interval after the optimal filter is applied. How small the interval can be depends on the design of the optimal filter. More depth information results in tighter bounds of the spectral support, thus a smaller number of images. Plenoptic sampling in the joint image and geometry space determines the minimum sampling curve which quantitatively describes the relationship between the number of images and the information on scene geometry under a given rendering resolution. This minimal sampling curve can serve as one of the design principles for IBR systems. Furthermore, it bridges the gap between image-based rendering and traditional geometry-based rendering. Minimum sampling rate and minimum sampling curves are illustrated in Figure 8.7. Note that this analysis ignores the effects of both occlusion events and non-rigid motion.

As shown in Figure 8.7 (a), a minimum sampling rate (i.e., the minimum number of images) can be obtained if only minimum and maximum depths of the scene are known. Figure 8.7 (b) illustrates that any sampling point above the minimum sampling curve is redundant. Figure 11 in [9] demonstrated that the rendered images with five sampling points (of the number of images and the number of depth layers) above the minimum sampling curve are visually indistinguishable. Such a minimum sampling curve is also related to the rendering resolution, as shown in Figure 8.7 (c).

Isaksen *et al.* [24] did a similar analysis in frequency domain, in the context of their work on dynamically reparameterized light fields. Here, they were concerned about the effect of variable focus and depth-of-field on output quality. Zhang and Chen [66] extended the IBR sampling analysis by proposing a generalized sampling strategy to replace the conventional rectangular sampling in the high dimensional signal space. Their analysis was performed in continuous and discrete spatial domains.

There are a number of techniques that can be applied to reduce the size of the representation; they are usually based on local coherency either in the spatial or temporal domains. The following subsections describe some of these techniques.

8.5.2 View-dependent Geometry

Another interesting representation that trades off geometry and images is view-dependent geometry, first used in the context of 3D cartoons [51]. We can

potentially extend this idea to represent real or synthetically-generated scenes more compactly. As described in [29], view-dependent geometry is useful to accommodate the fact that stereo reconstruction errors are less visible during local viewpoint perturbations, but may show dramatic effects over large view changes. In areas where stereo data is inaccurate, they suggest that we may well represent these areas with view-dependent geometry, which comprises a set of geometry extracted at various positions (in [51], this set is manually created).

View-dependent geometry may also be used to capture visual effects such as highlights and transparency, which are likely to be locally coherent in image and viewpoint spaces. This is demonstrated in the work described in [21], in which structure from motion is first automatically computed from input images acquired using a camera following a serpentine path (raster style left-to-right and top-to-bottom). The system then generates local depth maps and textures used to produce new views in a manner similar to the Lumigraph [20]. The important issue of automatically determining the minimum amount of local depth maps and textures required has yet to be resolved. This area should be a fertile one for future investigation with potentially significant payoffs.

8.5.3 Dynamically Reparameterized Light Field

Recently, Isaksen *et al.* [24] proposed the notion of dynamically reparameterized light fields by adding the ability to vary the apparent focus within a light field using variable aperture and focus ring. Compared with the original light field and Lumigraph, this method can deal with a much larger depth variation in the scene by combining multiple focal planes. Therefore, it is suitable not only for outside-looking-in objects, but also for inside-looking-out environments. When multiple focus planes are used for a scene, a scoring algorithm is used before rendering to determine which focus plane is used during rendering.

While this method does not need to recover actual or approximate geometry of the scene for focusing, it does need to assign which focus plane to be used. The number of focal planes needed is not discussed. This light field variant exposes another factor that needs to be considered in the trade-off, i.e., the ability to vary the apparent focus on the scene (the better the focus/defocus effect required, the more image samples needed). It is not currently clear, though, how this need can be quantified in the trade-off.

8.5.4 Geometric Proxies

Many approximated geometric models, or geometric proxies have been proposed in various IBR systems in order to reduce the number of images needed for anti-aliased rendering. Light field, dynamically reparameterized light field and CMs have used simple planar surfaces. The Lumigraph used an approximated model extracted using "shape-from-silhouette." The Unstructured Lumigraph

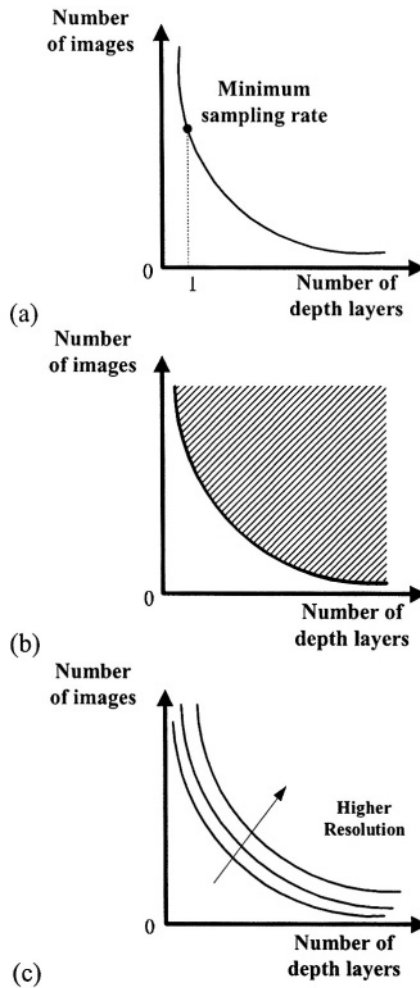


Figure 8.7. Minimum sampling: (a) the minimum sampling rate in image space; (b) the minimum sampling curve in the joint image and geometry space; (c) minimum sampling curves at different rendering resolutions.

work demonstrated that realistic rendering can be achieved although the proxies are significantly different from the true models. The image-based visual hull [41] is another geometry proxy that can be constructed and updated in real-time.

Acquiring an adequate geometric proxy is, however, difficult when the sampling of light field is very sparse. The geometric proxy, albeit approximate,

needs to be continuous because every desired ray must intersect some point on the proxy in order to establish the correspondence between rays. Traditional stereo reconstruction unfortunately cannot provide accurate enough geometric proxies especially at places where occlusion happens. Scam light field rendering [65] has been recently proposed to build a geometric proxy using only sparse correspondence.

8.6. Rendering with Layered Geometry

In this section, we discuss in detail a practical IBR technique, called *pop-up light field*, that addresses the following problem: Can we use a relatively sparse set of images of a complex scene and produce photorealistic virtual views free of aliasing? A straightforward approach would be to perform stereo reconstruction or to establish correspondence between all pixels of the input images. The geometric proxy is a depth map for each input image. Unfortunately, state-of-the-art automatic stereo algorithms are inadequate for producing sufficiently accurate depth information for realistic rendering. Typically, the areas around occlusion boundaries [32, 30] in the scene have the least desirable results, because it is very hard for stereo algorithms to handle occlusions without prior knowledge of the scene.

Pop-up light field approaches this problem by suggesting that it is not necessary to reconstruct accurate 3D information for each pixel in the input light field. The solution is to construct a pop-up light field by segmenting the input sparse light field into multiple coherent layers. Pop-up light field differs from other layered modeling and rendering approaches (e.g., [36, 56, 4]) in a number of ways. First, the number of layers needed in a pop-up light field is not pre-determined. Rather, it is decided interactively by the user. Second, the user specifies the layer boundaries in key frames. The layer boundary is then propagated to the remaining frames automatically. Third, layered representation is simple. Each layer is represented by a planar surface without the need for per-pixel depth. Fourth and most importantly, layers are coherent so that anti-aliased rendering using these coherent layers is achieved. Each coherent layer must have sufficiently small depth variation so that anti-aliased rendering of the coherent layer itself becomes possible. Moreover, to render each coherent layer with its background layers, not only accurate layer segmentation is required on every image, but segmentation across all images must be consistent as well.

Figure 8.8 represents a coherent layer L_j by a collection of corresponding layered image regions R_j^i in the light field images I^i . These regions are modeled by a simple geometric proxy without the need for accurate per-pixel depth. For example, a global planar surface (P_j) is used as the geometric proxy for each layer L_j in the example shown in Figure 8.8. To deal with complicated scenes

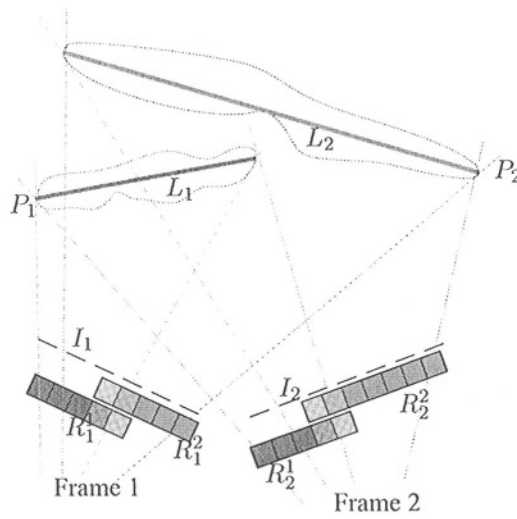


Figure 8.8. A light field (with images I_1 and I_2) can be represented by a set of coherent layers (L_1 and L_2). A coherent layer is a collection of layered images in the light field. For instance, L_1 is represented by layered image R_1^1 (from I_1) and R_2^1 (from I_2). Each layered image has an alpha matte associated with its boundary. Part of the scene corresponding to each layer (e.g., L_1) is simply modeled as a plane (e.g., P_1).

and camera motions, we can also use a local planar surface P_j^i to model the layer in every image i of the light field.

A layer in the pop-up light field is considered as "coherent" if the layer can be rendered free of aliasing by using a simple planar geometric proxy (global or local). Anti-aliased rendering occurs at two levels when

- 1 the layer itself is rendered; and
- 2 the layer is rendered with its background layers.

Therefore, to satisfy the first requirement, the depth variation in each layer must be sufficiently small, as suggested in [10]. Moreover, the planar surface can be adjusted interactively to achieve the best rendering effect. This effect of moving the focal plane has been shown in [25, 10].

However, to meet the second requirement, accurate layer boundaries must be maintained across all the frames to construct the coherent layers. A natural approach to ensuring segmentation coherence across all frames is to propagate the segmented regions on one or more key frames to all the remaining frames [56, 67]. Sub-pixel precision segmentation may be obtained on the key frames by meticulously zooming on the images and tracing the boundaries. Propagation from key frames to other frames, however, causes inevitable under-segmentation

or over-segmentation of a foreground layer. Typically over-segmentation of a foreground layer leads to the inclusion of background pixels, thus introducing ghosting along the occlusion boundaries in the rendered image. A possible example of foreground over-segmentation is exhibited in Figure 4 (g) of [56] where black pixels on the front object's boundary can be observed. To alleviate the rendering artifacts caused by over-segmentation or under-segmentation of layers, we need to refine the layer boundary with alpha matting [50].

Figure 8.8 illustrates coherent layers of a pop-up light field. All the pixels at each coherent layer have consistent depth values (to be exact, within a depth bound), but may have different fractional alpha values along the boundary.

To produce fractional alpha mattes for all the regions in a coherent layer, a straightforward solution is to apply video matting [14]. The video matting problem is formulated as a maximum a posterior (MAP) estimation as in Bayesian matting (c.f. Equation (4) of [15]),

$$\begin{aligned} & \arg \max_{F, B, \alpha} L(F, B, \alpha | C) \\ = & \arg \max_{F, B, \alpha} L(C | F, B, \alpha) + L(F) + L(B) + L(\alpha) \end{aligned} \quad (8.5)$$

where C is the observed color for a pixel, and F , B and α are foreground color, background color and alpha value to be estimated, respectively. In video matting [14], the log likelihood for the alpha $L(\alpha)$ is assumed constant so that $L(\alpha)$ is dropped from Equation (8.5).

In video matting, the optical flow is applied to the trimap (the map of foreground, background and uncertain region), but not to the output matte. The output foreground matte is produced by Bayesian matting on the current frame, based on the propagated trimap. Video matting works well if we simply replay the foreground mattes against a different background. However, these foreground mattes may not have in-between frame coherence that is needed for rendering novel views, i.e. interpolation.

A novel approach, called *coherence matting*, is to construct the alpha mattes in a coherent layer that have in-between frame coherence. First, the user-specified boundaries are propagated across frames. Second, the uncertain region along the boundary is determined. Third, the under-segmented background regions from multiple images are combined to construct a sufficient background image. Fourth, the alpha matte for the foreground image (in the uncertain region) is estimated. The key to coherence matting is to model the log likelihood for the alpha $L(\alpha)$ as a deviation from the feathering function across the corresponding layer boundaries. Note that, for a given layer, a separate foreground matte is estimated independently for each frame in the light field, and the coherence across frames is maintained by foreground boundary consistency.

Construct Pop-up Light Field. To construct a pop-up light field, an easy-to-use user interface (UI) is necessary. The user can easily specify, refine and

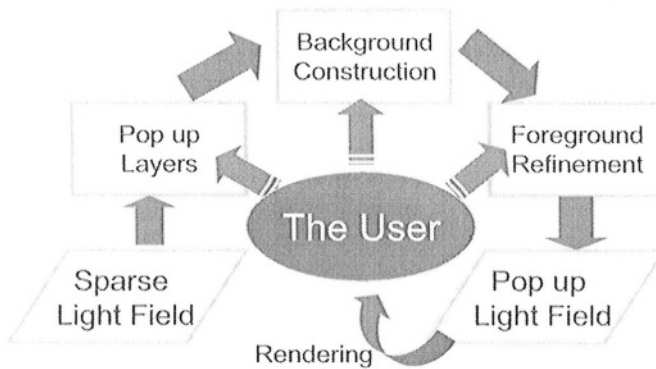


Figure 8.9. The pop-up light field construction.

propagate layer boundaries, and indicate rendering artifacts. More layers can be popped up and refined until the user is satisfied with the rendering quality.

Figure 8.9 summarizes the operations in the pop-up light field construction UI. The key is that a human is in the loop. The user supplies the information needed for layer segmentation, background construction and foreground refinement. By visually inspecting the rendering image from the pop-up light field, the user also indicates where aliasing occurs and thus which layer needs to be further refined. The user input or feedback is automatically propagated across all the frames in the pop-up light field. The four steps of operations in the UI are summarized as follows:

- 1 *Layer pop-up.* This step segments layers and specifies their geometries. To start, the user selects a key frame in the input light field, specifies regions that need to be popped up, and assigns the layer's geometry by either a constant depth or a plane equation. This step results in a coarse segmentation represented by a polygon. The polygon region and geometry configuration can be automatically propagated across frames. Layers should be popped up in order of front to back.
- 2 *Background construction.* This step obtains background mosaics that are needed to estimate the alpha mattes of foreground layers. Note that the background mosaic is useful only for the pixels around the foreground boundaries.
- 3 *Foreground refinement.* Based on the constructed background layers, this step refines the alpha matte of the foreground layer by applying the coherence matting algorithm. Unlike layer pop-up in step 1, foreground refinement in this step should be performed in back-to-front order.

- 4 *Rendering feedback.* Any modification to the above steps will update the underlying pop-up light field data. The rendering window will be refreshed with the changes as well. By continuously changing the view-point the user can inspect for rendering artifacts. The user can mark any rendering artifacts such as ghosting areas by brushing directly on the rendering window. The corresponding frame and layer will then be selected for further refinement. The rendering of pop-up light field can achieve a real-time frame rate by using modern graphics hardware.

Figure 8.10 shows an aliasing-free novel view rendered using the pop-up light field constructed from the Plaza sequence, which is a collection of only 16 images. The sequence was captured by a series of "time-frozen cameras" arranged along a line or curve. Because the scene is very complex, stereo reconstruction is very difficult. Note that nearly perfect matting is achieved for the floating papers in the air. The boundaries for the foreground characters are visually acceptable, made possible mainly by the coherent layers produced by coherence matting.



Figure 8.10. Result of pop-up light field rendering of the *Plaza* sequence rendered from a novel viewpoint (in the position midway between the 11th and 12th frames). The input consists of only 16 images. 16 layers are used to model the pop-up light field.

Pop-up light field is an image-based modeling technique that does not rely on accurate 3D depth/surface reconstruction. Rather, it is based on accurate layer extraction/segmentation in the light field images. In a way, we trade a difficult correspondence problem in 3D reconstruction for another equally difficult segmentation problem. However, for a user, it is much easier to specify accurate contours in images than accurate depth for each pixel.

8.7. Discussion

Image-based rendering is an area that straddles both computer vision and computer graphics. The continuum between images and geometry is evident from the image-based rendering techniques reviewed in this chapter. However, the emphasis of this chapter is more on the aspect of rendering and not so much on image-based modeling. Other important topics such as lighting and animation are also not treated here.

In this chapter, image-based rendering (IBR) techniques are divided based on how much geometric information has been used, i.e., whether the method uses explicit geometry (e.g., LDI), implicit geometry or correspondence (e.g., view interpolation), or no geometry at all (e.g., light field). Other methods of dividing image-based rendering techniques have also been proposed by others, such as on the nature of the pixel indexing scheme [26].

8.7.1 Challenges

Efficient Representation. What is very interesting is the trade-off between geometry and images needed to use for anti-aliased image-based rendering. The design choices for many IBR systems were made based on the availability of accurate geometry. Plenoptic sampling provides a theoretical foundation for designing IBR systems.

Both light field rendering and Lumigraph avoid the feature correspondence problem by collecting many images with known camera poses. Because of the size of the database (even after compression), virtual walkthroughs of a real scene using light fields have not yet been fully demonstrated.

Rendering Performance. How would one implement the "perfect" rendering engine? One possibility would be to adapt current hardware accelerators to produce, say, an approximate version of an LDI or a Lumigraph by replacing it with view-dependent texture-mapped sprites. The alternative is to design new hardware accelerators that can handle both conventional rendering and IBR. An example in this direction is the use of PixelFlow to render image-based models [42]. PixelFlow [18] is a high-speed image generation architecture that is based on the techniques of object-parallelism and image composition.

Capturing. Panoramas are relatively not difficult to construct. Many previous systems have been built to construct cylindrical and spherical panoramas by stitching multiple images together (e.g., [39, 60, 13, 44, 61]). When the camera motion is very small, it is possible to put together only small stripes from registered images, i.e., slit images (e.g., [68, 49]), to form a large panoramic mosaic. Capturing panoramas is even easier if omnidirectional cameras (e.g., [46, 45]) or fisheye lens [64] are used.

It is, however, very difficult to construct a continuous 5D complete plenoptic function [44, 28] because it requires solving the difficult feature correspondence problem. To date, no one has yet shown a collection of 7D complete plenoptic functions (authoring a dynamic environment with time-varying lighting conditions is a very interesting problem).

Because of the large amount of data used in most IBR techniques, data compression is essential to make it practical for storage and transmission. Some of the challenges in IBR compression such as rendering directly from compressed streams and producing more efficient scalable and embedded representations are still open problems.

8.7.2 Two Scenarios

Image-based rendering can have many interesting applications. Two scenarios, in particular, are worth pursuing:

Large Environments. Many successful techniques, e.g., light field, CMs, have restrictions on how much a user can change his viewpoint. QuickTime VR [13] is still popular for showcasing large environments despite the visual discomfort caused by jumping between panoramas. While this can be alleviated by having multiple panoramic clusters and enabling single DOF transitioning between these clusters [27], the range of virtual motion is nevertheless still restricted. To move around in a large environment, one has to combine image-based techniques with geometry-based models, in order to avoid excessive amount of data required.

Dynamic Environments. Until now, most of image-based rendering systems have been focused on static environments. With the development of panoramic video systems, it is conceivable that image-based rendering can be applied to dynamic environments as well. Two issues must be studied: sampling (how many images should be captured), and compression (how to reduce data effectively).

8.7.3 Concluding Remarks

In this chapter, we have surveyed recent developments in the area of image-based rendering, and in particular, categorized them based on the extent of use of geometric information in rendering. Geometry is used as a means of compressing representations for rendering, with the limit being a single 3D model with a single static texture. While the purely image-based representations have the advantage of photorealistic rendering, they come with the high costs of data acquisition and storage requirements.

Demands on realistic rendering, compactness of representation, speed of rendering, and costs and limitations of computer vision reconstruction techniques force the practical representation to be fall somewhere between the two extremes. It is clear that IBR and the traditional 3D model-based rendering techniques have complimentary characteristics that can be capitalized. As a result, we believe that it is important that future graphics rendering hardware and video technology be customized to handle both the traditional 3D model-based rendering as well as IBR.

References

- [1] E. H. Adelson and J. Bergen. "The plenoptic function and the elements of early vision". In *Computational Models of Visual Processing*, pp. 3-20. MIT Press, Cambridge, MA, 1991.
- [2] D. G. Aliaga and I. Carlbom. "Plenoptic stitching: A scalable method for reconstructing 3D interactive walkthroughs". *Computer Graphics (SIGGRAPH'01)*, pp. 443-450, August 2001.
- [3] S. Avidan and A. Shashua. "Novel view synthesis in tensor space". In *Conference on Computer Vision and Pattern Recognition*, pp. 1034-1040, San Juan, Puerto Rico, June 1997.
- [4] S. Baker, R. Szeliski and P. Anandan. "A layered approach to stereo reconstruction". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 434-441, 1998.
- [5] C. Buehler, M. Bosse, L. McMillan, S. Gortler and M. Cohen. "Unstructured Lumigraph rendering". *Computer Graphics (SIGGRAPH'01)*, pp. 425-432, August 2001.
- [6] E. Camahort. "4D light-field modeling and rendering". Technical Report TR01-52, The University of Texas at Austin, May 2001.
- [7] E. Camahort, A. Leros and D. Fussell. "Uniformly sampled light fields". In *9th Eurographics Workshop on Rendering*, pp. 117-130, Vienna, Austria, June/July 1998.
- [8] D. Capel and A. Zisserman. "Super-resolution from multiple views using learnt image models". In *Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 627-634, Kauai, HI, December, 2001.
- [9] J. X. Chai, X. Tong, S. C. Chan and H. Y. Shum. "Plenoptic sampling". *Computer Graphics (SIGGRAPH'00)*, pp. 307-318, July 2000.

- [10] J. X. Chai, X. Tong, S. C. Chan and H. Y. Shum. "Plenoptic sampling". In *Proceedings of ACM SIGGRAPH'00*, pp. 307-318, 2000.
- [11] C. Chang, G. Bishop and A. Lastra. "LDI tree: A hierarchical representation for image-based rendering". *Computer Graphics (SIGGRAPH'99)*, pp. 291-298, August 1999.
- [12] S. Chen and L. Williams. "View interpolation for image synthesis". *Computer Graphics (SIGGRAPH'93)*, pp. 279-288, August 1993.
- [13] S. E. Chen, "QuickTime VR – An image-based approach to virtual environment navigation". *Computer Graphics (SIGGRAPH'95)*, pp. 29–38, August 1995.
- [14] Y. Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin and R. Szeliski. "Video matting of complex scenes". In *Proceedings of ACM SIGGRAPH'02*, pp. 243-248, 2002.
- [15] Y. Y. Chuang, B. Curless, D. H. Salesin and R. Szeliski. "A bayesian approach to digital matting". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, pp. vol. II, pp. 264-271, 2001.
- [16] P. Debevec, Y. Yu and G. Borshukov. "Efficient view-dependent image-based rendering with projective texture-mapping". In *Proc. 9th Eurographics Workshop on Rendering*, pp. 105-116, 1998.
- [17] P. E. Debevec, C. J. Taylor and J. Malik. "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach". *Computer Graphics (SIGGRAPH'96)*, pp. 11-20, August, 1996.
- [18] J. Eyles, S. Molnar, J. Poulton, T. Greer, A. Lastra, N. England and L. Westover. "Pixelflow: The realization". In *Siggraph/Eurographics Workshop on Graphics Hardware*, Los Angeles, CA, August, 1997.
- [19] O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.
- [20] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen. "The Lumigraph". In *Computer Graphics Proceedings, Annual Conference Series*, pp. 43-54, Proc. SIGGRAPH'96 (New Orleans), ACM SIGGRAPH, August, 1996.
- [21] B. Heigl, R. Koch, M. Pollefeys, J. Denzler and L. Van Gool. "Plenoptic modeling and rendering from image sequences taken by hand-held camera". In *DAGM*, pp. 94-101, 1999.
- [22] I. Ihm, S. Park and R. Lee. "Rendering of spherical light fields". In *Pacific Graphics*, pp. 59-68, Seoul, Korea, October, 1997.
- [23] M. Irani and S. Peleg. "Improving resolution by image registration". *Graphical Models and Image Processing*, vol. 53, no. 3, pp. 231-239, May 1991.
- [24] A. Isaksen, L. McMillan and S. Gortler. "Dynamically reparameterized light fields". *Computer Graphics (SIGGRAPH'00)*, pp. 297-306, July, 2000.
- [25] A. Isaksen, L. McMillan and S. J. Gortler. "Dynamically reparameterized light fields". In *Proceedings of ACM SIGGRAPH'00*, pp. 297-306, 2000.
- [26] S. B. Kang. "A survey of image-based rendering techniques". In *VideoMetrics*, SPIE, vol. 3641, pp. 2-16, 1999.
- [27] S. B. Kang and P. K. Desikan. "Virtual navigation of complex scenes using clusters of cylindrical panoramic images". In *Graphics Interface*, pp. 223-232, Vancouver, Canada, June, 1998.
- [28] S. B. Kang and R. Szeliski. "3-D scene data recovery using omnidirectional multibase-line stereo". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 364-370, San Francisco, California, June, 1996.

- [29] S. B. Kang, R. Szeliski and P. Anandan. "The geometry-image representation tradeoff for rendering". In *International Conference on Image Processing*, Vancouver, Canada, September, 2000.
- [30] S. B. Kang, R. Szeliski and J. Chai. "Handling occlusions in dense multi-view stereo". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, vol. I, pp. 103-110, 2001.
- [31] A. Katayama, K. Tanaka, T. Oshino and H. Tamura. "A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images". In *Stereoscopic Displays and Virtual Reality Systems II, Proc. SPIE*, vol. 2409, pp. 11-20, S. Fisher, J. Merritt, and B. Bolas, Eds., 1995.
- [32] V. Kolmogorov and R. Zabih. "Multi-camera scene reconstruction via graph cuts". In *Proc. European Conference on Computer Vision*, pp. 82-97, 2002.
- [33] S. Laveau and O. Faugeras. "3-D scene representation as a collection of images and fundamental matrices". Technical Report 2205, INRIA-Sophia Antipolis, February, 1994.
- [34] S. Laveau and O. D. Faugeras. "3-D scene representation as a collection of images". In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, vol. A, pp. 689-691, IEEE Computer Society Press, Jerusalem, Israel, October, 1994.
- [35] J. Lengyel. "The convergence of graphics and vision". *IEEE Computer*, vol. 31, no. 7, pp.46-53, July, 1998.
- [36] J. Lengyel and J. Snyder. "Rendering with coherent layers". In *Proceedings of ACM SIGGRAPH'97*, pp. 233 - 242, 1997.
- [37] M. Levoy and P. Hanrahan. "Light field rendering". In *Computer Graphics Proceedings, Annual Conference Series*, pp. 31-42, Proc. SIGGRAPH'96 (New Orleans), ACM SIGGRAPH, August, 1996. .
- [38] M. Magnor and B. Girod. "Model-based coding of multi-viewpoint imagery". In *SPIE Visual Communication and Image Processing*, vol. 4067, no. 2, pp. 14-22, Perth, Australia, June, 2000.
- [39] S. Mann and R. W. Picard. "Virtual bellows: Constructing high-quality images from video". In *First IEEE International Conference on Image Processing (ICIP-94)*, vol. I, pp. 363-367, Austin, Texas, November, 1994.
- [40] W. Mark, L. McMillan and G. Bishop. "Post-rendering 3D warping". In *Proc. Symposium on 3D Graphics*, pp. 7-16, 1997.
- [41] W. Matusik, C. Buehler, R. Raskar, S. Gortler and L. McMillan. "Image-based visual hulls". *Computer Graphics (SIGGRAPH'00)*, pp. 369-374, July, 2000.
- [42] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra and C. McCue. "Real-time rendering of real world environments". In *Eurographics Workshop on Rendering*, Granada, Spain, June, 1999.
- [43] L. McMillan. "An Image-based Approach to Three-dimensional Computer Graphics". Technical report, Ph.D. Dissertation, UNC Computer Science TR97-013, 1999.
- [44] L. McMillan and G. Bishop. "Plenoptic modeling: An image-based rendering system". *Computer Graphics (SIGGRAPH'95)*, pp. 39-46, August, 1995.
- [45] V. S. Nalwa. "A True Omnidirectional Viewer". Technical report, Bell Laboratories, Holmdel, NJ, USA, February, 1996.
- [46] S. Nayar. "Catadioptric omnidirectional camera". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 482-488, San Juan, Puerto Rico, June, 1997.

- [47] M. Oliveira and G. Bishop. "Relief Textures". Technical report, UNC Computer Science TR99-015, March, 1999.
- [48] S. Peleg and M. Ben-Ezra. "Stereo panorama with a single camera". In *Proc. Computer Vision and Pattern Recognition Conf.*, 1999.
- [49] S. Peleg and J. Herman. "Panoramic mosaics by manifold projection". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 338-343, San Juan, Puerto Rico, June, 1997.
- [50] T. Porter and T. Duff. "Compositing digital images". *Computer Graphics*, vol. 18, no. 3, pp. 253-259, 1984.
- [51] P. Rademacher. "View-dependent geometry". *SIGGRAPH*, pp. 439-446, August, 1999.
- [52] P. Rademacher and G. Bishop. "Multiple-center-of-projection images". In *Computer Graphics Proceedings, Annual Conference Series*, pp. 199-206, Proc. SIGGRAPH'98 (Orlando), ACM SIGGRAPH, July, 1998.
- [53] D. Scharstein. "Stereo vision for view synthesis". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 852-857, San Francisco, California, June, 1996.
- [54] S. M. Seitz and C. M. Dyer. "View morphing". In *Computer Graphics Proceedings, Annual Conference Series*, pp. 21-30, Proc. SIGGRAPH'96 (New Orleans), ACM SIGGRAPH, August, 1996.
- [55] J. Shade, S. Gortler, L.-W. He and R. Szeliski. "Layered depth images". In *Computer Graphics (SIGGRAPH'98) Proceedings*, pp. 231-242, ACM SIGGRAPH, Orlando, July, 1998.
- [56] J. Shade, S. Gortler, L. W. He and R. Szeliski. "Layered depth images". In *Proceedings of ACM SIGGRAPH'98*, pp. 231-242, 1998.
- [57] H. Y. Shum and L. W. He. "Rendering with concentric mosaics". In *SIGGRAPH'99*, pp. 299-306, ACM SIGGRAPH, Los Angeles, August, 1999.
- [58] H.-Y. Shum and R. Szeliski. "Construction and refinement of panoramic mosaics with global and local alignment". In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 953-958, Bombay, January, 1998.
- [59] P. P. Sloan, M. F. Cohen, and S. J. Gortler. "Time critical Lumigraph rendering". In *Symposium on Interactive 3D Graphics*, pp. 17-23, Providence, RI, USA, 1997.
- [60] R. Szeliski. "Video mosaics for virtual environments". *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 22-30, March, 1996.
- [61] R. Szeliski and H. Y. Shum. "Creating full view panoramic image mosaics and texture-mapped models". *Computer Graphics (SIGGRAPH'97)*, pp. 251-258, August, 1997.
- [62] T. Wong, P. Heng, S. Or and W. Ng. "Image-based rendering with controllable illumination". In *Proceedings of the 8-th Eurographics Workshop on Rendering*, pp. 13-22, St. Etienne, France, June, 1997.
- [63] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. "Surface light fields for 3D photography". *Proceedings of SIGGRAPH*, pp. 287-296, July, 2000.
- [64] Y. Xiong and K. Turkowski. "Creating image-based VR using a self-calibrating fisheye lens". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 237-243, San Juan, Puerto Rico, June, 1997.

- [65] J. Yu, L. McMillan and S. Gortler. "Scam light field rendering". In *Proc. Pacific Graphics 2002*, pp. 137-144, 2002.
- [66] C. Zhang and T. Chen. "Generalized Plenoptic Sampling". Technical Report AMP01-06, Carnegie Mellon University, September, 2001.
- [67] Z. Zhang, L. Wang, B. Guo, and H. Y. Shum. "Feature-based light field morphing". In *Proceedings of ACM SIGGRAPH'02*, pp. 457-464, 2002.
- [68] J. Y. Zheng and S. Tsuji. "Panoramic representation of scenes for route understanding". In *Proc. of the 10th Int. Conf. Pattern Recognition*, pp. 161-167, June, 1990.

This page intentionally left blank

Chapter 9

IMAGE-BASED RELIGHTING: REPRESENTATION AND COMPRESSION

Tien-Tsin Wong and Pheng-Ann Heng

*Department of Computer Science & Engineering, The Chinese University of Hong Kong,
Hong Kong*

ttwong@acm.org, pheng@cse.cuhk.edu.hk

Abstract Image-based modeling and rendering has been demonstrated as a cost-effective and efficient approach to computer games and virtual reality applications. The computational model that most image-based techniques are based on is the plenoptic function. Since the original formulation of the plenoptic function does not include illumination, previous image-based applications simply assume that the illumination is fixed. We have proposed a new formulation of the plenoptic function, called the *plenoptic illumination function*, which explicitly specifies the illumination component. Techniques based on this formulation can be extended to support relighting as well as view interpolation. The core of this framework is compression, and we show how to exploit three types of data correlation, the *intra-pixel*, the *inter-pixel* and the *inter-channel* correlations, in order to achieve a manageable storage size. The proposed coding method outperforms JPEG, JPEG2000 and MPEG.

Keywords: Image-based relighting, image-based modeling and rendering, plenoptic illumination function, virtual reality, spherical harmonic, multimedia data compression, panorama

9.1. Introduction

The plenoptic function [1] was originally proposed for evaluating low-level human vision models. In the recent years, several image-based techniques [18, 6, 15, 8, 25] that are based on this computational model has been proposed to interpolate views. The original formulation of the plenoptic function is very general. All the illumination and scene changing factors are embedded in a single aggregate time parameter. However, it is too general to be useful. This is also one reason that most early research concentrates on the view interpolation

(the sampling and interpolation of the viewing direction and viewpoint) and leaves the time parameter untouched. The time parameter (the illumination and the scene) is usually assumed constant for simplicity. Unfortunately, the capability to change illumination (*relight*) [30, 28] is traditionally an important parameter in computer graphics and virtual reality. Its existence enhances the 3D illusion. Moreover, if the modification of the illumination configuration is performed in the image basis, the relighting time will be independent of scene complexity. Hence, we can easily achieve *dynamic lighting* [32] (the ability to modify lighting in real-time) of complex background in computer game applications.

In this chapter, we extract the *illumination* component from the aggregate time parameter. A new formulation that explicitly specifies the illumination component is proposed. We call it the *plenoptic illumination function* [29, 27]. Techniques based on it can be extended to support relighting as well as view interpolation. To generalize the relighting process to support various illumination configurations, we make use of the superposition properties of images. We show that the plenoptic illumination function allows us to relight image-based scenes with complex lighting conditions.

Introducing a new dimension in the plenoptic function, however, suffers from an increase of storage requirement. To make the model practical, we point out three types of data correlation that can be exploited. They are the *intra-pixel*, the *inter-pixel* and the *inter-channel* data correlations. A series of compression methods [31, 10, 14] is recommended to reduce the data storage to a manageable size. We shall show that the proposed compression scheme outperforms those popular image and video coding standards such as JPEG, JPEG2000 and MPEG.

9.2. Computational Model

9.2.1 The Plenoptic Function

Adelson and Bergen [1] proposed a seven-dimensional *plenoptic function* for evaluating the low-level human vision models. It describes the radiance received along any direction $\vec{V}$ arriving at any point $\vec{E}$ in space, at any time t and over any range of wavelength λ .

$$I = P(\vec{V}, \vec{E}, t, \lambda), \quad (9.1)$$

where I is the radiance; $\vec{E}$ is the position of the center of projection or the viewpoint; $\vec{V}$ specifies the viewing direction originated from the viewpoint; t is the time parameter; λ is the wavelength. Basically, the function tells us how the environment looks like when our eye is positioned at $\vec{E}$. The time parameter t actually models all other unmentioned factors such as the change

of illumination and the change of the scene. When t is constant, the scene is static and the illumination is fixed.

9.2.2 The Plenoptic Illumination Function

The original formulation of the plenoptic function is very general. However, the illumination and other scene changing factors are embedded inside a single time parameter t . Techniques [18, 6, 8, 15] based on this model also inherit this rigidity. Most of them assume that the illumination is unchanged and the scene is static, *i.e.* t is constant. However, the ability to express the illumination configuration is traditionally an important parameter in image synthesis. We proposed a new formulation of the plenoptic function to include the illumination component [30, 29]. We extract the illumination component ($\vec{L}$) from the aggregate time parameter t and explicitly specify it in the following new formulation,

$$I = P_I(\vec{L}, \vec{V}, \vec{E}, t', \lambda), \quad (9.2)$$

where $\vec{L}$ specifies the direction of a directional light source illuminating the scene; t' is the time parameter after extracting the illumination component.

The difference between this new formulation and the original (Eq. (9.1)) is the explicit specification of an illumination component, $\vec{L}$. The new function tells us the radiance coming from a viewing direction $\vec{V}$ arriving at our eye $\vec{E}$ at any time t' over any wavelength λ , when the whole scene is illuminated by a directional light source with the lighting direction $-\vec{L}$ (the negative sign is because we use local coordinate system). Intuitively speaking, it tells us how the environment looks like when the scene is illuminated by a directional light source (Figure 9.1).

Expressing the illumination component using a directional light source is not the only way. In fact, one can parameterize the illumination component using a point light source [16]. Then this *point-source formulation* will tell us how the environment looks like when illuminated by a point light source positioned at certain point, say $\vec{S} = (S_x, S_y, S_z)$. The reason we choose the *directional-source formulation* (Eq. (9.2)) is that specifying a direction requires only two extra parameters, while specifying a position in space requires three

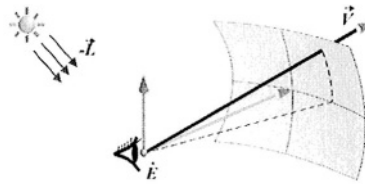


Figure 9.1. Geometry components of the plenoptic illumination function.

extra parameters. We try to minimize the dimensionality of the new formulation as the original plenoptic function already involves seven dimensions. Moreover, the directional-source formulation is also physically meaningful. Since the light vector due to a directional light source is constant throughout the space, the radiance in Eq. (9.2) should be the reflected radiance from the surface element, where the reflection takes place, when this surface element is illuminated by a light ray along $-\vec{L}$. We shall see in Section 9.4.3 how this property facilitates the image-based relighting when a non-directional light source (e.g. point source) is specified.

9.3. Sampling

Sampling the plenoptic illumination function is actually a process of taking pictures. The question is how to take pictures. The time parameter t' (the scene) is assumed fixed and the wavelength parameter λ is conventionally sampled and reconstructed at three positions (red, blue and green). Chai *et al.* [4] have addressed the theoretical issue on how to optimally take samples for the light field representation. On the other hand, Lin *et al.* [16] proved the theoretical sampling bound for image-based relighting. Interested readers are referred to [16] for an in-depth proof.

Since the parameter $\vec{L}$ is a directional vector, its sampling is equivalent to taking samples on the surface of a sphere. For simplicity, we take samples on a spherical grid as depicted in Figure 9.2. The disadvantage is that sample points are not evenly distributed on the sphere. More samples are placed near the poles of sphere.

For synthetic scenes, the samples can be easily collected by rendering the scene with a directional light source oriented in the required lighting direction. For real scenes, spotlight positioned at a sufficiently far distance can be used to approximate a directional light source. However, precise control of the lighting direction may require the construction of a robotic arm. In this chapter, we demonstrate the usefulness of the plenoptic illumination function with synthetic

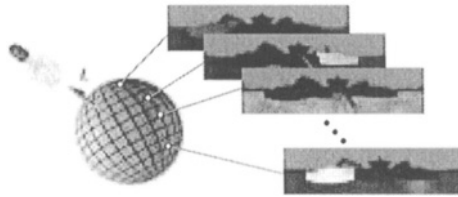


Figure 9.2. Sampling the lighting direction on the spherical grid.

data only. It is obvious that the function can also be applied to real data when available.

The sampled data can simply be stored in a multi-dimensional table of radiance values, indexed by light vector, viewing vector (screen coordinates and viewpoints), and wavelength (color channels). Since the size of table is enormous, in Sections 9.5-9.8, we present a series of compression techniques in order to effectively reduce its size to a manageable level.

9.4. Relighting

9.4.1 Reconstruction

Given a desired light vector which is not one of the samples, the desired image of the scene can be estimated by interpolating the samples. The interpolation on the illumination dimension is called *relighting*.

We first consider the relighting with a single directional light source. The simplest interpolation is picking the nearest sample as the result. The disadvantage is the discontinuity as the desired light vector moves. This discontinuity is not noticeable in Figure 9.3(b) as it is a still picture. But the specular highlight locates at the wrong position as compared to the correct one in Figure 9.3(a). Figures 9.3(c) and (d) show the results of bilinear and bicubic interpolations respectively. Although the highlight in the bilinear case is closer to the correct location, it looks less specular. The highlight is more accurate in the case of bicubic interpolation. In general, the result improves as we employ higher-order interpolation. The accuracy of result also depends on the actual geometry of the scene and its surface properties.

Besides the polynomial basis functions, other basis functions can also be employed. Nimeroff *et al.* [20] used steerable functions for the relighting due to the natural illumination. Eigenimages [2, 19, 21] extracted from the principal component analysis are the popular basis images used in the recognition of object under various illumination configurations. In our work, we use spherical harmonics [7] as the basis functions. Instead of interpolation in the spatial

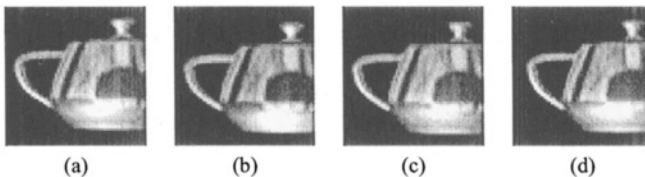


Figure 9.3. Relighting as an interpolation process. (a) Synthetic (correct) image. (b) The nearest neighbor. (c) Bilinear interpolation. (d) Bicubic interpolation.

domain, we interpolate the coefficients in the frequency domain. One major reason we choose spherical harmonics is that zonally sampling the coefficients also significantly reduces the storage requirement (described in Section 9.5).

9.4.2 Superposition

Even though the sampled plenoptic illumination function only tells us how the environment looks like when it is illuminated by a directional light source with unit intensity, other illumination configuration can be simulated by making use of the properties of *image superposition*. Image-based relighting with various illumination configurations can be done by calculating the following formula for each pixel for each color channel.

$$\sum_i^n P_I^*(\vec{L}_i) L_r(\vec{L}_i), \quad (9.3)$$

where n is the total number of desired light sources; $\vec{L}_i$ is the direction of the i -th desired light source; $P_I^*(\vec{L}_i)$ is the result of interpolating the samples given the desired light vector $\vec{L}_i$. The parameters $\vec{V}$, $\vec{E}$, t' and λ are dropped for simplicity; L_r is the radiance along $\vec{L}_i$.

It allows us to manipulate the direction, the color, and the number of the desired light sources. Although reference images are all captured under a white light, image relit by colored light sources can be approximated by feeding different values of L_r to different color channel. An image relit by two light sources can be synthesized by superimposing two images, each relit by a single light source. Figure 9.4 shows an image-based scene relit by a point source, a directional source, a spotlight, and a slide projector source. Note that shadow can also be approximated if it exists in the reference images.

Figure 9.5 shows further examples of panoramic images relit by various complex lighting conditions. The first column of Figure 9.5 shows three relit panoramic images. The second and third columns show the corresponding perspective snapshots which are generated by image warping. Panoramas in

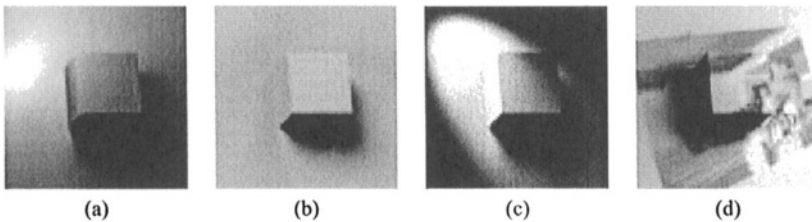


Figure 9.4. (a) Point light source. (b) Directional light source. (c) Spotlight. (d) Slide projector source.

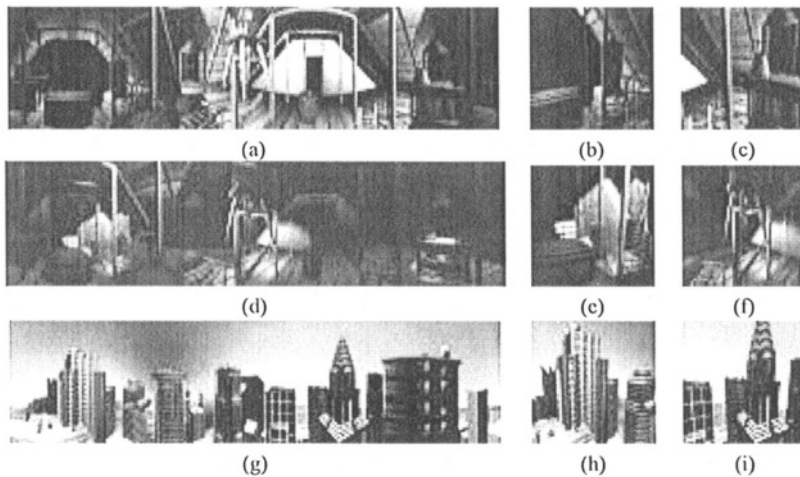


Figure 9.5. Relighting of panoramas. (a)-(f): attic, and (g)-(i): city.

Figures 9.5 (a) and (g) are relit with a single directional light source while Figure 9.5 (d) is relit with multiple spotlights and slide projector sources. Note how the illumination in the region with occlusion (pillar & chair) is correctly accounted in Figures 9.5 (e) and (f).

The attic scene in Figure 9.5 (a) and (d) contains 50k triangles and each reference image requires 133 seconds to render on a SGI Octane with a MIPS 10000 CPU, using the software renderer Alias|Wavefront. The city scene in Figure 9.5 (g) contains 187k triangles, and each reference image requires 337 seconds to render. For both cases, a 1024×256 cylindrical panorama is used to represent the scene. The relighting of both image-based scenes is real-time using our latest GPU-accelerated relighting engine [32]. This demonstrates the major advantage of image-based computer graphics – the rendering independence of scene complexity. A panoramic viewer with real-time relighting ability is available at the web address listed in the section of Web Availability.

9.4.3 Non-directional Light Source

The plenoptic illumination function tells us the reflected radiance from the *surface element where the physical reflection takes place* when that surface element is illuminated by a directional light along $\vec{L}$. When a directional light source is specified for image-based relighting, we simply feed the specified $\vec{L}$ to Eq. (9.3). However, if a non-directional light source (such as point source and spotlight) is specified for relighting, only the position of light source is given. The light ray impinging on the surface element is equal to the vector

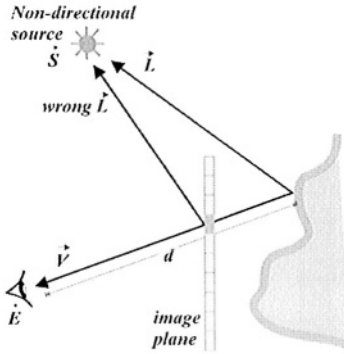


Figure 9.6. Finding the correct light vector.

from the surface element towards the position of the non-directional light source (Figure 9.6). It is *not* the vector from the pixel window to the light source as the actual reflection does not take place at the pixel window. Therefore, the only way to calculate the correct light vector for a non-directional light source is to know the depth value d and use the following equation

$$\vec{L} = \vec{S} - \vec{E} + \frac{\vec{V}}{|\vec{V}|}d \quad (9.4)$$

where $\vec{S}$ is the position of non-directional light source; and d is the depth value.

Area and volume light sources can also be simulated if the light source is subdivided into a finite number of point sources and relighting is done for each approximated point source.

9.5. Intra-Pixel Compression

The major drawback of including the illumination component is the increase of storage requirement. Without compression, the data size for a 1024×256 image-based scene sampled under 1,200 (the sampling rate on the spherical grid is 30 × 40) lighting conditions requires 900MB of storage. Therefore, compression is a must. We proposed a compression scheme which exploits three kinds of data correlation, namely *intra-pixel*, *inter-pixel*, and *inter-channel* correlations.

Consider a pixel in an image-based scene, there must be an associated viewing ray which connects the viewpoint to the pixel window. If the viewpoint, the viewing direction and the scene are all static during the data capture and only the illumination is allowed to change, it is very likely that radiance values captured under various lighting conditions are strongly correlated because most geomet-

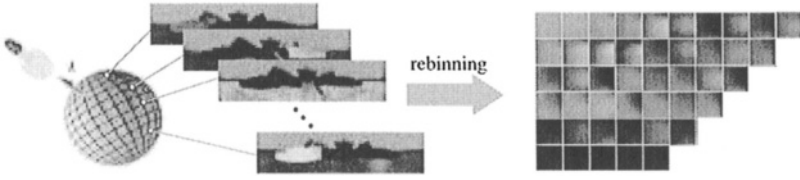


Figure 9.7. To increase the data redundancy, reference images are rebinned to form tiles of radiance values.

ric factors are frozen. Geometry is usually the major source of discontinuity of radiance values [17]. Moreover the received radiance values are strongly related to the surface reflectance of the visible surface element. However, they are not the same because there exists discontinuity in the received radiances due to the shadow cast by the nearby geometry.

We first group together those radiance values related to the same pixel window (or the same viewing ray) and try to make use of this *intra-pixel data correlation*. Figure 9.7 illustrates that we rebin the radiance values from different reference images to form tiles of radiance values. Each tile is a spherical function corresponds to all possible values of that pixel window. This spherical function is indexed by the light vector.

To compress them, we transform the spherical function into the spherical harmonic domain [7], and the resultant spherical harmonic coefficients are zonally sampled and quantized (Figure 9.8). The spherical harmonic transform has been used for compressing the BRDF [12] in various previous works [3, 26]. The spherical harmonic transform are summarized as follows.

$$C_{l,m} = \int_0^{2\pi} \int_0^\pi P_l(\theta, \phi) B_{l,m}(\theta, \phi) \sin \theta d\theta d\phi, \quad (9.5)$$

where $P_l(\theta, \phi)$'s are the sampled radiance values,

$$B_{l,m}(\theta, \phi) = \begin{cases} N_{l,m} Q_{l,m}(\cos \theta) \cos(m\phi) & \text{if } m > 0 \\ N_{l,0} Q_{l,0}(\cos \theta) / \sqrt{2} & \text{if } m = 0 \\ N_{l,m} Q_{l,m}(\cos \theta) \sin(|m|\phi) & \text{if } m < 0, \end{cases}$$

$$N_{l,m} = \sqrt{\frac{2l+1}{2\pi} \frac{(l-|m|)!}{(l+|m|)!}},$$

and

$$Q_{l,m}(x) = \begin{cases} (1-2m)\sqrt{1-x^2}Q_{m-1,m-1}(x) & \text{if } l = m \\ (2m+1)xQ_{m,m}(x) & \text{if } l = m+1 \\ \frac{2l-1}{l-m}xQ_{l-1,m}(x) - \frac{l+m-1}{l-m}Q_{l-2,m}(x) & \text{otherwise.} \end{cases}$$

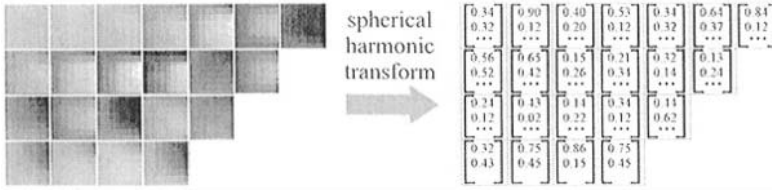


Figure 9.8. Spherical harmonic transform. Each tile is converted to a spherical harmonic vector.

and $Q_{0,0}(x) = 1$.

Coefficients $C_{l,m}$ are the spherical harmonic coefficients to be zonally sampled, quantized and stored. Functions $Q_{l,m}(x)$ are the Legendre polynomials.

Intuitively speaking, the spherical harmonic transform can be regarded as a Fourier transform in the spherical domain. Just like the Fourier transform, the more coefficients are used for representation, the more accurate the reconstructed value is. Figure 9.9 shows the first few harmonics (basis functions $B_{l,m}(\theta, \phi)$). Besides the first basis function (which is a sphere), all other basis functions exhibit directional preferences. Different input functions should exhibit differences in the size of harmonics after transform. Note that basis functions may return negative values. Negative lobes are visualized through "flipping" about the origin in Figure 9.9. In some cases, these negative lobes may coincide with positive lobes.

To reconstruct an interpolated radiance value, the following summation of multiplications is calculated,

$$P_I^*(\theta, \phi) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^l C_{l,m} B_{l,m}(\theta, \phi). \quad (9.6)$$

where $(l_{\max} + 1)^2$ is the number of spherical harmonic coefficients stored.

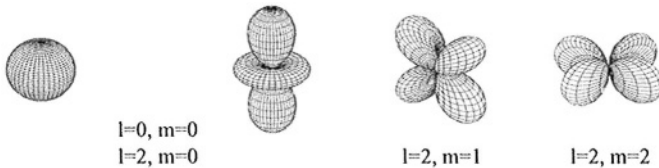


Figure 9.9. Spherical harmonics. Negative lobes are visualized through "flipping" about the origin.

9.6. Inter-Pixel Compression

Up to now, we only utilize the correlation among radiance values received through the same pixel window. The data correlation between adjacent pixels has not yet been exploited. We called it the *inter-pixel* correlation as it refers to the correlation between adjacent abstract pixels.

Since radiance values associating with a pixel are now transformed to a coefficient vector, we have to exploit the correlation between these adjacent coefficient vectors. One may suggest to compress the data using vector quantization (VQ) techniques with the coefficient vectors as input. However, a well-known visual artifact of VQ techniques is *contouring* at the region with smooth changes. Although dithering can partially reduce the objectionable visual artifact, there is an even more difficult problem, large computational expense. It is well known that the computational time of VQ techniques is substantially high when the input data size is large, which is in our case. Moreover, with the same tolerance of reconstruction error, the compression ratio obtained by VQ techniques is usually lower than that of other methods. All these disadvantages prohibit us from using VQ techniques.

Instead, we pick the first coefficients (Figure 9.10) from all coefficient vectors and form a coefficient map. To avoid terminology ambiguity, this kind of coefficient map is called *SH map* (SH stands for Spherical Harmonics) from now on. The same grouping process is applied to the second coefficients and all other coefficients. The result is a set of k SH maps if the coefficient vectors are k -dimensional. Interestingly, each SH map is in fact an image (right hand side of Figure 9.10). These SH maps are somewhat analogous to the eigenimages found by the principal component analysis in the computer vision literatures [2]. This observation suggests us a way to utilize the *inter-pixel correlation*. We can simply treat each SH map as an image of real values and apply standard image compression. To do so, we apply the discrete wavelet transform (DWT).

It seems that different amount of storage (bits) should be allocated to encode different SH maps based on the data statistics. For example, more bits should be assigned to low-order SH maps than other higher-order SH maps. However, based on standard bit allocation methods, low-order SH maps occupy too many bits while high-order SH maps starve (of bits). It is expected as low-order SH maps contribute more energy to the reconstructed images than the high-order SH maps do. As a result, specular highlight and shadow are poorly preserved or even lost since they are represented by high-order SH maps. To preserve these human-sensitive visual features, we assign the same bit rate to encode each SH map in the same color channel to prevent high-order SH maps from starvation.

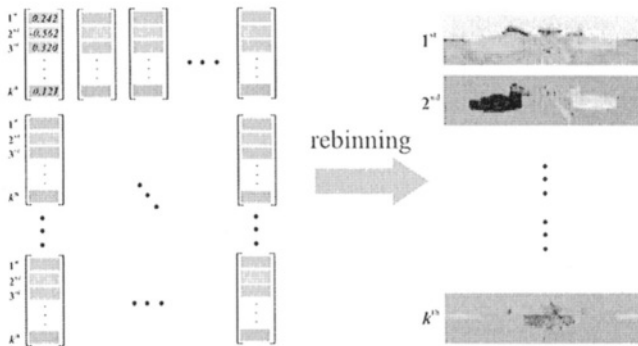


Figure 9.10. Picking the coefficients from the vectors and forming the SH maps.

9.6.1 Discrete Wavelet Transform

The wavelet technique for encoding SH maps described below is very similar to the one for encoding natural images proposed by Joshi *et al.* [11]. Each SH map is decomposed by the 9-7 tap bi-orthogonal wavelet decomposition. Instead of decomposing a SH map into several levels, only two levels of decomposition is performed.

After this decomposition, each SH map is partitioned into 16 uniform bands. One of these band is the lowest frequency subband (LFS) while the other 15 bands are high frequency subbands (HFS). Figure 9.11 shows the 16 resultant subbands. The dark gray region is the LFS while the light gray regions are HFS's. If the size of an input SH map is $x \times y$, the size of each band will be $\frac{x}{4} \times \frac{y}{4}$.

From our observation, the LFS is similar to a downsampled version of the input SH map. In other words, it is also an image. Hence we can apply the common approach for compressing the data in this subband. The LFS

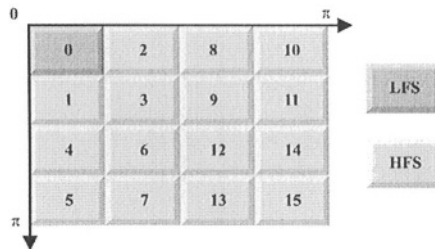


Figure 9.11. After the two-level wavelet decomposition, the coefficients in the frequency domain can be partitioned into 16 subbands.

is first subdivided into 4×4 non-overlapping blocks. The discrete cosine transform (DCT) [22] is then applied to each of these blocks. The DCT itself does not reduce the data volume, the later quantization does. To facilitate the quantization, 16 data subsources are formed (one for each DCT coefficient in the transformed 4×4 block).

On the other hand, a different approach is applied to the 15 HFS's left. It is well known that after the wavelet decomposition, there is only small intra-band correlation among the data within each HFS. Thus we assume that each HFS is a memoryless data source.

Consider a SH map of the size $x \times y$. There will be 16 data subsources from the LFS (after DCT), each subsourse contains $xy/256$ samples. For the HFS's, there will be 15 data subsources (one for each HFS), each subsourse contains $xy/16$ samples.

9.6.2 Quantization and Bit Rate Allocation

Each source is quantized by the generalized Lloyd algorithm [9]. The bit rate of each source is given by [11],

$$R_i = R + 0.5 \log_2(\alpha_i \sigma_i^2) - \left\{ \sum_{j=1}^M 0.5 p_j \log_2(\alpha_j \sigma_j^2) \right\}.$$

where R is the target bit rate; R_i is the encoding rate of the i -th source in bits/sample; p_i is the normalized weight for the source; σ_i^2 is the variance of the source; α_i is a user-defined constant; and M is the number of sources.

Intuitively speaking, more bits should be allocated to a source with a larger variance. Constant α_i depends on the density of source as well as the type of encoding method used. We assume that it is independent of the encoding rate. The DC component of the DCT in the LFS is assumed to have a Gaussian distribution as described in [23]. All other sources are modeled as Laplacian sources. Constant α_i is chosen to be 2.7 and 4.5 for Gaussian and Laplacian sources respectively.

9.7. Inter-Channel Compression

The final compression is done on the wavelength dimension of the plenoptic illumination function. Representing a color in the RGB space is usually wasteful in terms of storage, because the RGB model does not pack important visual signal efficiently. A common color model used in PAL video coding and JPEG standard is YUV. The major advantage is that it packs important visual energy into a single luminous channel Y. The other two channels, U and V, are chrominous channels. Human perception is less sensitive to signal in chrominous channels than that in the luminous channel. Therefore, it is desirable to

reduce the bits allocated for U and V in order to trade for an increase in bits allocated for Y.

A naive approach to adopt the YUV color model is to convert the RGB values of each pixel in the input reference images to YUV before passing the values to the spherical harmonic transform. However, this approach is very inefficient as a matrix multiplication is needed for each radiance value and there may be thousands of reference images.

In fact, the color conversion can be done in the spherical harmonic domain. That is, we can perform color conversion after the spherical harmonic transform. Since there are only a few spherical harmonic coefficients (say 16 to 25) for each pixel, the color conversion can be done efficiently.

The question is whether the color conversion in the spherical harmonic domain is correct. Since the color transform is linear, the correctness of this approach can be proved by the following derivation. Recall spherical harmonic transform in Eq. (9.5).

$$C_{l,m}^Y = \int_0^{2\pi} \int_0^\pi P_l^Y B_{l,m} \sin \theta d\theta d\phi \quad (9.7)$$

where $C_{l,m}^Y$ is the spherical harmonic coefficient for the Y channel; P_l^Y is the radiance value in the Y channel.

The parameter (θ, ϕ) of P_l and $B_{l,m}$ is dropped for simplicity and clarity. From the PAL video standard, we have,

$$Y = \beta_1 R + \beta_2 G + \beta_3 B \quad (9.8)$$

where Y is the value in the Y channel; R, G, B are pixel values in R, G & B channels respectively; $\beta_1 = 0.299, \beta_2 = 0.587$ and $\beta_3 = 0.114$.

Substituting Eq. (9.8) into Eq. (9.7),

$$\begin{aligned} C_{l,m}^Y &= \int_0^{2\pi} \int_0^\pi (\beta_1 P_l^R + \beta_2 P_l^G + \beta_3 P_l^B) B_{l,m} \sin \theta d\theta d\phi \\ &= \beta_1 \int_0^{2\pi} \int_0^\pi P_l^R B_{l,m} \sin \theta d\theta d\phi + \beta_2 \int_0^{2\pi} \int_0^\pi P_l^G B_{l,m} \sin \theta d\theta d\phi \\ &\quad + \beta_3 \int_0^{2\pi} \int_0^\pi P_l^B B_{l,m} \sin \theta d\theta d\phi \\ &= \beta_1 C_{l,m}^R + \beta_2 C_{l,m}^G + \beta_3 C_{l,m}^B \end{aligned}$$

where P_l^R, P_l^G, P_l^B are pixel values in R, G & B channels; $C_{l,m}^R, C_{l,m}^G, C_{l,m}^B$ are spherical harmonic coefficients in R, G & B channels.

Similar derivation can be done for channels U and V. Hence, the color conversion can be applied to spherical harmonic coefficients in the frequency domain instead of the original spatial domain.

9.8. Overall Evaluation

To evaluate the overall performance of the proposed method, we compare it to standard image compression methods, JPEG, JPEG2000, and MPEG. JPEG is a standard compression method for still images. The wavelet-based JPEG2000 [5] is a new image coding standard which performs better than JPEG at low bit rates. However, both JPEG and JPEG2000 exploits no inter-image correlation. Hence we also compare our method to MPEG which is designed for video coding. Two synthetic data sets, 'attic' (Figure 9.5 (a)) and 'forbid' (Figure 9.13 (a)), are tested. The 'attic' data set contains three hundred 1024×256 reference images while the 'forbid' data set contains one thousand and two hundred 1024×256 reference images. The 'attic' data set contains specular objects while the data set 'forbid' contains shadow. Table 9.8 shows the data property of each data set.

When compressing the data sets with our method, we keep the number of spherical harmonic coefficients to be 25 and use the YUV color model. The bit rate allocation for Y:U:V channels is 2:1:1 in ratio. The only parameter to vary is the target bit rate for wavelet compression. Then we reconstruct images from the compressed data at the sampling positions. The control images are the original reference images. The same input image set is passed to JPEG, JPEG2000 and MPEG for compression. Two graphs of PSNR versus bits per pixel are plotted, one for 'attic' (Figure 9.12 (a)) and the other for 'forbid' (Figure 9.12(b)). The "pixel" in the "bits per pixel" refers to the original image pixel (each pixel stores RGB values, 3 bytes in size), not the abstract pixel we referred in the previous sections. The performance curves of our method, JPEG, JPEG2000 and MPEG are plotted in these two graphs.

From the graphs, the proposed method out-performs JPEG, JPEG2000 and MPEG. At the same PSNR level, we achieve a much lower bit rate (or higher compression ratio) than other three coding standards, especially at low bit rates. Note that, since the size of our data size is *always enormous*, it is meaningless to perform well at high bit rates. Consider the case of 'forbid', the total data

Table 9.1. Characteristics of tested data sets.

Data set	Resolution	Sampling rate($\theta \times \phi$)	Real/Synthetic	With Shadow
attic	1024×256	15×20	Synthetic	No
forbid	1024×256	30×40	Synthetic	Yes

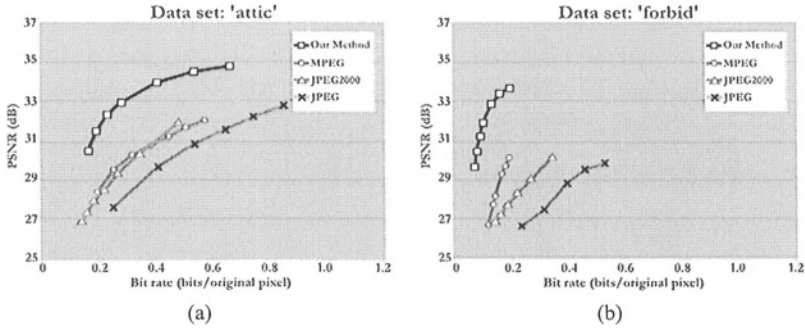


Figure 9.12. Performance comparison with JPEG, JPEG2000 and MPEG. (a) & (b): PSNR vs. Bit per original pixel.

size is 900MB, compressing with 0.3 bit per pixel implies a storage of 90 MB is required. This is still too much for Internet-based applications. Hence, one major criterion for our codec is that it must perform extremely well at low bit rates. From the statistics, our method satisfies such requirement. Using our method, the sizes of data with PSNR of around 30 dB are roughly 1.5 MB and 6 MB for 'attic' and 'forbid' respectively. Obviously, these compressed files can be rapidly transferred through Internet nowadays.

Among the compared methods, JPEG is the worst. JPEG2000 performs better than JPEG, especially at low bit rates. In the case of 'attic', JPEG2000 is comparable to MPEG. However, both JPEG and JPEG2000 are inferior to ours because they do not exploit the correlation among images. As MPEG only exploits the correlation in 1D (time domain), it may not fully utilize the correlation in our data set which is a set of samples on the 2D spherical surface. It might also because MPEG is tailored for object motion in video, but not for change of lighting in our case.

Besides, we also visually compare the images compressed by our method to that of JPEG, JPEG2000 and MPEG. We fix the compression ratio at around 120 and recover an image from the data compressed by four methods. Figure 9.13 shows the comparison. A region in the reconstructed image (Figure 9.13 (a)) is blown up for comparison. The JPEG result (Figure 9.13(c)) contains serious blocky and blurry artifacts. The color tones of some blocks are even distorted. Although the JPEG2000 result (Figure 9.13(d)) is visually better than that of JPEG, it is too blurry that details (e.g. fence) are completely lost. The MPEG one (Figure 9.13(e)) is better but is still contaminated with blocky artifact. Note that the details of fence cannot be clearly observed. The image coded by our method (Figure 9.13(f)) exhibits the minimal visual artifact. Comparing to the original image in Figure 9.13(b), there are some differences near the high contrast region. However the artifact is not apparent.

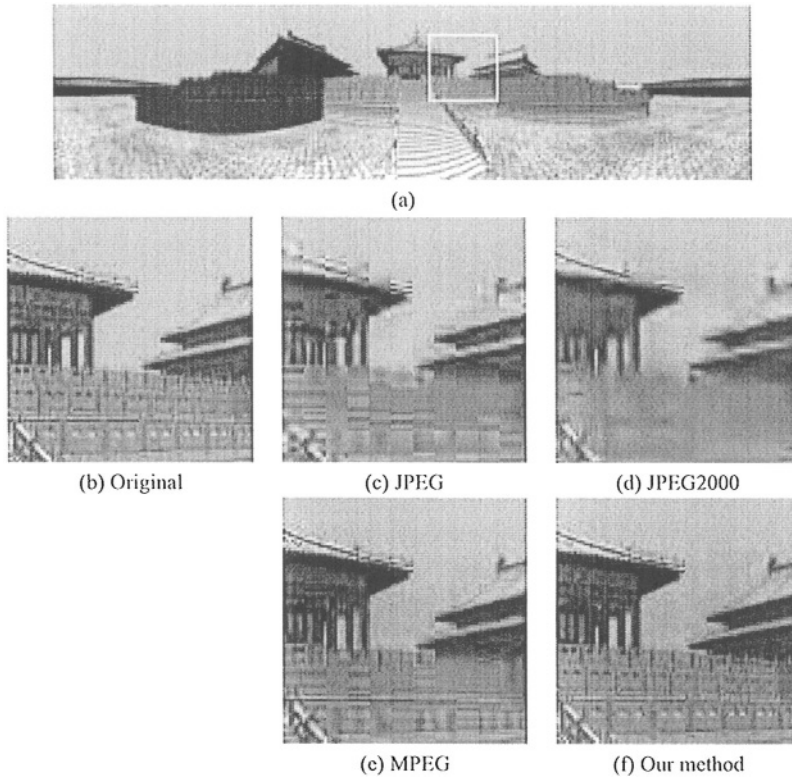


Figure 9.13. Visual comparison of the proposed method, JPEG, JPEG2000 and MPEG.

9.9. Conclusions and Future Directions

In this chapter, we describe a new formulation of the plenoptic function that allows the explicit specification of illumination component. Techniques based on this new model can record and interpolate not just viewing direction and viewpoint, but also the illumination. The core of the proposed model is data compression. We propose a 3-step compression scheme for compressing illumination-adjustable images. A series of compression techniques is applied to exploit the intra-pixel, inter-pixel and inter-channel data correlations. The proposed method out-performs standard image and video coding methods, JPEG, JPEG2000 and MPEG. A 1024×256 illumination-adjustable panoramic image sampled under 1,200 illumination configurations required only a few megabytes of storage. Note that arbitrary lighting configuration can

be specified to relight from this compressed data and the time for relighting is independent of scene complexity.

There is a lot of work to be done in the future. Currently, we only extract the illumination component from the aggregate time parameter t . No other scene changing factors are investigated. If other factors can be extracted, the rigidity of image-based computer graphics can be further relaxed. However, the trade-off is storage requirement.

Another direction to investigate is to further compress the data by exploiting more sophisticated approximation methods of BRDF, such as spherical wavelet [24] and non-linear approximation [13]. If the coefficients recovered by these methods also exhibit strong correlation, inter-pixel compression can also be applied to further reduce the storage. For encoding the SH maps, we believe that coding methods like SPIHT and EBCOT may also be applied after adapting to our data.

Acknowledgments

This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC Earmarked Grants (Project No. CUHK 4189/03E) and RGC Co-operative Research Centres (CRC) Scheme (Project No. CRC 4/98).

Web Availability

A real-time demonstrative panoramic viewer with relighting capability is available through the following web page:

<http://www.cse.cuhk.edu.hk/~ttwong/demo/panoshader/panoshader.html>

References

- [1] E. H. Adelson and J. R. Bergen. "The plenoptic function and the elements of early vision". *Computational Models of Visual Processing*, Chapter 1, pp. 3-20, M. S. Landy and J. A. Movshon Eds., MIT Press, 1991.
- [2] P. N. Belhumeur and D. J. Kriegman. "What is the Set of Images of an object under all possible lighting conditions". *IEEE Conference on Computer Vision and Pattern Recognition*, 1996, sig-11-1994.
- [3] B. Cabral, N. Max and R. Springmeyer. "Bidirectional reflection functions from surface bump maps". *Computer Graphics (SIGGRAPH '87 Proceedings*, vol. 21, no. 4, pp. 273-281, July, 1997.
- [4] J. X. Chai, X. Tong, S. C. Chan and H. Y. Shum. "Plenoptic sampling". *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series*, pp. 307-318, July, 2000.

- [5] M. Charriera, D. S. Cruz and M. Larsson. "JPEG2000, the next millenium compression standard for still images". *Proceedings of the IEEE ICMCS'99*, June, 1999.
- [6] S. E. Chen. "QuickTime VR - an image-based approach to virtual environment navigation". *Computer Graphics Proceedings. Annual Conference Series, SIGGRAPH'95*, pp. 29-38, August, 1995.
- [7] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience Publisher, Inc., New York, 1953.
- [8] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen. "The lumigraph". *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH'96*, pp. 43-54, August, 1996.
- [9] R. M. Gray. *Source Coding Theory*, Kluwer Academic, 1990.
- [10] P. M. Ho, T. T. Wong, K. H. Choy and C. S. Leung. "PCA-based compression for image-based relighting". *Proceedings of IEEE International Conference on Multimedia & Expo 2003 (ICME 2003)*, Baltimore, USA, July, 2003, to appear.
- [11] R. L. Joshi, H. Jafarkani, J. H. Kasner, T. R. Fischer, N. Farvardin, M. W. Marcellin and R. H. Bamberger. "Comparison of different methods of classification in subband coding of images". *IEEE Transaction on Image Processing*, vol. 6, pp. 1473-1486, 1997.
- [12] J. T. Kajiya. "Anisotropic reflection models". *Computer Graphics (SIGGRAPH '85 Proceedings)*, vol. 19, no. 3, pp. 15-21, July, 1985.
- [13] E. P. F. Lafortune, S. C. Foo, K. E. Torrance and D. P. Greenberg. "Non-linear approximation of reflectance functions". *Computer Graphics Proceedings, Annual Conference Series 1997*, pp. 117-126, August, 1997.
- [14] C. S. Leung, P. M. Lam and T. T. Wong. "Data compression for a massive image data set in IBMR". *Proceedings of Computer Graphics International 2003 (CGI 2003)*, Tokyo, Japan, July, 2003, to appear.
- [15] M. Levoy and P. Hanrahan. "Light field rendering". *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH '96*, pp 31-42, August, 1996.
- [16] Z. Lin and T. T. Wong and H. Y. Shum. "Relighting with the reflected irradiance field: representation sampling and reconstruction". *International Journal of Computer Vision*, vol. 49, no. 2-3, pp. 229-246, September-October, 2002.
- [17] D. Lischinski, F. Tampieri and D. P. Greenberg. "Discontinuity meshing for accurate radiosity". *IEEE Computer Graphics and Applications*, vol. 12, no. 6, pp.25-39, Nov, 1992.
- [18] L. McMillan and G. Bishop. "Plenoptic modeling: An image-based rendering system". *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH'95*, pp. 39-46, August, 1995.
- [19] S. K. Nayar and H. Murase. "Dimensionlity of illumination in appearance matching", *IEEE International Conference on Robotics and Automation*, pp. 1326-1332, April, 1996.
- [20] J. S. Nimeroff, E. Simoncelli and J. Dorsey. "Efficient re-rendering of naturally illuminated environments". *Fifth Eurographics Workshop on Rendering* pp. 359-373, Darmstadt, Germany, June, 1994, sig-11-1994.
- [21] K. Nishino, Y. Sato and K. Ikeuchi. "Eigen-texture method: Appearance compression based on 3D model". *Proceedings of IEEE Computer Vision and Pattern Recognition Conference 99*, pp. 618-624, June, 1999.
- [22] K. R. Rao. *Discrete Cosine Transform, Algorithms, Advantages and Applications*, Academic Press, 1990.

- [23] R.C. Reininger and J.D. Gibson. "Distribution of the two-dimensional DCT coefficients for images". *IEEE Transactions on Communications*, vol. 31, pp. 835-839, 1983.
- [24] P. Schröder and W. Sweldens. "Spherical wavelets: efficiently representing functions on the sphere". *Annual Conference Series, SIGGRAPH 95 Conference Proceedings*, pp. 161-172, August, 1995.
- [25] H. Y. Shum and L. W. He. "Rendering with concentric mosaics". *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH'99*, pp. 299-306, August, 1999.
- [26] F. X. Sillion, J. R. Arvo, S. H. Westin, Donald P. Greenberg. "A global illumination solution for general reflectance distributions". *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol. 25, no.4, pp. 187-196, July, 1991.
- [27] T. T. Wong. "Image-based relighting as the sampling and reconstruction of the plenoptic illumination function". *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, vol. IV, pp. 764-767, Hong Kong, April, 2003.
- [28] C. W. Fu, T. T. Wong and P. A. Heng. "Interactive relighting of panoramas". *IEEE Computer Graphics & Applications*, vol. 21, no. 2, pp. 32-41, March-April, 2001.
- [29] T. T. Wong, C. W. Fu, P. A. Heng and Chi-Sing Leung. "The plenoptic illumination function". *IEEE Transactions on Multimedia*, vol. 4, no. 3, pp. 361-371, September, 2001.
- [30] T. T. Wong, P. A. Heng, S. H. Or and W. Y. Ng. "Image-based rendering with controllable illumination". *Eighth Eurographics Workshop on Rendering*, pp. 13-22, Saint Etienne, France, June, 1997.
- [31] T. T. Wong and C. S. Leung. "Compression of illumination-adjustable images". *IEEE Transactions on Circuits and Systems for Video Technology (special issue on Image-based Modeling, Rendering and Animation)*, 2003, to appear.
- [32] T. T. Wong, S. H. Or and C. W. Fu. "Real-time relighting of compressed panoramas". *Graphics Programming Methods*, J. Lander Ed., Charles Rivers Media, 2003, to appear.

Chapter 10

CONSTRUCTION OF COMPLEX ENVIRONMENTS FROM A SET OF DEPTH IMAGES

Enhua Wu^{1,2}, Yanci Zhang² and Xuehui Liu²

1. Department of Computer and Information Science, University of Macau

2. Lab of Computer Science, Institute of Software, Chinese Academy of Sciences

ehwu@umac.mo, weh@ios.ac.cn, v-ycizhang@msrchina.research.microsoft.com,

lxh@ios.ac.cn

Abstract In this chapter, we discuss the research issues and state-of-the-art of construction of complex environments from a set of depth images. A newly proposed automatic construction algorithm is mainly introduced in this chapter. The algorithm proposes a hybrid representation of complex models by a combination of points and polygons. Through the model, a real time walkthrough of a complex scene can be achieved. By the approach, starting from a set of depth reference images, all pixels of the images are classified into two categories, corresponding respectively to the planar and non-planar surfaces in the scene and then all redundant pixels are eliminated according to their sampling rate. For the pixels corresponding to non-planar surfaces, a local reconstruction and resampling process is employed to generate a set of new samples, organized by OBB-tree. For the pixels corresponding to planar surfaces, their corresponding textures are reconstructed and holes in textures are analyzed and filled in pre-processing stage. Under this hybrid representation, a culling algorithm can be employed to greatly improve rendering efficiency and a real time walkthrough of a complex scene can be achieved with no restriction on user's motion.

Keywords: Depth image, automatic object modeling, sampling rate, point-based representation

10.1. Introduction

A typical application of virtual reality is to allow people to freely walk through a complex environment in interactive speed. The traditional computer graphics based on geometry models has made great effort to achieve the goal for decades

and the realism of images and rendering efficiency have been greatly improved. However, it is still quite difficult and time consuming to create models of complex objects. Sophisticated realistic rendering algorithms such as ray tracing and radiosity are hard to meet the requirement of rendering complex environment in interactive speed.

Recently, a significant trend in computer graphics has been using 3D scanners and high resolution digital cameras to generate depth images for complex objects or a complex scene [4, 11, 13, 14, 17], and then directly adopt the samples from depth images to represent the objects or environment. In this chapter, we will introduce a new method of complex environment construction from depth images, that may bypass the time-consuming modeling process in traditional computer graphics and the sophisticated shading calculation.

For the method proposed, an automatic construction algorithm for complex environments from depth images will be introduced. In the method, a complex scene is sampled by a set of depth images produced from arbitrary viewpoints. A hybrid representation of the scene is built up by classifying the objects into non-planar and planar surfaces represented respectively as points and textured polygons. Under this hybrid representation, a real time walkthrough in the complex environment can be achieved.

The remainder of the chapter is organized as follows. First, some typical algorithms on the construction of complex environments from depth images are introduced in Section 10.2. Then the automatic construction algorithm in combining polygons and points is introduced from Section 10.3 to Section 10.6. Finally conclusions are drawn in the last section.

10.2. Typical Algorithms

First, let's review some of the previous works on the model construction from depth images. Basically there are two common ways in which a complex environment is constructed from depth images. One way is to build 3D geometry models from depth images and use the models reconstructed to represent the environment [5, 18, 19]. The depth images only play an intermediate role in this method and the actual rendering input is the 3D geometry models reconstructed. From this point of view, it is actually a traditional computer graphics method.

Another way is to directly adopt the samples from depth images to represent the environment without reconstruction of its geometry models. The approach in this method is totally different from that in traditional computer graphics since the rendering primitive in this approach is the samples from depth images which contain color and depth information instead of geometry models.

The algorithm proposed in Layered Depth Image (LDI)[2, 22] is the most typical algorithm where the samples from depth images are directly taken as

rendering primitive. The algorithm is based on the 3D warping equation [12] proposed by McMillan. As a matter of fact, the method in LDI actually proposed an efficient organization method of samples, which merges all the samples from depth images under a single center of projection. It keeps multiple depth pixels for a location while still maintaining the simplicity of warping a single image.

The Multiple-Center-of-Projection Images (MCOP Images) proposed by Rademacher [21] correctly reconstruct 3D scenes where every pixel in the source image was potentially acquired from a different but known position. MCOP images have the advantage that objects can essentially be scanned, but the data is still a single image. If a strip camera is the acquisition model, the pose information is only required for each column of data.

Oliveira presents a compact, image-based representation for three-dimensional objects named Image-based Object (IBO) [15]. Objects are represented by six LDIs [22] sharing a single center of projection. They can be scaled, and freely translated and rotated, being used as primitives to construct more complex scenes.

10.3. Framework of Hybrid Modeling

Although direct application of samples from depth images as rendering primitive may prevent from the time-consuming modeling process and sophisticated shading calculation in rendering, attention must be paid to some important issues when interactive rendering speed is required:

- **Efficient Organization of Samples.** For a complex environment, hundreds of depth images may be required to cover all the parts and sufficient details of the environment. An efficient and compact mechanism should be elaborately designed to organize the huge set of samples from the depth images.
- **Elimination of Redundant Samples.** The same surface may appear in different images and the multiple appearances must contain large amount of redundant information. The redundant samples should be removed to guarantee a compact representation of the environments.
- **Ability to Fill Holes.** For a complex scene, it commonly happens that some parts of the scene are not captured by all the images and this may cause holes in rendering. This kind of hole should be filled as possible as we can in the process of constructing the environment instead of in rendering stage.

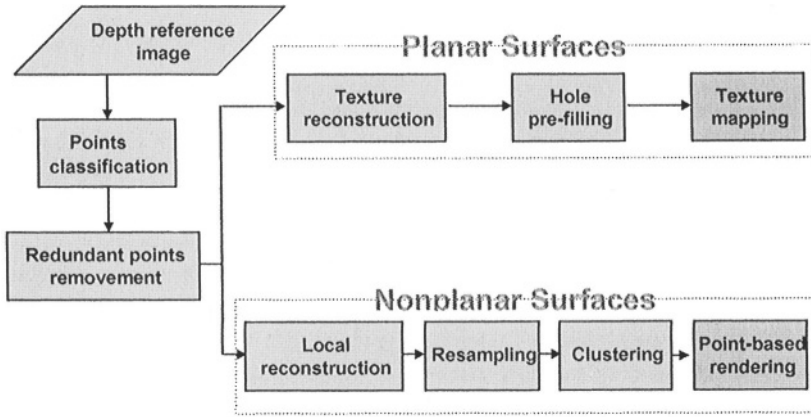


Figure 10.1. Algorithm overview.

Zhang et al. [24, 25, 26] proposed a hybrid modeling technique to tackle the problems above. The algorithm is based on the fact that point-based representation works better for objects with rich, organic shapes or high surface details than traditional polygon-based representation. On the other hand, polygon representation is more suitable for large flat surfaces than point-based representation. Consequently, a hybrid representation is built up by assigning different representations to their corresponding categories of objects in the environment.

This algorithm contains two main stages, preprocessing and rendering. The outline of algorithm is as shown in Figure 10.1. In the pre-processing stage, all samples from the depth images will be classified into two categories, planar and non-planar surfaces, and redundant samples will be eliminated. For the planar surfaces, polygons and their corresponding textures will be reconstructed, and holes will be analyzed and filled. For the non-planar objects represented as point primitive, a set of new samples will be generated and the new samples will be organized in OBB-tree with normal vector clusters. In the rendering stage, a real time walkthrough system for the complex environment based on the hybrid representation is preformed.

10.4. Pick up of Valid Samples

The fundamental of hybrid modeling is to classify the original samples/pixels into two categories, planar and non-planar pixels, respectively suitable for polygon-based and point-based representations. Then a hybrid representation may

be built up on the classification results. Moreover, redundant samples should be eliminated to guarantee a compact representation of the scene.

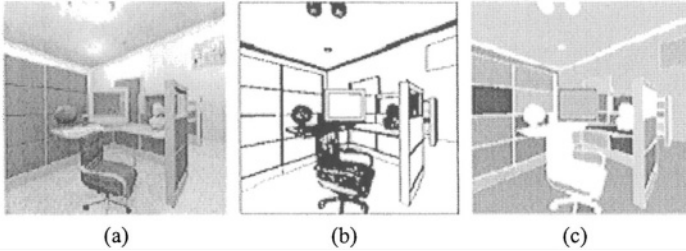


Figure 10.2. Pixel Classification. (a) One reference image. (b) Pixel regions in white are planar regions extracted from image. (c) Planar surfaces reconstructed via least-square method.

10.4.1 Sample Classification

Generally speaking, the classification work can be treated as the problem of extracting planar surfaces from the depth images (as shown in Figure 10.2). Edge detection in depth space or some other algorithms [10] can be employed to solve this problem. After planar regions are detected in success, a least-square method is used to verify that the regions segmented really correspond to planar surfaces in the scene.

Different planes reconstructed from different images, even from the same image, may correspond to the same planar surface in a scene. These planes should be regarded as identical and should be merged into one surface. The merge conditions are defined by planes' equations reconstructed in the step above with the bounding boxes of the surfaces. Two planes are taken as the same planar surface in a scene if they have similar equation and their bounding boxes are close enough.

10.4.2 Redundancy Elimination

In the algorithm, pixels sampled at lower frequencies are regarded as redundant information. The determinant of the Jacobi matrix is used as the measurement of the sampling rate among different reference images and the determinant calculation is widely used to measure the local area change introduced by a mapping.

Generally speaking, the sampling rate of pixel p in a reference image R depends on three factors: 1) distance between p and the view point of R ; 2) angle between normal vector of p and the view direction; 3) resolution of R .

Normal vector of p is the only unknown factor, so we have to estimate it before the sampling rate comparison could be made. For a planar surface point, their

normal can be directly derived from their corresponding plane equation. For a non-planar surface point, a calculation by normal estimation algorithm [6] could be made.

The sampling rate comparison is employed between two pixels that come from different reference images and correspond to the same region in the scene. Because all the reference images have the same resolution, we can use pixels' window coordinates to replace their pixel coordinates to calculate Jacobi matrix.

The map between two pixels m (x_m, y_m, z_m, l) and n (x_n, y_n, z_n, l) can be expressed as Eq. (10.1), where M_1 and M_2 are transformation matrix which maps the world coordinate to the window coordinates of m, n respectively.

$$\frac{w_n}{w_m} \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} = M_2 M_1^{-1} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} wx_n \\ wy_n \\ wz_n \\ w \end{bmatrix} = M \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix} \quad (10.1)$$

In fact, we only need to focus on the X, Y coordinates and Eq. (10.1) is actually a 2D mapping because z_m can be represented in x_m and y_m by the expression:

$$z_m = \begin{pmatrix} -\frac{a}{c} & -\frac{b}{c} & -\frac{d}{c} \end{pmatrix} \begin{pmatrix} x_m \\ y_m \\ 1 \end{pmatrix} \quad (10.2)$$

where $\begin{pmatrix} a & b & c & d \end{pmatrix} = \begin{pmatrix} A & B & C & D \end{pmatrix} M_1^{-1}$

$$\begin{aligned} \begin{bmatrix} wx_n \\ wy_n \\ w \end{bmatrix} &= \begin{bmatrix} m_0 & m_4 & m_{12} \\ m_1 & m_5 & m_{13} \\ m_3 & m_7 & m_{15} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} + \begin{bmatrix} m_8 \\ m_9 \\ m_{11} \end{bmatrix} z_m \\ &= \begin{bmatrix} m_0 - \frac{a}{c} m_8 & m_4 - \frac{b}{c} m_8 & m_{12} - \frac{d}{c} m_8 \\ m_1 - \frac{a}{c} m_9 & m_5 - \frac{b}{c} m_9 & m_{13} - \frac{d}{c} m_9 \\ m_3 - \frac{a}{c} m_{11} & m_7 - \frac{b}{c} m_{11} & m_{15} - \frac{d}{c} m_{11} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} \end{aligned}$$

$$= M' \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix}$$

According to Eq. (10.1) and Eq. (10.2), the 2D mapping can be expressed as:
Extracting Jacobi matrix according to the above equation by:

$$|J| = \begin{vmatrix} \frac{\partial x_n}{\partial x_m} & \frac{\partial x_n}{\partial y_m} \\ \frac{\partial y_n}{\partial x_m} & \frac{\partial y_n}{\partial y_m} \end{vmatrix} = \frac{|M'|}{w^3} \quad (10.3)$$

The sampling rate of m , n may be compared according to Eq. (10.3): if $|J| > 1$, the sampling frequencies of n are higher than m , then m is abandoned; otherwise, m has a higher sampling frequencies than n , m is reserved.

10.5. Hybrid Representation

10.5.1 Polygon-based Representation

After sampling rate comparison, a single planar surface in the scene may be split into several parts which appear in different reference images and each part represents the region best sampled among all the images (as shown in Figure 10.3 (b)). In order to obtain a more efficient and packed representation, all these parts should be merged to create a single texture (as shown in Figure 10.3 (c)). This work can be done by mapping and splatting all the points best sampled which belong to the planar surface by an EWA filter [9] to the object space of the surface.

For a complex scene, it is hard to capture all the information of the scene by images. Therefore the appearance of holes in textures reconstructed is almost inevitable. The most common method to fill holes is to search from other reference images or interpolate from neighbor pixels in rendered image. However, these operations tend to reduce the rendering efficiency and the result produced is often unsatisfactory and unsteady.

A hole pre-filling algorithm is proposed to overcome these drawbacks in the pre-processing stage. The key task is to distinguish the holes caused by image capture from those that do not belong to the plane reconstructed itself. As illustrated in Figure 10.4 (b), not all the hole pixels (in black color) in a texture of the desktop belong to the polygon itself. In fact, as shown in Figure 10.4 (c), pixels in blue belong to the reconstructed polygon while pixels in red do not. If the

corresponding space region of these red pixels is filled, the region wrongly filled would occlude the objects behind.

In the solution, the distinguish algorithm is based on a fact that if any hole point is caused by image capturing, then for any reference image from which we may see this hole point, there should be a point in the image to occlude it. In other words, a point s is regarded as hole caused by image capturing if no farther point that locates in the same ray from viewpoint to s can be found from all the reference images. By backward warping the corresponding 3D points of hole pixel to all the reference images, we can solve this distinguish problem.

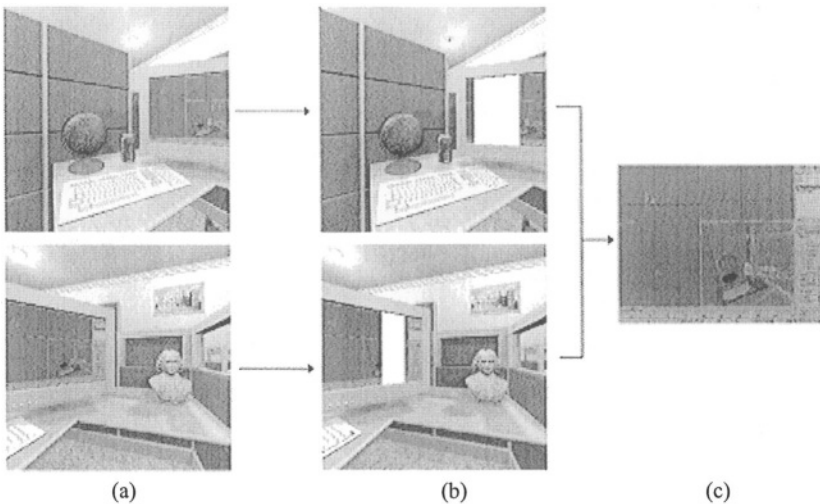


Figure 10.3. Texture reconstruction from two images after sampling rate comparison. (a) Two reference images. (b) Images after sampling rate comparison. Pixels in white are tagged as redundant information in sampling rate comparison. (c) Texture extracted from the parts best sampled from two images.

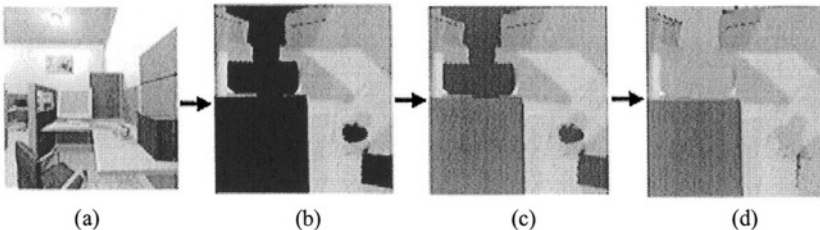


Figure 10.4. Hole filling in preprocess stage. (a) Reference image which contains the desktop. (b) Corresponding texture of desktop reconstructed from images, pixels which are not assigned color in texture reconstruction are tagged in black. (c) Hole pixels should be

classified before filled. Pixels in blue are fillable while pixels in red are unfillable. (d) Texture after hole filling.

10.5.2 Point-based Representation

In fact, the original points directly taken from the reference images are not suitable for rendering because their sampling rate may vary greatly so as to lead to two problems:

- Different sampling rate always requires different splatting size in walkthrough, but it is unrealistic to calculate splatting size point by point;
- It is difficult to build multi-resolutions based on a set of points in different sampling rate.

To solve the problems we have to conduct resampling work to generate a set of new points that distribute more uniformly on the object's surface from the original points. Based on these new samples, a cluster algorithm can be employed and a multi-resolution framework can be built up.

In the resampling work, a local reconstruction and resample scheme is employed. In the local reconstruction, the original samples are treated as the vertices of a triangle mesh. After the local reconstruction, all the triangles reconstructed are resampled in one direction that is aligned to the coordinate axes on a surrounding bounding box. One axis-aligned direction is chosen as resampling direction if it has the smallest angle between itself and the normal vector of triangle. In comparison with other sampling method [16], this resampling method can generate the samples more uniformly distributed on the surfaces. The two-dimension illustration of the resampling strategy is as shown in Figure 10.5.

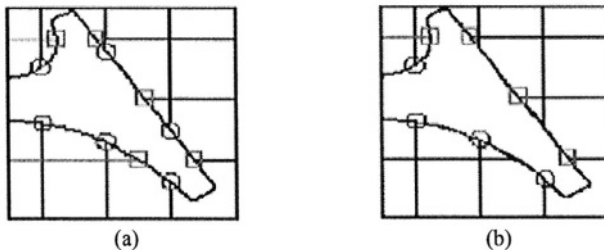


Figure 10.5. (a) Sampling from all directions [16]. (b) Sampling from only one direction [24, 26].

In order to improve rendering efficiency, it is desirable to be able to switch

Finally the new samples will be organized as the structure shown in Figure 10.6. More details on the solution may be seen from [24, 26].

Points in same normal vector cluster (*N-cluster* for short) and resolution level have almost the same normal vector, sampling rate and 3D position. Consequently, we can assign to these points almost the same splat size in rendering.

10.6. Real Time Rendering

In rendering phase, the polygons reconstructed can be easily rendered through texture mapping. However, for rendering the point-based objects, we encounter two challenges. The first is how to cull the invisible points to accelerate rendering, and the second is how to render points efficiently.

10.6.1 Culling

To tackle the first challenge, visibility culling is performed in traversing the OBB-trees during walkthrough. If an OBB node is culled, we stop traverse its children nodes and cull it including its entire offspring.

A visibility culling includes three aspects:

View-frustum Culling. View-frustum culling can be easily done based on OBB node. Since no 3D models are available, we only cull the OBB node which is fully outside the current view-frustum. Various algorithms [1] can be employed to fulfill this work.

Back-faced Points Culling. Back-faced points culling is based on *N-cluster* structure. If the average normal vector of an *N-Cluster* is back-faced, all points in the *N-cluster* are culled.

Occlusion Culling. There are two ways to solve this problem. The first solution is to cull OBB nodes which are occluded by polygons reconstructed. A reduced version of the visibility culling technique in [3] is employed to fulfill the task. Another solution to conduct the occlusion culling is to use OpenGL extension *NV_occlusion_query*. Comparing with the first method, *NV_occlusion_query* may cull more points since it may cull the OBB nodes which are occluded both by other OBB nodes and by the polygons reconstructed. In order to use *NV_occlusion_query*, the polygons reconstructed should be rendered first and a visible object list which is sorted by the distance between individual object and current view point should be maintained. Based on this visible object list, all visible objects are rendered from near to far so that the distant OBB nodes may be culled by the nearer OBB nodes rendered.

10.6.2 Rendering Issues

In order to achieve the real time rendering, the following issues are considered:

Cluster-based Splatting. As previously mentioned, it is inefficient to calculate the splat size for individual points. The strategy adopted is to calculate the center point at each resolution level in visible *N-Clusters* and use this point to determine the splat size for all the points in the same *N-Cluster* and resolution level. Because the points in the same *N-Cluster* of individual resolution level have almost the same sampling rate and normal vectors, the splat size of the center point may accurately approximate the splat size of all the other points.

Choosing Proper Level. Before splatting, the splat size at each resolution level in the same *N-Cluster* is calculated and the resolution which more closely matches the output display resolution is chosen as the proper rendering level under current viewpoint.

Splatting. If higher rendering quality is preferred, the hardware assisted splatting algorithm, such as that introduced in [20], can be employed to fulfill the splatting. On the other hand, if higher rendering efficiency is preferred, we can directly use OpenGL *GL_POINTS* and *glPointSize* as a coarse approximation of the splatting.

Antialiasing. In order to alleviate alias caused by using *GL_POINTS* and *glPointSize* to approximate splatting or some other reasons, the OpenGL extension *WGL_ARB_multisample*, used in conjunction with *NV_multisample_filter_hint*, can be employed to enable high resolution antialiasing.

Frame-to-frame Coherence. Frame-to-frame coherence can be utilized to improve rendering efficiency. For example, once a splat size is calculated, it may keep valid in the following *N* frames; if an OBB node is regarded as a visible node, it remains visible in the following *N* frames without further check required.

OpenGL Issues. Combining *glDrawArrays* and *ARB_vertex_buffer_object* can achieve a much faster rendering speed than immediate mode. An entire array of points can be passed to OpenGL in one call by *glDrawArrays*, and the vertex data can be cached in a high-performance graphics memory to increase the rate of data transfers via *ARB_vertex_buffer_object*.

10.6.3 Rendering Results

The algorithm is tested on a PC with AMD 1GHz CPU, 256M RAM and Geforce4 Ti-4200 display card. 100 original reference images with resolution 500X500 which need about 170M to store color and depth information are taken as input. After hybrid modeling, 94 textured polygons are picked from images and

about 5.16M new point samples in four different resolutions are generated. The final storage requirement for the scene is about 106.1M, including 23.6M for textures and 82.5M for point samples. The storage requirement of hybrid representation is only about 62% of that for the original images, and this ratio may be further reduced if more reference images were employed because more images always mean more redundancy. As for rendering efficiency, at least 17 fps can be guaranteed in the walkthrough.

Four images in walkthrough are shown as in Figure 10.7.

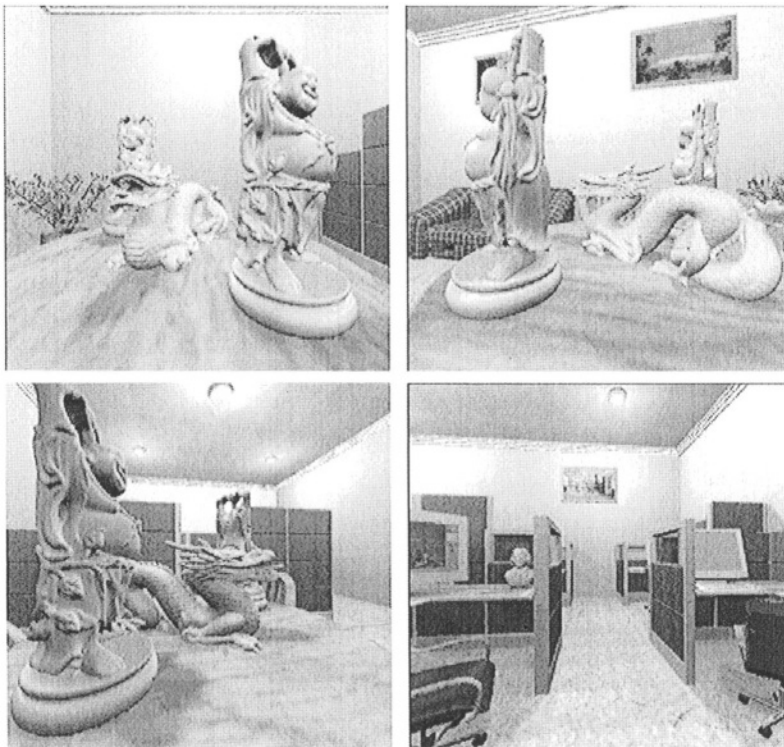


Figure 10.7. Rendering results.

10.7. Summary and Conclusions

In this chapter, we introduce a hybrid modeling technique for complex environment from depth images. To represent the complex objects in an environment, we introduce a hybrid representation by combining points and polygon models. From a set of depth images, the original points are automatically

classified into two categories that correspond to planar and non-planar surfaces respectively in 3D space. Textured polygons are reconstructed from planar surfaces while for non-planar surface points, a local reconstruction and resampling process is employed to generate a set of new point samples. In order to improve the rendering efficiency, these new points resampled are organized by OBB-tree. At the same time, a sampling rate comparison algorithm is carried out to remove the redundant information among reference images so as to obtain a compact representation of the scene. Under this hybrid representation, we can fulfill a real time walkthrough in a complex environment without any restriction to user's motion.

Acknowledgments

The work described in this chapter has been supported by 973 Fundamental Research Fund of China (Grant No. 2002CB312102), NSFC Projects and Research Grant of University of Macau.

References

- [1] U. Assarsson, T. Moller. "Optimized view frustum culling algorithms for bounding boxes". *Journal of Graphics Tools*, vol. 5, no. 1, pp. 9-22, 2000.
- [2] C. F. Chang, G. Bishop, A. Lastra. "LDI tree: a hierarchical representation for image-based rendering". *Proceedings of SIGGRAPH 1999*, pp. 291-298.
- [3] S. Coorg, S. Teller. "Real-time occlusion culling for models with large occluders". *Symposium on Interactive 3D Graphics*. 1997, pp. 83-90.
- [4] W. T. Correa, S. Fleishman, C. T. Silva. "Towards point-based acquisition and rendering of large real-world environments". *XV Brazilian Symposium on Computer Graphics and Image Processing*, 2002, pp. 59-68.
- [5] B. Curless and M. Levoy. "A volumetric method for building complex models from range images". *Proceedings of SIGGRAPH 1996*, pp. 303-312.
- [6] M. Gopi, S. Krishnan and C. T. Silva. "Surface reconstruction based on lower dimensional localized delaunay triangulation". *Proceedings of Eurographics 2000*, Vol. 19, no. 3, pp. C467-C478.
- [7] S. Gottschalk, M. C. Lin, D. Manocha, OBB Tree: "A hierarchical structure for rapid interference detection". *Proceedings of SIGGRAPH 1996*, pp. 171-180.
- [8] S. Gottschalk, "Collision queries using oriented bounding boxes". PhD Thesis. Department of Computer Science, UNC Chapel Hill, 2000.
- [9] P. Heckbert. "Fundamentals of texture mapping and image warping". Master's thesis, University of California at Berkeley, Department of Electrical Engineering and Computer Science, 1989.
- [10] X. Y. Jiang, H. Bunke. "Fast segmentation of range images into planar regions by scan line grouping", Technical report IAM 92-006, Institute for Computer Science, University of Bern, Switzerland, April 1992.

- [11] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, C. McCue. "Real-time rendering of real world environment". *Proceedings of the Eurographics Workshop on Rendering* (Rendering Techniques '99), 1999, pp. 145-160.
- [12] L. McMillan. "An image-based approach to three-dimensional computer graphics". PhD thesis of University of North Carolina, 1997.
- [13] L. Nyland, D. McAllister, V. Popescu, C. McCue, A. Lastra, P. Rademacher, M. Oliveira, G. Bishop, G. Meenakshisundaram, M. Cutts, and H. Fuchs. "The impact of dense range data on computer graphics". *IEEE Workshop on Multi-View Modelling and Analysis of Visual Scenes*, 1999, pp. 3-10.
- [14] L. Nyland, A. Lastra, D. K. McAllister, V. Popescu, C. McCue. "Capturing, processing and rendering real-world scenes". *Videometrics and Optical Methods for 3D Shape Measurement*, Electronic Imaging 2001, Photonics West, SPIE, vol. 4309, 2001.
- [15] N. Oliveira, M. Manuel. and G. Bishop. "Image-based objects". *Proceedings of 1999 ACM Symposium on Interactive 3D Graphics*, 1999, pp. 191-198.
- [16] H. Pfister, M. Zwicker, J. van Baar, and M Gross. "Surfels: surface elements as rendering primitives". *Proceedings of SIGGRAPH 2000*, pp. 335-342.
- [17] P. Poulin, M. Stamminger, F. Duranleau, M. C. Frasson, G. Drettakis. "Interactive point-based modeling of complex objects from images". *Proceedings of Graphics Interface 2003*, June 2003.
- [18] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, W. Stuetzle. "View-based rendering: visualizing real objects from scanned range and color data". Technical Report UW-CSE-97-04-01, University of Washington.
- [19] K. Pulli. "Surface reconstruction and display from range and color data". Ph.D. dissertation, University of Washington, 1997.
- [20] L. Ren, H. Pfister, M. Zwicker. "Object space EWA surface splatting: a hardware accelerated approach to high quality point rendering". *Computer Graphics Forum*, vol. 21, no. 3, pp. 461-470, 2002.
- [21] P. Rademacher and G. Bishop. "Multiple-center-of-projection images". *Proceedings of SIGGRAPH 1998*, pp. 199-206.
- [22] J. Shade, S. J. Gortler, L. He, and R. Szeliski. "Layered depth images". *Proceedings of SIGGRAPH 1998*, pp. 231-242.
- [23] L. Y. Wei, M. Levoy. "Fast texture synthesis using tree-structured vector quantization". *Proceedings of SIGGRAPH 2000*, pp. 479-488.
- [24] Enhua Wu, Yanci Zhang, Xuehui Liu. "Point representation of geometry for irregular objects in image-based VR". *The 15th International Conference on Computer Animation (Computer Animation '02)*, June, 2002, IEEE Press, Geneva, Switzerland, pp. 55-64.
- [25] Enhua Wu. "Challenge of highly detail object modeling and illumination calculation in virtual reality". Keynote Speech at ACM VRST2002, Nov. 11-13, 2002, Hong Kong, China.
- [26] Yanci Zhang, Xuehui Liu and Enhua Wu. "A hybrid representation of environment models in image-based real time walkthrough". *The 10th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2002)*, Oct. 2002, IEEE Computer Society Press, Beijing, China, pp. 335-345.

This page intentionally left blank

Chapter 11

QUADRATIC SURFACE RECONSTRUCTION FROM MULTIPLE VIEWS USING SQP

Rubin Gong and Gang Xu

Department of Computer Science, Ritsumeikan University, Kusatsu,

Shiga 525-8577, Japan

{gong,xu}@cv.cs.ritsumei.ac.jp

Abstract

We propose using SQP (Sequential Quadratic Programming) to directly recover 3D quadratic surface parameters from multiple views. A surface equation is used as a constraint. In addition to the sum of squared reprojection errors defined in the traditional bundle adjustment, a Lagrangian term is added to force recovered points to satisfy the constraint. The minimization is realized by SQP. Our algorithm has three advantages. First, given corresponding features in multiple views, the SQP implementation can directly recover the quadratic surface parameters optimally instead of a collection of isolated 3D points coordinates. Second, the specified constraints are strictly satisfied and the camera parameters and 3D coordinates of points can be determined more accurately than that by unconstrained methods. Third, the recovered quadratic surface model can be represented by a much smaller number of parameters instead of point clouds and triangular patches. Experiments with both synthetic and real images show the power of this approach.

Keywords: Quadratic surface reconstruction, constrained minimization, Sequential Quadratic Programming, bundle adjustment, error analysis

11.1. Introduction

Bundle adjustment is usually used in recovering 3D structures and camera intrinsic/extrinsic parameters from a given sequence of images. It works to refine the recovery in an optimal way and thus obtain a more accurate solution. Traditional bundle adjustment aims at recovering isolated 3D features using nonlinear unconstrained optimization methods. Since it does not rely on

relations between isolated features, a very wide variety of scenarios can be handled. However, in practice, scenes often contain some prior 3D constraints, such as 3D distances and planar constraints. If applied carefully, more accurate 3D scene structure and camera parameters can be recovered [2,5]. Furthermore, in many scenes such as indoor scenes and man-made objects, scenes often contain structures with strong geometry regularities such as floors, walls and globe. And it is more suitable to use parameterized models than isolated features to represent these objects [2,3,4,5]. Some previous work has already noted this. For example, G. Cross et al. proposed recovering quadric surfaces from multiple views in [3]. Ying Shan et al. utilized the point-on-surface constraint in their model-based bundle adjustment method to directly recover face model from multiple views [2].

When scene constraints are incorporated into bundle adjustment, nonlinear constrained minimization methods are needed to minimize the objective cost function while keeping the specified constraints strictly satisfied. Previous work has given their own nonlinear constrained minimization methods to incorporate constraints in bundle adjustment. For example, the work in [2] has used a kind of penalty method that converts constrained minimization problem into an unconstrained one. The work in [4] has described a scheme for incorporating surface and other scene constraints into a VSDF filter to directly recover the surfaces and camera motion.

In this chapter Sequential Quadratic Programming (SQP) [1,6,7,8,9,10] is used to incorporate scene constraints in bundle adjustment to directly recover quadratic surface parameters from multiple views. SQP is a powerful constrained minimization method and has been successfully applied in a wide variety of industrial fields. It has a concise mathematical formulation and can incorporate a wide variety of constraints. Triggs has used it in camera calibration [10]. However, it has seldom been used in surface reconstruction work so far. In this chapter the SQP concept is introduced and a novel implementation that aims at solving constrained bundle adjustment problem is given. The point-on-surface constraint described in [2] is used to directly recover quadratic surfaces from multiple views using SQP. The optimization step and the surface reconstruction step are combined into one single step by SQP. Since surface geometry constraints are incorporated in optimization step, the 3D scene points' coordinates and camera parameters can be recovered more accurately by SQP than that by unconstrained algorithms. A much simpler representation that uses camera parameters and quadratic surface parameters is illustrated in our work to represent quadratic surface models. It can tremendously reduce storage space or network transmission needs. If point clouds and triangular patches were used to represent quadratic surface models, we would have to match hundreds of points across the images to obtain a visually smooth surface. However, in our work, only a dozen of matching points across images are needed to calculate the surface parameters. We can

generate as many points as we need on the surface and then project them onto observed images instead of matching points across images.

The proposed technique is different from previous quadratic surface reconstruction work [3,11,12,13], where the outlines in multiple views needed to be estimated first to recover the corresponding quadratic surface parameters. In principle the technique introduced in this chapter can be used to recover arbitrary parametric models such as lines, planes and freeform surfaces from multiple views. The chapter is organized as follows. In Section 11.2, we formulate quadratic surface reconstruction problem. In Section 11.3, we describe SQP nonlinear minimization concept and its novel implementation. In Section 11.4, we outline the steps in quadratic surface reconstruction. In Section 11.5, the experimental results with both computer simulation data and real images are shown and the power of SQP is verified. We conclude the chapter in Section 11.6.

11.2. Formulation

The quadratic surface reconstruction problem is formulated in this section.

11.2.1 Quadratic Surface Representation

A quadratic surface is a second-order algebraic surface given by:

$$h(\mathbf{X}, \mathbf{Q}) = \mathbf{X}^T \mathbf{Q} \mathbf{X} = 0 \quad (11.1)$$

where $\mathbf{Q}$ is a symmetric matrix, $\mathbf{X}=(x,y,z,1)^T$ is a homogeneous 4-vector which represents a point in 3D. Some instances of quadratic surfaces are shown in Figure 11.1.

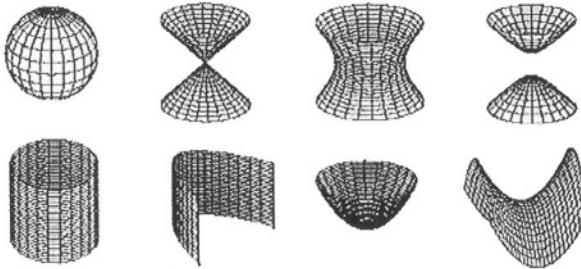


Figure 11.1. Some instances of quadratic surfaces. First row: ellipsoid, cone, hyperboloid of one sheet and hyperboloid of two sheets. Second row: elliptic cylinder, paraboloid and hyperbolic paraboloid.

A quadratic has nine degrees of freedom corresponding to the independent elements of Q up to an overall scale. Eq. (11.1) can be rewritten in the form:

$$a\mathbf{v} = 0, \quad (11.2)$$

where a is a 1×10 matrix which can be decided by point X only, $\mathbf{v}$ is a homogeneous 10-vector containing the distinct matrix elements of Q . Each 3D point provides a similar constraint, so that from N points a matrix equation $A\mathbf{v}=0$ can be constructed, where A is $N \times 10$ matrix formed from the stacked matrices a . The solution of $\mathbf{v}$ corresponds to the one dimension null-space of A . If $N \geq 9$ and the N points are in general position then the quadratic surface parameters can be uniquely determined.

11.2.2 3D Reconstruction from Multiple Views

Suppose we have matched a number of points of interest across M images using for example the technique described in [14]. Because of occlusion, feature detection failure and other reasons, a scene point can only be observed and detected in a subset of the M images (c.f. Figure 11.2). Suppose a 3D point X is observed as $\mathbf{x}=\mathbf{P}\mathbf{X}$, $\mathbf{x}'=\mathbf{P}'\mathbf{X}$ in arbitrary two images, where image points $\mathbf{x}, \mathbf{x}'$ are represented by homogeneous 3-vectors, $\mathbf{x}=(x,y,1)^T$, and $\mathbf{P}, \mathbf{P}'$ are 3×4 camera projection matrices for the two views. Given the fundamental matrix $\mathbf{F}$ for the view pair, then from [15,16,17] the camera matrices can be chosen as: $\mathbf{P} = [\mathbf{I} | \mathbf{O}]$, $\mathbf{P}' = [[\mathbf{e}']_x \mathbf{F} | \mathbf{e}']$, where $\mathbf{e}'$ is the epipole in the second image ($\mathbf{F}^T \mathbf{e}' = 0$) and $[\mathbf{e}']_x$ is the 3×3 skew matrix such that $[\mathbf{e}']_x \mathbf{x} = \mathbf{e}' \times \mathbf{x}$. The 3D point X is then reconstructed from its image correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$ by back-projection (via $\mathbf{P}, \mathbf{P}'$) and triangulation[18]. After the projective reconstruction is obtained, the technique described in [19] can be used to upgrade the projective reconstruction to a metric one.

11.2.3 Traditional Unconstrained Bundle Adjustment

If initial parameters have been estimated by linear method as illustrated above, a bundle adjustment step is often used to refine initial parameters. A cost function needs to be defined in bundle adjustment to quantify the fitting error of the estimated parameters. In traditional unconstrained minimization method, the cost estimation is often obtained by minimizing the sum of squared errors between the observed image points and the predicted image points. More formally, it can be represented as:

$$C = \sum_{j=1}^M \sum_{i=1}^N \left\{ \left(f_j \frac{x_{ij}}{z_{ij}} + u_{j0} - u_{ij} \right)^2 + \left(f_j \frac{y_{ij}}{z_{ij}} + v_{j0} - v_{ij} \right)^2 \right\}$$

where M is the total number of cameras, N is the total number of points, f_j is the focal length of j th camera. X_{ij} , Y_{ij} and Z_{ij} are the i th point's coordinates under j th camera's coordinate system. u_{ij} and v_{ij} are the coordinates of i th point observed in j th image.

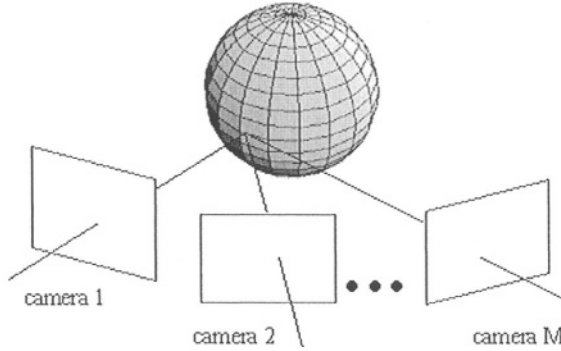


Figure 11.2. Quadratic surfaces observed by multiple views.

Since 3D scene constraints are often not enforced in traditional unconstrained bundle adjustment, the optimized isolated features do not satisfy geometry constraints. For example, if a quadratic surface is recovered using traditional unconstrained minimization method, the isolated feature points will not be strictly on the same quadratic surface.

11.3. Sequential Quadratic Programming

In order to impose 3D scene constraints in optimization, constrained minimization is often needed in bundle adjustment algorithm. Sequential Quadratic Programming is a powerful algorithm and has been proved highly effective for solving general constrained optimization problems.

11.3.1 SQP Problem Formulation

First the basic SQP principle is introduced. Consider the general equality constrained minimization problem P:

$$\begin{aligned} & \text{(P) } \min f(x) \\ & \text{s.t. } h_j(x)=0 \quad j=1,\dots,m. \end{aligned}$$

Here x is the desired variable vector, $f(x)$ is the objective function and $h_j(x)$ is the equality constraint. The Lagrangian function associated with problem (P) is:

$$L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \mathbf{h}(\mathbf{x})^T \mathbf{u}, \mathbf{u} \in \mathbf{R}^m, \quad (11.3)$$

where $\mathbf{u}$ is the corresponding Lagrangian multiplier vector and $\mathbf{h}(\mathbf{x})$ is the vector of equality constraints.

11.3.2 Local Analysis

Given a current solution $(\mathbf{x}_k, \mathbf{u}_k)$ which is sufficiently close to an optimal solution $(\mathbf{x}^*, \mathbf{u}^*)$, we seek to locally approximate problem (P) by a quadratic subproblem, i.e., an optimization problem with a quadratic objective function and linear constraints. The form of the quadratic subproblem most often found in literature [8], and the one that will be employed here, is

$$(QP) \min_{\mathbf{d}_x} \nabla f(\mathbf{x}_k)^T \mathbf{d}_x + \frac{1}{2} \mathbf{d}_x^T \nabla^2 L(\mathbf{x}_k, \mathbf{u}_k) \mathbf{d}_x \quad (11.4)$$

$$\text{s.t. } \nabla \mathbf{h}(\mathbf{x}_k) \mathbf{d}_x + \mathbf{h}(\mathbf{x}_k) = \mathbf{0}, \quad (11.5)$$

where $\mathbf{d}_x = \mathbf{x} - \mathbf{x}_k$. From the first order optimality conditions for the quadratic subproblem [8], the following equations can be obtained to compute the update directions $(\mathbf{d}_x, \mathbf{d}_u)$:

$$\nabla^2 L(\mathbf{x}_k, \mathbf{u}_k) \mathbf{d}_x + \nabla f(\mathbf{x}_k) + \nabla \mathbf{h}(\mathbf{x}_k)^T (\mathbf{u}_k + \mathbf{d}_u) = 0$$

$$\nabla \mathbf{h}(\mathbf{x}_k) \mathbf{d}_x = -\mathbf{h}(\mathbf{x}_k)$$

The equations can be rewritten in matrix format as:

$$\begin{bmatrix} \nabla^2 L(\mathbf{x}_k, \mathbf{u}_k) & \nabla \mathbf{h}(\mathbf{x}_k)^T \\ \nabla \mathbf{h}(\mathbf{x}_k) & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_u \end{bmatrix} = - \begin{bmatrix} \nabla L(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{h}(\mathbf{x}_k) \end{bmatrix} \quad (11.6)$$

where $\mathbf{d}_u = \mathbf{u} - \mathbf{u}_k$. The solution of $\mathbf{d}_x$ and $\mathbf{d}_u$ can be used to generate the new iterate. If we choose a suitable step-size α_k , the new iterate can be defined as:

$$(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) = (\mathbf{x}_k, \mathbf{u}_k) + \alpha_k (\mathbf{d}_x, \mathbf{d}_u).$$

Once the new iterate is constructed, a set of new linear equations can be built and solved at point $(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$. In the analysis above it has been assumed that the current solution $(\mathbf{x}_k, \mathbf{u}_k)$ is sufficiently close to the optimal solution and the quadratic subproblem is always feasible. For the quadratic sub-problem to be solved, four conditions [8] should be satisfied. It has also been proved that if the

initial solution is sufficiently close to the optimal $\mathbf{x}^*$, the algorithm has quadratic local convergence property [8].

11.3.3 Global Analysis

If the current solution $(\mathbf{x}_k, \mathbf{u}_k)$ is not sufficiently close to an optimal solution $(\mathbf{x}^*, \mathbf{u}^*)$, the questions of whether the sequences generated by quadratic programming will converge must then be resolved. To ensure global convergence, SQP needs to be equipped with a measure of progress, a merit function ϕ , whose reduction implies progress towards an optimal solution [7,8]. The merit function used in constrained minimization must blend the need to reduce the objective function while keeping the constraints satisfied. And it is generally different from the unconstrained one.

One commonly used merit function is called ℓ_1 penalty merit function[7,8]. It can be written as:

$$\phi_1(\mathbf{x}; \rho) = f(\mathbf{x}) + \rho \sum_i |h_i(\mathbf{x})|,$$

where ρ is a positive constant to be chosen and $||$ means the absolute value of a function. It is sufficient to note that ϕ_1 is an exact penalty function [7,8]; that is, there exists a positive ρ^* such that for all $\rho \geq \rho^*$, an unconstrained minimum of ϕ_1 corresponds to a solution of the constrained nonlinear minimization problem.

11.3.4 New SQP Implementation

In practical implementation there are some problems need to be considered. We have assumed that the quadratic subproblem always has a feasible solution in the analysis above. To have a feasible solution, it has been illustrated in [7,8] that the system of constraints of the quadratic subproblem must have a nonempty feasible set and the quadratic objective function should be bounded below on that set. If the initial solution $\mathbf{x}_k$ is sufficiently close to the optimized solution, the above consistence conditions can be guaranteed. For nonlocal points, it is not necessarily true. An appropriate estimate of $\nabla^2 \mathbf{L}(\mathbf{x}_k, \mathbf{u}_k)$ can assure that a consistent quadratic problem will always have a solution. Some implementations have used BFGS algorithm to approximate the Hessian matrix. In our work, a novel implementation is used to avoid the infeasibilities. Consider function (11.4) in QP, for nonlocal point $(\mathbf{x}_k, \mathbf{u}_k)$, it may be a poor local approximation to solve the original problem (P). In that case, the original problem can only be described as below:

$$(\text{SP}) \min_{\mathbf{d}_x} L(\mathbf{x}, \mathbf{u}) \quad (11.7)$$

$$\text{s.t. } \nabla \mathbf{h}(\mathbf{x}_k) \mathbf{d}_x + \mathbf{h}(\mathbf{x}_k) = \mathbf{0}.$$

Since solution $(\mathbf{x}_k, \mathbf{u}_k)$ is a remote point, to minimize function (11.7), about all we can do is to take a step down the gradient, as in the steepest descent method[20]. It can be formally represented as:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = -\eta \nabla L(\mathbf{x}_k, \mathbf{u}), \eta > 0.$$

It can be rewritten as:

$$\lambda \mathbf{d}_x + \nabla \mathbf{h}(\mathbf{x}_k)^T \mathbf{d}_u = -\nabla L(\mathbf{x}_k, \mathbf{u}_k), \lambda = \frac{1}{\eta}. \quad (11.8)$$

Here λ is a suitable value and it should not exhaust the downhill direction. Eqs. (11.5) and (11.8) can be combined and rewritten in matrix format as:

$$\begin{bmatrix} \lambda \mathbf{I} & \nabla \mathbf{h}_k^T \\ \nabla \mathbf{h}_k & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_u \end{bmatrix} = - \begin{bmatrix} \nabla L(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{h}_k \end{bmatrix}, \quad (11.9)$$

where $\mathbf{I}$ is a $n \times n$ diagonal matrix. Combine Eqs. (11.6) and (11.9), we get the following Eq. (11.10):

$$\begin{bmatrix} \mathbf{B}_k & \nabla \mathbf{h}_k^T \\ \nabla \mathbf{h}_k & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_u \end{bmatrix} = - \begin{bmatrix} \nabla L(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{h}_k \end{bmatrix}, \quad (11.10)$$

where $\mathbf{B}_k = \nabla^2 L(\mathbf{x}_k, \mathbf{u}_k) + \lambda \mathbf{I}$.

The new equation has combined the merits of steepest descent method and Newton method. When $\mathbf{x}_k$ is sufficiently close to $\mathbf{x}^*$, λ can be adjusted to be very small, the modified matrix $\mathbf{B}_k$ is very close to the Hessian matrix. The Newton direction is used to approximate the next QP step. When $\mathbf{x}_k$ is not sufficiently close to $\mathbf{x}^*$, λ can be adjusted to be very large, the matrix $\mathbf{B}_k$ is forced to be diagonally dominant, the steepest descent direction is mainly used to approximate the next step. It has been proved that add a strictly positive diagonal matrix to $\mathbf{B}_k$ can produce generally more robust results than by basic SQP implementation [8]. Given an initial guess for parameters $\mathbf{x}$, our SQP implementation can be described as below:

SQP Implementation

- 1 Compute $\phi_1(\mathbf{x};\rho)$.
- 2 Pick a modest value for λ , say $\lambda \approx 0.001$.
- 3 Solve the linear equations in (11.10) for $(\mathbf{d}_x, \mathbf{d}_u)$ and evaluate $\phi_1(\mathbf{x} + \mathbf{d}_x; \rho)$.
- 4 If $\phi_1(\mathbf{x} + \mathbf{d}_x; \rho) \geq \phi_1(\mathbf{x}; \rho)$, increase λ by a factor of 10 and go back to step 3.
- 5 If $\phi_1(\mathbf{x} + \mathbf{d}_x; \rho) < \phi_1(\mathbf{x}; \rho)$, decrease λ by a factor of 10, update the trial solution $(\mathbf{x}, \mathbf{u}) = (\mathbf{x}, \mathbf{u}) + (\mathbf{d}_x, \mathbf{d}_u)$, and go back to step 3.

For the algorithm to stop, the same strategy employed in Levenberg Marquardt algorithm [20] has been used. The loop is stopped at the first occasion where $\phi_1(\mathbf{x}; \rho)$ decreases by a negligible amount. Once the acceptable minimum has been found, we set $\lambda = 0$ and compute the matrix $\mathbf{B}_k^{-1}$, the upper left part of which is the standard covariance matrix of the standard errors in the fitted parameters $\mathbf{x}$ [20,21].

11.3.5 Quadratic Surface Recovery Applications

In quadratic surface recovering applications, the quadratic surface equation is used as the constraint function. The corresponding Lagrangian function associated with our problem is:

$$LC = C + \sum_{j=1}^N u_j h(\mathbf{X}_j, \mathbf{Q}),$$

where u_j is the Lagrange multiplier, $\mathbf{X}_j$ is j th point coordinate vector in the world coordinate system, $\mathbf{Q}$ is the quadratic model matrix. The objective is to minimize function LC under constraints $h(\mathbf{X}_j, \mathbf{Q}) = 0$. The merit function used in our implementation is defined as:

$$\phi_1 = C + \rho \sum_{j=1}^N |h(\mathbf{X}_j, \mathbf{Q})|,$$

where $\rho > \max\{|u_j| \mid 1 \leq j \leq N\}$, N is the total number of constraints, $||$ means the absolute value of a variable or a function.

11.4. Outline of the Method

The algorithm can be outlined as follows:

- 1 Data Preparation: Collect matched image points across multiple views.
- 2 Compute the initial 3D points coordinates, intrinsic and extrinsic camera parameters and quadratic surface parameters using linear method.
- 3 Optimize the initial 3D point coordinates using the SQP implementation.
- 4 Build corresponding VRML model using the refined camera parameters and sphere parameters only.

11.5. Experimental Results

In this section, we provide experimental results of our algorithm with both synthetic and real data for a globe.

11.5.1 Synthesized Data

For the synthetic data, 3 views and total 16 points on a sphere are used. The 3 images have the same focal length of 1000 pixels. For each image, 16 image points are generated with isotropic uniform Gaussian noise of $\delta=1.0$. We first calculate the initial camera parameters and points coordinates using linear methods.

Comparison between SQP and Levenberg Marquardt Algorithm. After we have calculated the initial solutions, SQP and Levenberg Marquardt algorithm are used to optimize the initial solutions. SQP optimization converges within 10 steps. The constraint functions become strictly satisfied after SQP optimization step. The maximum absolute error of the constraint function is no more than $5.0E-7$. The cost function C has the value 8.32247. For each pixel, the mean error is 0.416395 pixels. The errors of the optimized coordinates are illustrated in Figure 11.3. The error is computed as $\|X - X_{\text{True}}\|$. X is the optimized coordinates of the points and X_{True} is the true coordinates of the points. It can be seen that the solutions computed by SQP are generally closer to the true ones. The errors of the focal length and rotation/translation parameters are illustrated in Figure 11.4, Figure 11.5 and Figure 11.6. Here the rotation is represented in angle/axis format. The length of the translation vector between the first camera and the second camera is normalized to 1. So the translation vector has only two free parameters. It can also be seen that the camera parameters computed by SQP are generally closer to the true ones. The errors of the sphere parameters are illustrated in Figure 11.7. Here the sphere parameters of Levenberg Marquardt algorithm are estimated from the

optimized 3D coordinates using the technique described in Section 11.2. It can also be seen that the sphere parameters computed by SQP are more close to the true parameters. The differences of the sphere surface constraints between SQP and Levenberg Marquardt algorithm are shown in Figure 11.8. The solutions computed by SQP strictly satisfy the point-on-surface constraint. But the solutions computed by Levenberg Marquardt algorithm often deviate from the sphere surface.

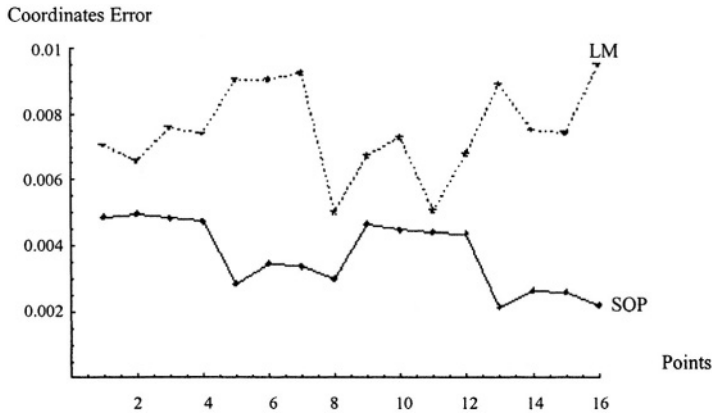


Figure 11.3. The error of the optimized coordinates optimized by SQP and Levenberg Marquardt algorithm.

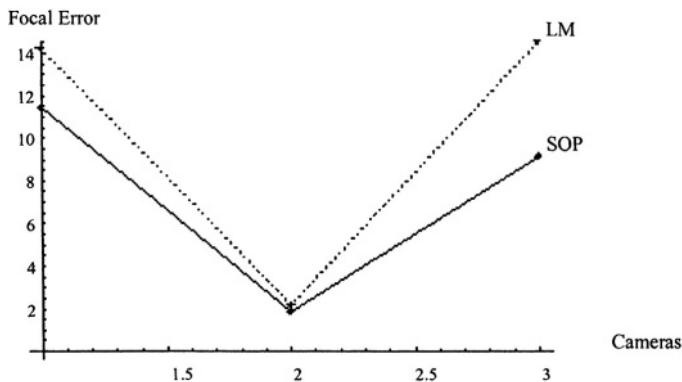


Figure 11.4. The error of the focal length optimized by SQP and Levenberg Marquardt algorithm.

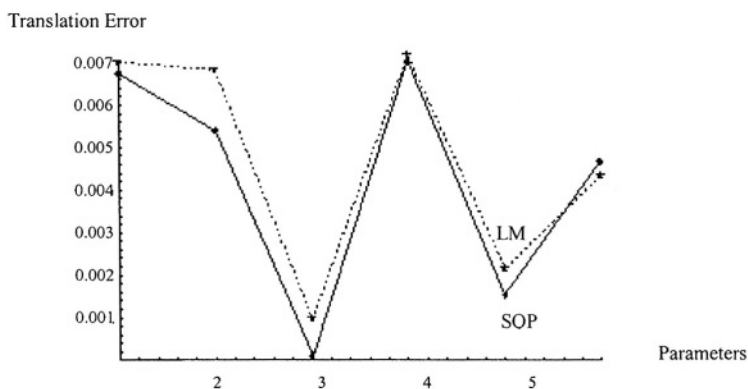


Figure 11.5. The error of the rotation parameters optimized by SQP and Levenberg Marquardt algorithm.

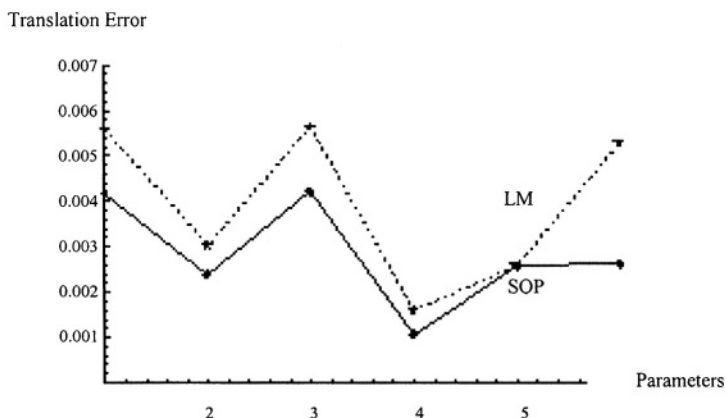


Figure 11.6. The error of the translation parameters optimized by SQP and Levenberg Marquardt algorithm.

Standard Deviation Comparison between SQP and Levenberg Marquardt Algorithm.

The square roots of the diagonal items of the upper left part of the inversed optimized gradient matrix B_k^{-1} represent the standard deviations of the fitted parameters [20,21]. If any diagonal item is too high, it means that the corresponding parameter has low confidence. The standard deviations of the 3D coordinates of the points are listed in Figure 11.9. It can be seen that the standard deviation values of the coordinates of the points computed by SQP are generally smaller than that by Levenberg Marquardt algorithm. The standard deviations of the camera intrinsic and extrinsic parameters are listed in Figure 11.10, Figure

11.11 and Figure 11.12. It can also be seen that the standard deviations of the camera parameters computed by SQP are generally smaller than that by Levenberg Marquardt algorithm. It means that the camera parameters computed by SQP generally have higher confidence than that by Levenberg Marquardt algorithm.

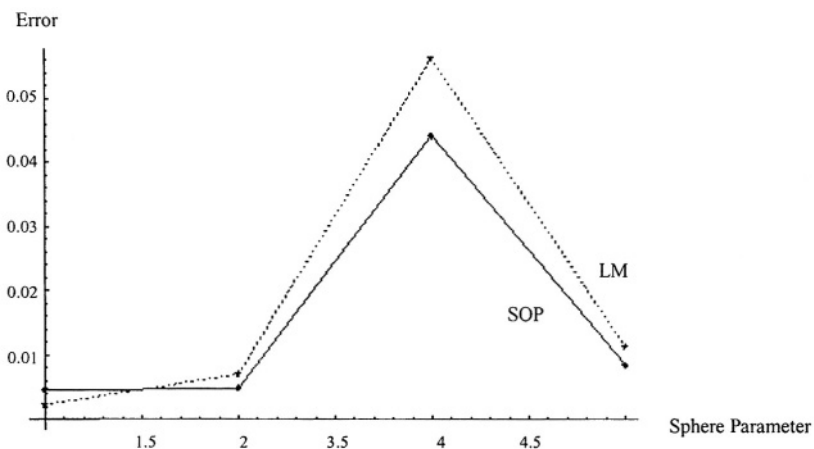


Figure 11.7. The error of the sphere parameters calculated by SQP and Levenberg Marquardt algorithm.

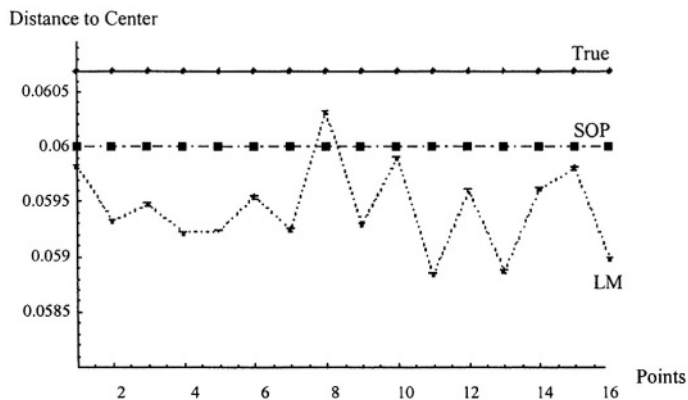


Figure 11.8. The comparison of the point-on-surface constraint between SQP and Levenberg Marquardt algorithm.

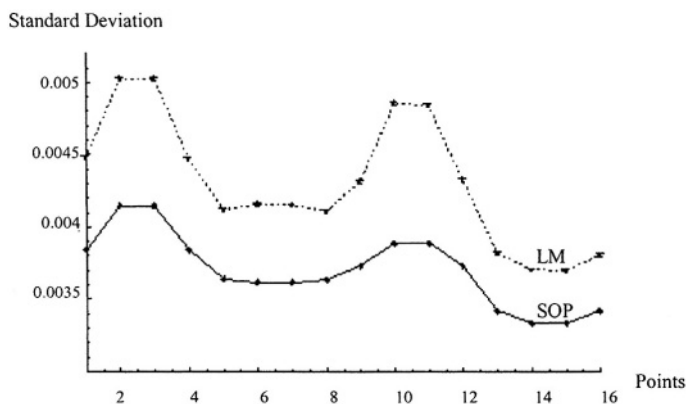


Figure 11.9. The standard deviations of the points computed by SQP and Levenberg Marquardt algorithm.

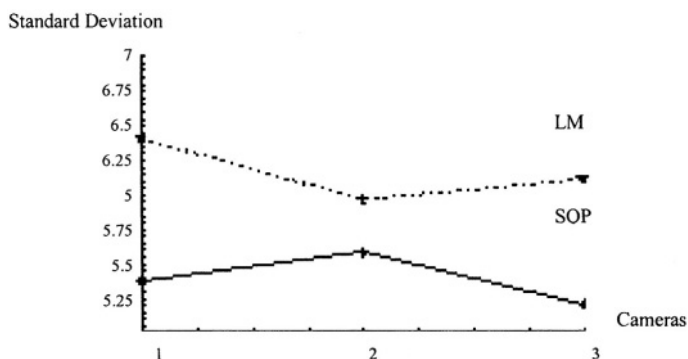


Figure 11.10. The standard deviations of the camera focal length computed by SQP and Levenberg Marquardt algorithm.

Relationship between Constraints Number and Standard Deviation.

We have also done experiments to verify the impact of the number of constraints on the final standard deviations. When there are no constraints, SQP degenerates into Levenberg Marquardt algorithm. By experiment we find that at least 4 points are needed in our experiment. When the constraints number is less than 4, SQP will not be able to calculate the sphere parameters correctly. In theory the sphere surface has 4 degrees of freedom and at least 4 points are needed, it is compatible to the experiment. We have also found that when the number of constraints increases, the standard deviations of the parameters decrease. This means that the

optimized parameters become more accurate as the number of constraints increases. The standard deviations of the sixteen points computed by SQP are illustrated in Figure 11.13.

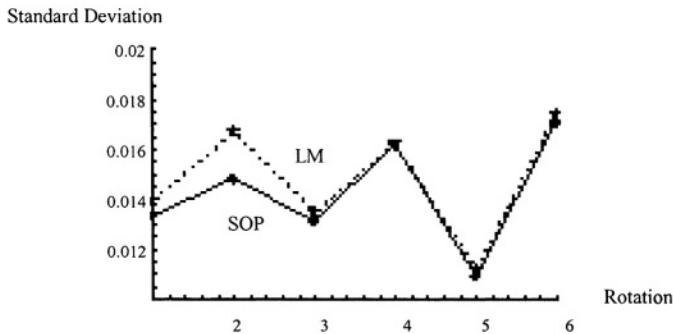


Figure 11.11. The standard deviations of the rotation parameters computed by SQP and Levenberg Marquardt algorithm.

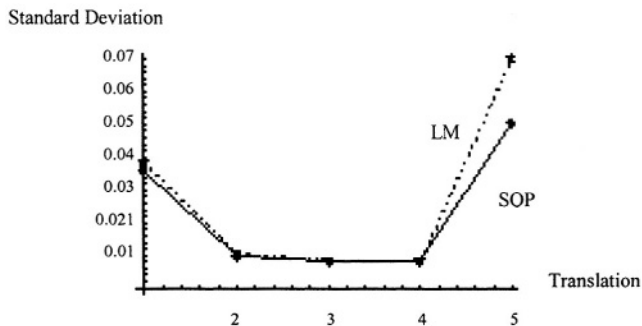


Figure 11.12. The standard deviations of the translation parameters computed by SQP and Levenberg Marquardt algorithm.

Relationship between Cost Function and Gaussian Noise. We have also done experiments with different Gaussian noise parameter δ . The constraint functions are strictly satisfied in SQP optimization, the maximum absolute error of the constraint function is no more than $1.0E-5$. The SQP implementation converges within 10 steps. The relationship between the square root mean of the

cost function $\sqrt{\frac{C}{N}}$ and Gaussian noise parameter δ is shown in Figure 11.14,

where C is cost function, N is the number of points. It can be seen from the figure that the value of the cost function C increases when the image Gaussian noise increases. The square root mean value is close to the gaussian noise parameter δ .

11.5.2 Real Images

For experiments on real images, 17 pictures have been taken around a globe. The camera is Power Shot Pro 70, a digital camera manufactured by Canon. The images have the same size. The size is 1525*1021 pixels. Four of them are shown in Figure 11.15. The initial 3D points coordinates, camera intrinsic and extrinsic parameters are calculated using the linear methods. The sphere parameters can then be calculated using the technique described in Section 11.2.

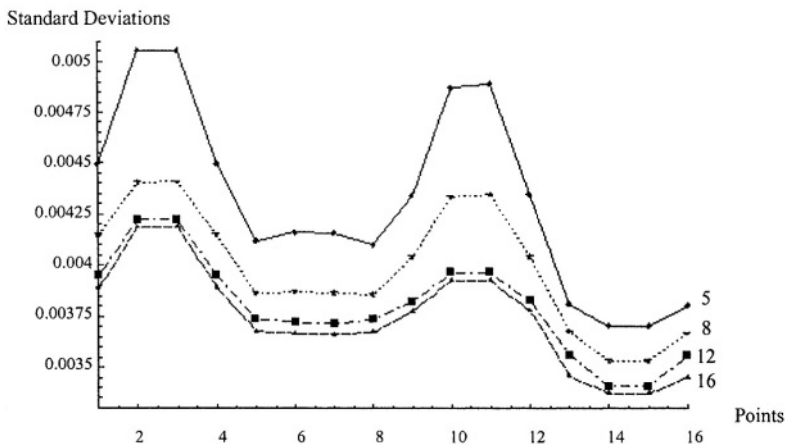


Figure 11.13. The standard deviations of the points computed by SQP algorithm with different constraint numbers.

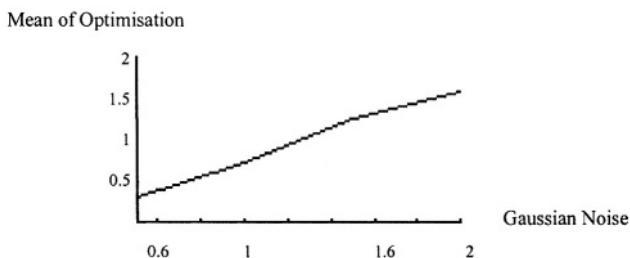


Figure 11.14. Relation between square root mean of the cost function C and gaussian noise.

SQP Optimisation. We first feed the initial 3D coordinates into SQP optimization. The SQP implementation converges within 20 steps. For the real images, the constraints are all strictly satisfied. The maximum absolute error of the constraint function is no more than $1.0E-6$. But the points computed by Levenberg Marquardt algorithm deviates from sphere surface by 4.5% or so. Once the refined

camera parameters and sphere parameters are computed, the small number of parameters can be used to build the globe VRML model.

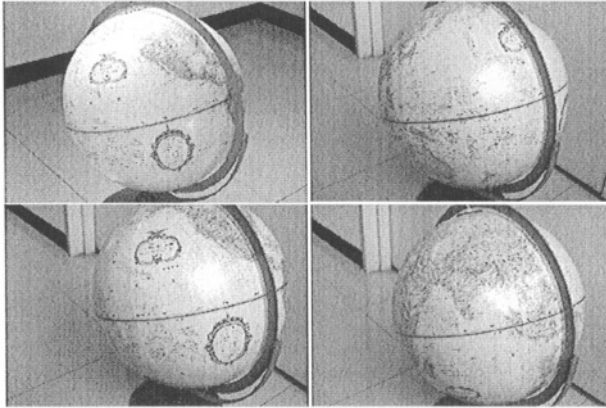


Figure 11.15. Four pictures of sequence of pictures.

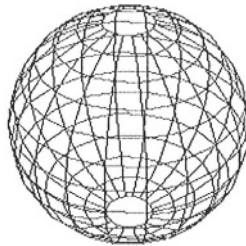


Figure 11.16. Automatically generated sphere points.

Automatic Texture Generation. Instead of using traditional methods to match the feature points among different pictures, the sphere parameters are used to generate the arbitrary number of points located on the surface as shown in Figure 11.16. The triangle patches and quadrilateral patches are then generated and projected onto different images. The normal of the 3D triangle/quadrilateral patches are then calculated. The vectors between sphere center and camera center are also calculated. Then we calculate the smallest angle between the normal of the 3D patch and sphere-camera vectors, the smallest angle of each 3D patch is found and the corresponding 2D image patch is selected as the texture. Figure 11.17 shows the selected patches on one picture. By combining the selected texture patches and the automatically generated 3D points coordinates, we then

cost of our current implementation is the approximated Hessian matrix computation. We are planning to use the sparseness matrix properties to speed up its computation. We also need to do the texture blending part to have a visually smooth surface map. We would like to apply SQP to model more free-form parametric surfaces such as the human face, body and arms.

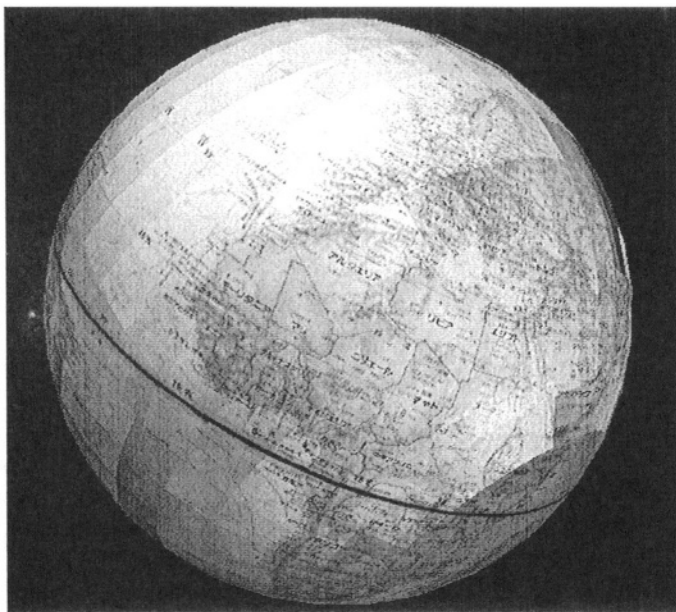


Figure 11.18. VRML model of the globe.

References

- [1] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. "Bundle adjustment -- modern synthesis". In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pp. 298--375. Springer Verlag, 2000.
- [2] Ying Shan, Zicheng Liu, Zhengyou Zhang. "Model-based bundle adjustment with application to face modeling". In *Proc.ICCV*, pp. 644-651, 2001
- [3] G. Cross and A. Zisserman. "Quadratic surface reconstruction from dual-space geometry". In *Proc. ICCV*, pp. 25-31, 1998.
- [4] P. McLauchlan, X. Shen, P. Palmer, A. Manassis, and A. Hilton. "Surface-based structure-from-motion using feature groupings". In *Proceedings of the Asian Conference On Computer Visions*, pp. 699-705, Taipai, Taiwan, Jan. 2000. Springer-Verlag.

- [5] R.Szeliski and P.Torr. "Geometrically constrained structure from motion:Points on planes". In *Proceedings of the European Workshop on 3D structure from Multiple Images of Large-Scale Environments(SMILE)*, pp. 171-186, Freiburg, June 1998.
- [6] Th.F.Coleman and A.R.Conn. "Nonlinear programming via an exact penalty function:Global analysis", *Mathematica Programming*, vol. 24, pp.137-161, 1982.
- [7] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization, numerical analysis". Report 97-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.
- [8] P.T. BOGGS and J.W. TOLLE. "Sequential quadratic programming". *Acta Numerica*, vol. 4, pp. 1- 50, 1995.
- [9] Paul T. Boogs, Anthony J. Kearsley, and Jon W. Toole. "A global divergence analysis of an algorithm for large scale nonlinear programming problems". *SIAM Journal on Optimizations*, vol. 9, no. 3, pp. 755-778, 1999.
- [10] B. Triggs. "Autocalibration and the absolute quadric". In *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 609-- 614, 1997.
- [11] S.D. Ma and L. Li. "Ellipsoid reconstruction from three perspective views". In *Proc. International Conference on Pattern Recognition*, Vienna,1996.
- [12] S.D. Ma and X. Chen. "Quadric surface reconstruction from its occluding contours". In *Proc. International Conference on Pattern Recognition*, Israel, 1994.
- [13] L. Quan. "Invariant of a pair of non-coplanar conics in space: definition, geometric interpretation and computations". In *Proc. 5th Int'l Conf. On Computer Vision*, pp. 926-931, 1995.
- [14] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong. "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry". *Artificial Intelligence*, pp. 78, pp. 87--119, 1995.
- [15] A. Hartley, R. Zisserman. "Multiple view geometry in computer vision". MIT Press, 2000.
- [16] O. Faugeras. *Three-dimensional Computer Vision: a Geometric Viewpoint*, MIT Press, 1993.
- [17] Q.-T. Luong, T.Vieville. "Canonic representations for the geometries of multiple projective views", *ECCV/94, Lecture Notes in Computer Science*, vol. 800. Ed. JanOlof Eklund, Springer-Verlag, pp. 589-599, 1994.
- [18] R. Hartley and P.Strum. "Triangulation". In *Proc. Conference Computer Analysis of Images and Patterns*, Prague, Czech Republic,1995
- [19] Marc Pollefeys and Luc Van Gool. "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters". In *Proc. Int. Conf. on Computer Vision*, 1998.
- [20] W. Press, S. Teukolski, W. Vetterling, B. Flannery. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, 2nd ed., 1992.
- [21] Hartmann, Wolfgang M. and Hartwig, Robert E.. "Computing the Moore-Penrose inverse for the covariance matrix in constrained nonlinear estimation". SAS Institute, Inc., Cary, NC , 1995

- [22] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno. "Multiple textures stitching and blending on 3d objects, rendering". Techniques '99, Springer -Verlag Wien, pp. 119—130, 1999.

This page intentionally left blank

Chapter 12

PHOTO-REALISTIC CONVERSATION AGENT

Bo Zhang ¹, Zicheng Liu ² and Baining Guo ¹

1. Microsoft Research Asia, 3/F Sigma Center, 49 Zhichun Road, Haidian District, Beijing 100080, P.R.China

2. Microsoft Research Redmond, One Microsoft Way Redmond, WA 98052-6399, USA
{bzhang,zliu,bainguo}@microsoft.com

Abstract

Image-based facial animation (IBFA) techniques allow us to build photo-realistic talking heads. In this chapter, we address two important issues in creating a conversation agent using IBFA. First, we show how to make a conversation agent appear to be alive during the conversation. Second, we present techniques for lip-syncing different languages in the same IBFA system. Using these techniques, we built two conversation agents, an English speaker and a Mandarin Chinese speaker, in our E-Partner system.

We also present a geometry-driven facial expression synthesis system, which is very important to make talking head more naturally in the future. Given the feature point positions (*geometry*) of a facial expression, our system automatically synthesizes the corresponding expression image which has photo-realistic and natural looking expression details. This technique can be used to enhance the traditional expression mapping technique, or let user edit facial expression directly by moving feature points in our expression editing system.

Keywords: Conversation agent, image-based facial animation, facial expression

12.1. Introduction

Computer-animated characters, particularly talking heads, have a variety of applications including video games and web-based customer services. A talking head attracts the user's attention and makes user interactions more engaging and entertaining. For a layperson, seeing a talking head makes interaction with a computer more comfortable. Subjective tests show that an E-commerce web site with a talking head gets higher ratings than the same web site without a talking

head [18]. Researchers have also reported that in web-based training, animated characters can improve learning results by increasing the user's attention span.

12.1.1 Image-based Facial Animation

Recently there has been a surge of interest in image-based facial animation (IBFA) techniques [4, 6, 9]. IBFA techniques allow us to build photo-realistic talking heads. In this chapter, we explore the use of these techniques in creating photo-realistic conversation agents, i.e., talking heads that not only look photo-realistic but also can have a conversation with the user about a given topic. To support conversational interactions, we address two important issues. First, we show how to make the conversation agent appear to be alive during a conversation. In particular, the conversation agent needs to appear to be listening (instead of just freezing) when the user speaks, and the transitions between talking and listening states must be smooth. Additionally, the conversation agent must exhibit intelligent and adaptive behavior during a conversation [29].

The second issue we address is that of lip-syncing different languages with the same IBFA system. Existing IFBA techniques are mostly designed with only English in mind. With video rewrite [4], for example, we found it very difficult to lip-sync Mandarin Chinese because of the difficulties in obtaining viseme classes. To accommodate several languages in the same IBFA system, we propose a novel technique for computing visemes and their distances for a given video database. With this technique, we can best capture the facial motion characteristics of the given language and speaker as recorded by the given video database.

We have implemented our techniques in the E-partner system for creating photo-realistic conversation agent. Our system combines the techniques described in this chapter and other component technologies, including speech recognition, natural language processing, and speech synthesis. We demonstrate the effectiveness of our techniques with two conversation agents, an English speaker and a Mandarin Chinese speaker.

12.1.2 Facial Expression Synthesis

Another big challenge to make conversation agent more naturally is the synthesis of realistic facial expression, which has been one of the most interesting yet difficult problems in computer graphics [19]. Great success has been achieved in generating the geometric motions of the facial expressions. For example, expression mapping [5, 13, 27, 19] is a popular technique that uses a performer's feature point motions to drive the feature point motions of a different person's face. Another approach is to pre-design a set of basic control mechanisms such as the action units [16, 20] or the FACS parameters [7]. At run time, the control parameters are applied to the face model to generate facial animations. Physically-based approach [1, 26, 23, 11] used mass-and-spring model to simulate the skin

and muscles. This approach is capable of generating skin deformations from facial muscle forces. Unfortunately, it is difficult to generate detailed skin deformations such as expression wrinkles because it is hard to model the dynamics in such a fine detail.

One effective approach to generate photo-realistic facial expressions with details is the morph-based approach [2, 22, 4, 21]. In particular, Pighin *et. al.* [21] used the convex combinations of the geometry and textures of the example face models to generate photo-realistic facial expressions. Their system was mainly designed for offline authoring purpose and it requires a user to manually specify blending weights to obtain a desired expression.

We propose a geometry-driven facial expression synthesis system [30]. We observe that the geometric positions of the face feature points are much easier to obtain than the photo-realistic facial expression details. So we propose to derive the expression images based on the geometric positions of the face feature points. Our approach is example-based. We first subdivide the face into a number of subregions, each associated with a subset of the face feature points. Given the feature point positions of a new expression, for each subregion we project the geometry associated with this subregion into the convex hull of the example expression geometries. The resulting convex combination coefficients are then applied to the example images to generate the desired image for the subregion. The final image is produced by seamlessly blending the subregion images together. Our technique can be used to enhance many existing systems which can generate feature point positions such as expression mapping, physically-based methods, and parameter control approaches. The combined system will be able to automatically produce photo-realistic and natural looking facial expressions.

12.1.3 Related Work

There are many other works on facial animation. It is virtually impossible to enumerate all of them here. We will name a few that we think are mostly related to our work.

There has been a lot of success on speech driven facial animation [2, 3, 8]. Speech driven facial animation systems are mainly concerned about the mouth region, while our method is mainly for facial expressions. One interesting analogy is that speech driven animation systems use audio signals to derive mouth images, and our system uses feature point motions to derive the facial images. It would be interesting to combine these two techniques together to generate speech-driven facial animations with expressions.

Toelg and Poggio [24] proposed an example-based video compression architecture. They divided the face into subregions. For each subregion, they used image correlation to find the best match in the example database and send the index over the network to the receiver.

Guenter *et. al.* [10] developed a system to use digital camera to capture 3D facial animations. Noh and Neumann [17] developed the expression cloning technique to map the geometric motions of one person's expression to a different person.

Liu *et. al.* [14] proposed a technique, called expression ratio image, to map one person's expression details to a different person's face. Given the feature point motions of an expression, their method requires an additional input of a different person's image with the same expression. In other words, their method requires an image of someone for every different expression. In contrast, our method can generate arbitrary number of expressions from a small set of example images. For the situations where no examples are available for the target face, their method is more useful. For the situations where we are given the feature point positions of an expression but no expression ratio images are available for this geometry, our method is more useful. For example, in expression editing applications, when a user manipulates the face feature points, it is unlikely that he/she can find a different person's expression image with exactly the same expression. In expression mapping applications, if the performer has markers on his/her face or if there are lighting variations due to the head pose changes, the ratio images may be difficult to create.

Our method differentiates from the work of Pighin *et. al.* [21] in that we automatically synthesize the expression images while their system is designed for offline authoring. The ability to automatically synthesize the expression details from the geometry makes our technique valuable for many existing facial animation systems which can generate the geometric information.

12.1.4 Structure of the Chapter

The remainder of this chapter is organized as follows. In next section, we provide the architecture of our E-Partner system. IBFA techniques are described in detail in section 12.3. In section 12.4 we present our facial expression synthesis system. We conclude in section 12.5 with discussions and future work.

12.2. E-Partner System Architecture

We have developed two conversation agents using E-Partner system: Maggie the tour guide speaks English, while Cherry can introduce Microsoft Research Asia to visitors in Chinese.

Figure 12.1 is a snapshot of a user interacting with Maggie. The user converses with Maggie through a microphone. Maggie resides in the lower right corner of the screen. She talks with the user about different places of the Forbidden City, and presents multimedia materials (images, videos, etc.) in the content window in the upper left corner of the screen.

The main components of our E-Partner system (Figure 12.2) are:



Figure 12.1. A user interacts with Maggie.

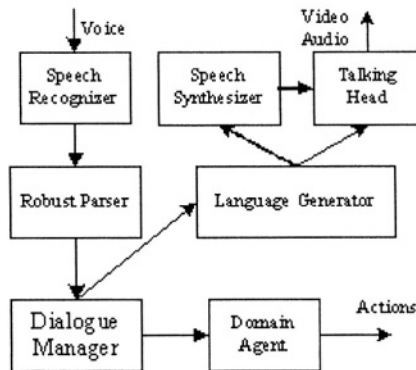


Figure 12.2. E-Partner system architecture.

Speech Recognizer. We use Microsoft speech recognition engine, and gather rich low-level information (sound or phrase start event, recognition hypothesis, etc.) to aid the dialogue manager to handle the uncertainty.

Robust Parser. Get the semantic interpretation of the text input using [25].

Dialogue Manager. Select appropriate actions based on the semantic input.

Domain Agent. Execute the non-verbal action, i.e., show picture.

Language Generator. Execute the verbal action. A template-based approach is used to generate responses.

Speech Synthesizer. Generate voice from text using MSR Asia TTS engine. We can also use pre-recorded speech if we want higher quality voice.

Talking Head. Synthesize facial animation from speech.

In this chapter, we focus on the facial animation synthesis of the talking head.

12.3. Facial Animation

Facial animation in our system is based on the technique of video rewrite [4]. For any given sentence and background footage, video rewrite works by synthesizing mouth motions, which are lip-synched with the sentence, and stitching the mouth motions with the background video.

In our application, we have to synthesize the background video at run time for any given sentence because it is practically impossible to store all possible

background video. We have developed a technique to automatically synthesize background video for sentences with any length, and further more, to concatenate the background video sequences smoothly.

Another advantage of our system is the ability to handle different languages easily. In some languages such as Chinese, there is neither standard classification of visemes nor confusion matrices of similar visemes. As a result, it is difficult to directly use the phoneme-context distance as in video rewrite [4]. We propose a language-independent representation of visemes so that we can compute viseme distances directly from video database without any knowledge of the language being spoken.

Figure 12.3 is the overview of our facial animation system. It consists of an offline analysis phase and an online synthesis phase.

12.3.1 Analysis

In the analysis phase, we first acquire a video database by filming a person talking. We then track the head pose and lip motion automatically in the video. The techniques that we use for head pose and lip tracking are similar to the ones used in [4]. But we have improved facial pose estimation so that the resulting head poses are smooth. As a result, we are able to completely eliminate the jerkiness in the synthesized result.

We use two strategies to improve the facial pose estimation. One is to apply second-order prediction to determine the initial pose. The other is to detect false abrupt motions (due to tracking errors), and to low-pass filter the pose parameters. For each frame with large motion, we first interpolate the pose parameters of its pre frame and post frame, and use the interpolated pose parameters to compute the residue error. If the residue error from using the interpolated pose is larger than the original one, we believe that the abrupt motion is a true motion. Otherwise, the parameters take the filtered value. Notice that it is not correct to linearly interpolate the elements of the pose matrices. We instead decompose the pose matrix as the following:

$$\begin{bmatrix} a_1 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & k & 0 \\ k & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If we maintain the order and assume there are only small motions, the parameters actually have physical meanings: t_x , t_y are X-direction and Y-direction offsets of the translation; θ is the rotation angle; k is the shear coefficient; s_x , s_y are X-direction and Y-direction scale coefficients. We use k and θ to determine whether abrupt motion occurs or not, and to predict and filter on the physical

parameters. Finally, the new affine matrix is computed by inverse transformation of the physical parameters. Our experiments show that the parameters are more continuous than the original method, and that the jerkiness is completely eliminated.

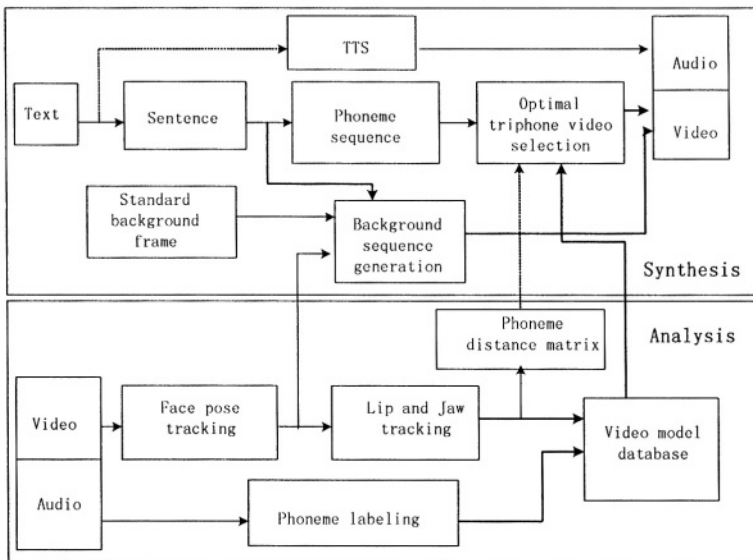


Figure 12.3. Facial animation analysis and synthesis.

12.3.2 Synthesis

The synthesis phase consists of four steps. We first convert the input text to wave file and phoneme sequence, from which we generate an appropriate background sequence. The third step is to find an optimal triphone video sequence which best fits the input phoneme sequence. Finally we time-align the resulting triphone video sequence and rewrite it back to the background sequence. As we mentioned before, there are two major differences between our system and the video rewrite system [4]. The first is that we automatically synthesize the background image sequence. The second difference is that we use a language-independent viseme representation so that we can handle new languages easily. Particularly, we are able to handle those languages without standard classification of visemes or confusion matrices of similar visemes.

12.3.2.1 Background Video Generation

The facial animation synthesized with video rewrite technique is a composite of two parts. The first part is a video sequence of the jaw and mouth, which is lip-synched with the sentence to be spoken. The second part is a "background video" of the face. The video sequence of the mouth region is generated from the annotated viseme database which is obtained from the video database during the analysis phase. To support interactive conversation in our system, we need to automatically synthesize the background video for any given length (of the sentence to be spoken or an idle period such as listening state) at run time.

We synthesize the background video sequences from the video database. The basic requirement is that these generated sequences can be concatenated seamlessly. The idea of our approach is to make sure that the starting and ending frames of all the generated sequences are similar to each other. We first analyze the video database to find a frame, called standard frame, with maximal number of similar frames. Some of those similar frames become candidate frames. At run time, we simply pick the video segment between two candidate frames such that the length best matches the given length.

Similarity Measurement. The similarity between two frames is measured based on how close their head poses are. For each video frame in the database, we have a matrix to represent the transformation of the character's head pose in that frame to a particular frame. With this matrix we can obtain the transformation matrix of the pose from one frame to another. Given any two frames F_1 and F_2 , suppose F_1 has pose matrix M_1 and F_2 has pose matrix M_2 , then the pose matrix from F_1 to F_2 is $M' = M_1 * M_2^{-1}$.

Using the same decomposition of the pose matrix as used in pose tracking, we define the distance $D(F_1, F_2)$ between frame F_1 and F_2 as a weighted sum of the six pose parameters of M' :

$$D = w_1 * t_x + w_2 * t_y + w_3 * \theta + w_4 * k + w_5 * s_x + w_6 * s_y$$

where $t_x, t_y, \theta, k, s_x, s_y$ are the physical parameters of M' , and $w_1, \dots, w_6$ are their weights. We assign larger weights to k, s_x, s_y and smaller weights to θ, t_x, t_y because k, s_x, s_y may introduce undesirable distortion of the human face while rotation and translation are rigid motion which preserves the shape of the character's face.

Two frames are considered to be similar to each other if the distance between them is smaller than a user-specified threshold.

Standard Frame Selection. To find a frame with maximal number of similar frames, a brute force approach is to simply test every frame and choose the best one. However, we find that the idle frames (when the speaker pauses between sentences) usually repeat more often. To reduce computation time, we only test the idle frames and select the one with the largest number of similar frame as the standard frame.

Candidate Frames Filtering. To avoid the jerkiness of the concatenated video sequence, we have to filter out those similar frames where there are sudden head pose changes. For example, in Figure 12.4 (b), the rightmost green frame is such a frame. To filter out frames like this, we check all the neighbouring frames for each similar frame. If any of its neighbouring frames is not similar frame, then this frame is not considered as a candidate frame. More specifically, the filter is defined as the following equation where h is user-defined constant that determines the neighbour window size:

$$W_i = 1, \quad \text{if } D(F_{i+j}, S) < T \quad j \in (-h, h) \\ = 0, \quad \text{otherwise}$$

Finally, we eliminate all the similar frames whose distance is not the local minimum. Only those locally most similar frames become candidate frames.

Background Synthesis. To synthesize a sentence of n frame, we search for two candidate frames F_i and F_j ($i < j$) with minimal $|j - i - n|$. If $j - i > n$, we set the extra $j - i - n$ background frames to be idle frames. If $j - i < n$, we repeat the last background frame $n - (j - i)$ times. In our experiments, there are no noticeable artifacts when $|j - i - n| < 10$.

Although the candidate frames are similar to the standard frame, usually their poses are not exactly same. To ensure seamless concatenation between two sequences, we apply a pose transformation on all the starting and ending frames so that their poses are exactly same as that of the standard frame. Given $M_i(M_j)$ as the pose matrix from $F_i(F_j)$ to the standard frame, we use $M_i(M_j)$ to transform $F_i(F_j)$ to the standard frame. The intermediate frames are also transformed with a matrix interpolated linearly between M_i and M_j on their six parameters t_x, t_y, θ, k, s_x and s_y .

12.3.2.2 Optimal Triphone Video Sequence Generation

Given a phoneme sequence as input, we would like to find an optimal triphone video sequence that best fits the target phoneme sequence. To apply the dynamic programming approach as used in [4], we define a distance function that measures the visual similarity between different visemes. For English, there have been scientific studies on the similarity measurement of visemes so that the distance

functions can be readily defined [4]. But for some languages such as Chinese, there is no similarity measurement data available. One possibility is to use the phoneme similarity measurement used by speech community. Unfortunately, as we will show later, the synthesized results are poor. The reason is that visual similarity is very different from aural similarity. To overcome this difficulty, we propose a generic viseme representation that does not require similarity measurement data.

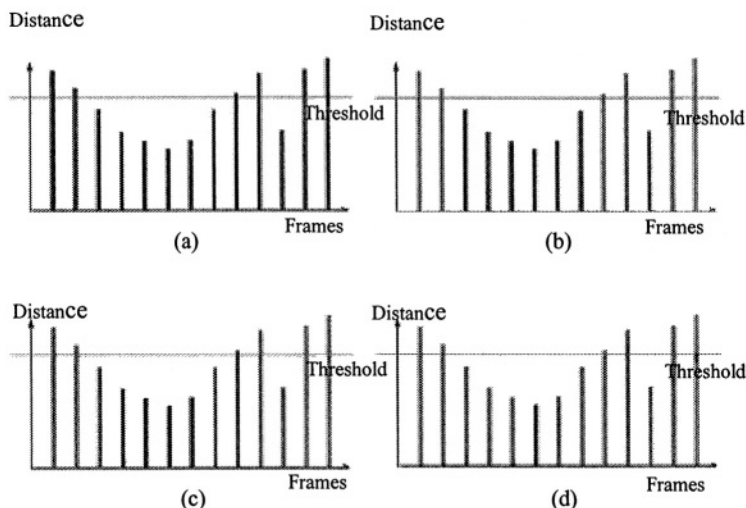


Figure 12.4. (a) Frames with distance to the standard frame. (b) Select similar frames. (c) Select continuous similar frames. (d) Select local-minimal similar frames.

Language-independent Viseme Representation. A viseme is the visual appearance of the lip when a person speaks a particular phoneme. We find that it is adequate to represent the lip shape by five appearance parameters (Figure 12.5): the width (a1) and the height (a2) of the mouth, the distance from the bottom of the upper lip to the top of the lower lip (a3), the distance from the bottom of the upper lip to the bottom of upper teeth (a4), and the distance from the top of the lower teeth to the top of the lower lip (a5). Since we have tracked the lip motion, the locations of the feature points around the lip are known for each frame in the database. From these feature points, we can easily obtain the values of these five parameters. Notice that a phoneme video segment usually contains multiple frames. Therefore a viseme can be represented as an array containing these parameters for each frame.

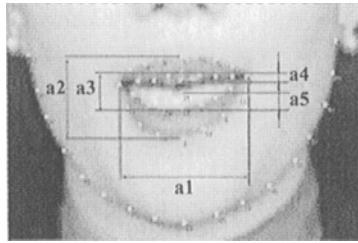


Figure 12.5. Five appearance parameters.

In general, each phoneme may occur in the video database many times and each occurrence may have different length and different lip shape. For each phoneme, we first time-align all of its corresponding video segments to a uniform length, which is the length of the longest video segment of this phoneme. Then we time-align each viseme using cubical interpolation over its appearance parameters. Finally we take the mean of all visemes corresponding to this phoneme, and use the mean as the viseme for this phoneme.

Distance Matrix Generation. The distance between two visemes is defined as a weighted sum of the absolute differences between all the appearance parameters over all the frames. The first 3 appearance parameters are assigned with a weight of 1 while the last two have a weight of 0.5.

This new viseme representation requires computation of the visemes and the distance matrix for each new database and a new person. However, the computation cost is very small compared to the total cost of the analysis phase. For our Chinese video database with 17376 frames and 66 phonemes, it takes less than 10 seconds in a Pentium III 667MHz PC with 256MB RAM.

Finally the distance from the synthesized video to the target phoneme sequence consists of two parts as in [4]: the phoneme-context distance and the smoothness measurement. For the first part, we use our new viseme representation to compute the distance of two triphones. The second part is computed in the same way as [4].

12.4. Geometry-driven Expression Synthesis

Given the feature point positions of a facial expression, to compute the corresponding expression image, one possibility would be to use some mechanism such as physical simulation [11] to figure out the geometric deformations for each point on the face, and then render the resulting surface. The problem is that it is difficult to model the detailed skin deformations such as the expression wrinkles, and it is also difficult to render a face model so that it looks photo-realistic. We instead take an example-based approach.

Given a set of example expressions, Pighin *et al.* [21] demonstrated that one can generate photo-realistic facial expressions through convex combination. Let $E_i = (G_i, I_i)$, $i=0, \dots, m$, be the example expressions where G_i represents the geometry and I_i is the texture image. We assume that all the texture images I_i are pixel aligned. Let $H(E_0, E_1, \dots, E_m)$ be the set of all possible convex combinations of these examples. Then

$$H\{E_0, E_1, \dots, E_m\} = \left\{ \left(\sum_{i=0}^m c_i G_i, \sum_{i=0}^m c_i I_i \right) \mid \sum_{i=0}^m c_i = 1, c_i \geq 0, i = 0, \dots, m \right\} \quad (12.1)$$

Pighin *et al.* [21] also developed a set of tools so that a user can use it to interactively specify the coefficients c_i to generate the desired expressions.

Notice that each expression in the space $H(E_0, E_1, \dots, E_m)$ has a geometric component $G = \sum_{i=0}^m c_i G_i$ and a texture component $I = \sum_{i=0}^m c_i I_i$. Since the geometric component is much easier to obtain than the texture component, we propose to use the geometric component to infer the texture component.

Given the geometric component G , we can project the G to the convex spanned by $G_0, \dots, G_m$, and then use the resulting coefficients to composite the example images and obtain the desired texture image.

One problem with this approach is that the space of $H(E_0, E_1, \dots, E_m)$ is very limited. A person can have expression wrinkles in different face regions, and the combinatorics is very high. So we subdivide the face into a number of subregions. For each subregion, we use the geometry associated with this subregion to compute the subregion texture image. We then seamlessly blend these subregion images to produce the final expression image.

The algorithm descriptions are restricted to 2D case where the geometry of an expression is the face feature points projected on an image plane. We would like to point out that it is straightforward to extend it to the 3D case.

12.4.1 System Overview

Figure 12.6 is an overview of our system. It consists of an offline processing unit and a run time unit. The example images are processed offline only once. At run time, the system takes as input the feature point positions of a new expression, and produces the final expression image. In the following sections, we describe each function block in more detail.

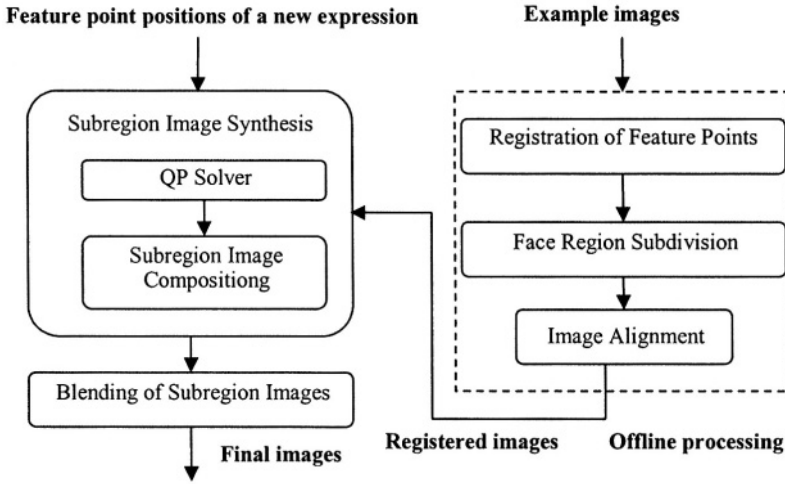


Figure 12.6. Geometry-driven expression synthesis system architecture.

12.4.2 Offline Processing of the Example Images

12.4.2.1 Feature Points

Figure 12.7 (a) shows the feature points that we use in our system. At the bottom left corner are the feature points of the teeth area when the mouth is open. There are 134 feature points in total. Given a face image, it is possible to automatically compute face features [12]. Since the number of example images is very small in our system (10 to 15 examples per person). We choose to manually mark the feature points of the example images.

12.4.2.2 Image Alignment

After we obtain the markers of the feature points, we align all the example images with a standard image which is shown in Figure 12.8 (a). The reason to create this standard image is that we need to have the mouth open so that we can obtain the texture for the teeth. The alignment is done by using a simple triangulation-based image warping, although more advanced techniques [2, 13] may be used to obtain better image quality.

12.4.2.3 Face Region Subdivision

We divide the face region into 14 subregions. Figure 12.7 (b) shows the subdivision. At the bottom left corner is the subregion of the teeth when the mouth is open. The guideline of our subdivision scheme is that we would like the subregions to be small while avoiding expression wrinkles crossing the subregion

boundaries. Since we have already aligned all the example images with the standard image, we only need to subdivide the standard image. We create an image mask to store the subdivision information where for each pixel, its subregion index is stored in its color channel.

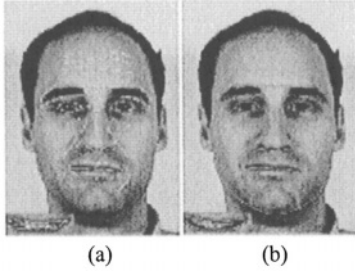


Figure 12.7. (a) Feature points.
(b) Face region subdivision.

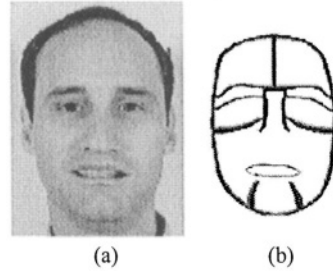


Figure 12.8. (a) Standard image.
(b) Weight map.

12.4.3 Subregion Expression Synthesis

Let n denote the number of feature points. For each example expression E_i , We use G_i to denote the $2n$ dimensional vector which consists of all the feature point positions. Let G be the feature point positions of a new expression. For each subregion R , we use G_i^R to denote the feature points of E_i which are in or at the boundary of R . Similarly we use G^R to denote the feature points of G associated with R .

Given G^R , we want to project it into the convex hull of $G_0^R, \dots, G_m^R$. In other words, we want to find the closest point in the convex hull. It can be formulated as an optimization problem:

$$\text{Minimize: } (G^R - \sum_{i=0}^m c_i G_i^R)^T (G^R - \sum_{i=0}^m c_i G_i^R) \quad (12.2)$$

$$\begin{aligned} \text{Subject to: } & \sum_{i=0}^m c_i = 1 \\ & c_i \geq 0, i = 1, \dots, m \end{aligned} \quad (12.3)$$

Denote

$$g = (G_0^R, G_1^R, \dots, G_m^R) \quad (12.4)$$

and

$$C = (c_0, c_1, \dots, c_m)^T \quad (12.5)$$

Then the objective function becomes

$$C^T g^T g C - 2G^{R^T} g C + G^{R^T} G^R \quad (12.6)$$

This is a quadratic programming formulation where the objective function is a positive semi-definite quadratic form and the constraints are linear. Since G_i^R s are in general linearly independent, the objective function is in general positive definite.

There are multiple ways to solve a quadratic programming problem [15, 28]. In the past decade, a lot of progresses have been made on the interior-point methods both in theory and in practice [28]. Interior-point methods have become very popular for solving many practical quadratic programming problems. This is the approach that we choose to use.

An interior point method works by iterating in the interior of the domain which is constrained by the inequality constraints. At each iteration, it uses an extension of Newton's method to find the next feasible point which is closer to the optimum. Compared to the traditional approaches, interior point methods have faster convergence rate both theoretically and in practice, and they are numerically stable. Even though an interior point method usually does not produce the optimal solution (since it is an interior point), the solution is in general very close to the optimum. In our experiments, we find that it works very well for our purpose.

12.4.3.1 Subregion Image Compositing

After we obtain the coefficients c_i 's, we compute the subregion image I^R by compositing the example images together:

$$I^R = \sum_{i=0}^m c_i I_i^R \quad (12.6)$$

Notice that since the example images have already been aligned, this step is simply pixel-wise color blending.

12.4.4 Blending along the Subregion Boundaries

To avoid the image discontinuity along the subregion boundaries, we do a fade-in-fade-out blending along the subregion boundaries. In our implementation, we use a weight map to facilitate the blending. Figure 12.8 (b) shows the weight map,

which is aligned with the standard image (Figure 12.8 (a)). The thick red-black curves are the blending regions along the boundary curves. The intensity of the R-channel stores the blending weight. We use G and B channels to store the indexes of the two neighboring subregions, respectively. Given a pixel in the blending region, let r denote the value of R-channel, and let i_1 and i_2 be the indexes of the two subregions. Then its blended intensity is

$$I = \frac{r}{255} * I^{i_1} + (1 - \frac{r}{255}) * I^{i_2} \quad (12.7)$$

Notice that we do not perform blending along some of the boundaries where there is a natural color discontinuity such as the boundary of the eyes and the outer boundary of the lips.

After blending, we obtain an image which is aligned with the standard image. We then warp the image back so that its feature point positions match the input feature point positions, thus obtain the final expression image.

12.4.5 Teeth

Since the teeth region is quite orthogonal to the other regions of the face, we use a separate set of examples for the teeth region. In our current system, only a small set of examples for the teeth region are used since we are not focusing on the speech animations where there are a lot of variations on mouth shapes.

12.4.6 Applications

Our technique can be used to enhance many existing facial animation systems which can generate the geometric information of the facial expressions such as expression mapping, physically-based approaches, parameter control methods, etc. In the following, we describe two of the applications that we have experimented with.

12.4.6.1 Enhanced Expression Mapping

Expression mapping technique (also called performance-driven animation) [5, 13, 27, 19] is a simple and widely used technique for facial animations. It works by computing the difference vector of the feature point positions between the neutral face and the expression face of a performer, and then adding the difference vector to the new character's face geometry. One main drawback is that the resulting facial expressions may not look convincing due to the lack of expression details.

Our technique provides a possible solution to this problem in the situation where we can obtain the example images for the new character. The example

images may be obtained offline through capturing or designed by an artist. At the run time, we first use the geometric difference vector to obtain the desired geometry for the new character as in the traditional expression mapping system. We then apply our synthesis system to generate the texture image based on the geometry. The final results are more convincing and realistic facial expressions.

12.4.6.2 Expression Editing

Another interesting application of our technique is on interactive expression editing. One common approach to designing facial expressions is to allow a user to interactively modify control point positions or muscle forces. The images are then warped accordingly. Our technique can be used to enhance such systems to generate expression details interactively.

We have developed a system that allows a user to drag a face feature point, and the system interactively displays the resulting image with expression details. Figure 12.9 is a snapshot of the expression editing interface where the red dots are the feature points which the user can click on and drag.

The first stage of the system is a geometry generator. When the user drags a feature point, the geometry generator figures out the "most likely" positions for all the feature points. For example, if a user drags a feature point on the top of the nose, the entire nose region will move instead of just this single point. The inference is based on a hierarchical principal component analysis of the feature point positions for a number of example expressions. We typically use 30-40 example expressions for the geometry generator. The details are omitted here. The second stage of this system is our expression synthesis system which generates the expression image from the feature point positions.

12.4.7 Results

We show some experimental results for two faces: a male and a female. For each person, we capture about 30-40 images with whatever expressions they can make. We then select the example images, and the use the rest of the images as the ground truth to test our system.

Figure 12.10 (a) shows the example images for the male. The teeth examples are shown in Figure 12.11. Figure 12.12 (a) is a side-by-side comparison where the images on the left column are ground truth while the images on the right are the synthesized results. We would like to point out that each of the expressions in Figure 12.12 (a) is different from the expressions in the examples. But the results from our system closely match the ground truth images. There is a small blurriness in the synthesized images because of the pixel misalignment resulted from the image warping process.

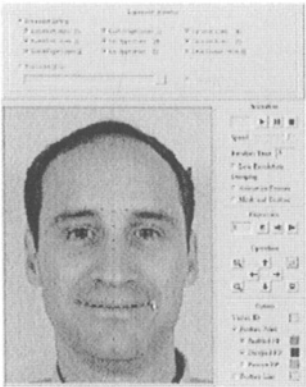


Figure 12.9. Expression editing interface.

Figure 12.10 (b) shows the example images of the female. Figure 12.12 (b) is the side-by-side comparison for the female where the ground truth images are on the left while the synthesized images are on the right. Again, the synthesized results match very well with the ground truth images.

Next we show the results of the expression mapping enhanced with our facial expression synthesis system. Figure 12.12 (c) shows some of the results of mapping the female's expressions to the male. The female's expressions are the real data. The images on the right are the results of the enhanced expression mapping. We can see that the synthesized images have the natural looking expression details.

Figure 12.13 shows some of the expressions generated by our expression editing system. Notice that each of these expressions has a different geometry than the example images. Our system is able to produce photo-realistic and convincing facial expressions.

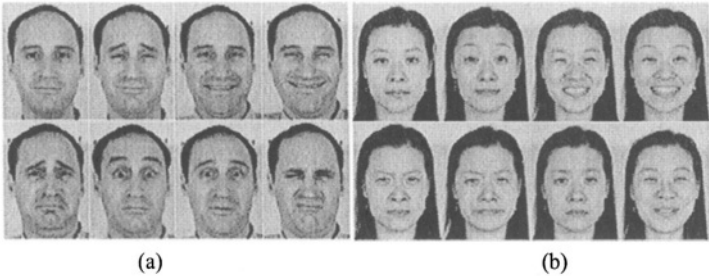


Figure 12.10. Example images of (a) the male; (b) the female.

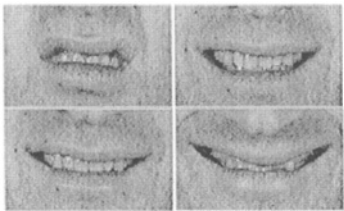


Figure 12.11. Teeth example images.

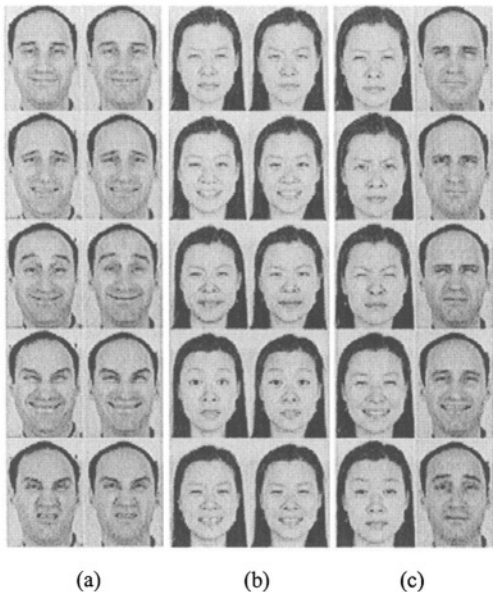


Figure 10.12. Side-by-side comparison with ground truth for (a) the male; (b) the female. The left column contains the ground truth image. The right column contains synthesis results. (c) Results of the enhanced expression mapping. The expressions of the female are mapped to the male



Figure 12.13. Expressions generated by the expression editing system.

12.5. Conclusion and Future Work

In this chapter, we present a system called E-Partner for building photo-realistic conversation agent, who can talk with the user about a given topic. To facilitate conversational interactions, we have extended existing techniques to create a photo-realistic talking head with a continuous presence. The smooth switch between the listening and talking states makes it more natural even in the user barge-in state. The new definition of the viseme makes it possible to have our agent speak Chinese. With this definition, we also achieve optimal synthesized result for a given person.

To make conversation agent more natural looking, we have presented a geometry-driven facial expression synthesis system. Given the feature point positions of a facial expression, our system automatically synthesizes its photo-realistic expression image. We have demonstrated that it works well both for static expressions and for continuous facial expression sequences. Our system can be used to enhance many existing facial animation systems such as expression mapping which generates the geometric information for facial expressions. We have also demonstrated the expression editing application where the user, while manipulating the feature point positions, can see the resulting realistic looking facial expressions interactively.

In the future, we are planning on improving the computation speed by accelerating the image compositing module. Another area that we would like to improve on is the image alignment so that the resulting images are sharper. Potential solutions include optical flow techniques and better image warping algorithms. To generate expressions with various poses, we currently need to use 3D face models. Another possibility is to extend our technique to synthesize expressions with various poses from examples.

We could potentially take as input the pose parameters as well as the feature point motions, and synthesize the corresponding expression from the database. Another area we would like to work on is to handle lip motions during speech. One potential approach is to combine our technique with the technique presented in [8]. One of our final goals is to be able to take the minimum information, such as the feature points, poses, and phonemes, of the performer and automatically synthesize the photo-realistic facial animations for the target character. This has applications not only in computer graphics but also in the compression of video conferencing. The work presented in this paper is a promising step forward toward this goal.

Acknowledgments

The authors would like to express their gratitude to Harry Shum, Hua Zhong, Changbo Hu, Qingshan Zhang, and Ying Song for their contributions on technical

discussions, system implementation, materials preparation and revision for this chapter.

References

- [1] N. Badler and S. Platt. "Animating acial expressions". In *Computer Graphics, Annual Conference Series*, pp. 245 – 252. Siggraph, August 1981.
- [2] T. Beier and S. Neely. "Feature-based image metamorphosis". In *Computer Graphics, Annual Conference Series*, pp. 35 – 42. Siggraph, July 1992.
- [3] M. Brand. "Voice Puppetry". In *Computer Graphics, Annual Conference Series*, pp. 22 – 28. Siggraph, August 1999.
- [4] C. Bregler, M. Covell, and M. Slaney. "Video rewrite: driving visual speech with audio". In *Computer Graphics, Annual Conference Series*, pp. 353 – 360. Siggraph, August 1997.
- [5] S. E. Brennan. "Caricature generator. M.S. visual studies". Dept of Architecture, Massachusetts Institute of Technology, Cambridge, MA., September 1982.
- [6] E. Cossatto and H. P. Graf. "Photo-realistic talking-heads from image samples". *IEEE Trans. on Multimedia*, vol. 2, no. 3, September 2000.
- [7] P. Ekman and W. V. Friesen. "Manual for the facial action coding system". Consulting Psychologists Press, Inc., 1978.
- [8] T. Ezzat, G. Geiger, and T. Poggio. "Trainable video realistic speech animation". In *Computer Graphics, Annual Conference Series*, pp. 388 – 398. Siggraph, August 2002.
- [9] T. Ezzat and T. Poggio. "Miketalk: a talking facial display based on morphing visemes". In *Proc. Computer Animation*, pp. 75-84. June 1998.
- [10] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. "Making faces". In *Computer Graphics, Annual Conference Series*, pp. 55-66. Siggraph, July 1998.
- [11] Y. Lee, D. Terzopoulos, and K. Waters. "Realistic modeling for facial animation". In *Computer Graphics, Annual Conference Series*, pp. 55 – 62. Siggraph, August 1995.
- [12] S. Z. Li and L. Gu. "Real-time multi-view face detection, tracking, pose estimation, alignment, and recognition". In *IEEE Conf. on Computer Vision and Pattern Recognition Demo Summary*, 2001.
- [13] P. Litwinowicz and L. Williams. "Animating images with drawings". In *Computer Graphics, Annual Conference Series*, pp. 235 – 242. Siggraph, August 1990.
- [14] Z. Liu, Y. Shan, and Z. Zhang. "Expressive expression mapping with ratio images". In *Computer Graphics, Annual Conference Series*, pp. 271 – 276. Siggraph, August 2001.
- [15] D. G. Luenberger. "Linear and nonlinear programming". Addison-Wesley Publishing Company, 1984.
- [16] N. Magneneat-Thalmann, N. E. Primeau, and D. Thalmann. "Abstract muscle actions procedures for human face animation". *Visual Computer*, vol. 3, no. 5, pp. 290–297, 1988.

- [17] J. yong Noh and U. Neumann. "Expression cloning". In *Computer Graphics, Annual Conference Series*, pp. 277 – 288. Siggraph, August 2001.
- [18] I. Pandzic, J. Ostermann, and D. Millen. "User evaluation: synthetic talking faces for interactive services". *The Visual Computer*, vol. 15, pp. 330-340, 1999.
- [19] F. I. Parke and K. Waters. *Computer Facial Animation*. AKPeters, Wellesley, Massachusetts, 1996.
- [20] K. Perlin and A. Goldberg. Improv: "A system for scripting interactive actors in virtual worlds". In *Computer Graphics, Annual Conference Series*, pp. 205 – 216. Siggraph, August 1996.
- [21] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. "Synthesizing realistic facial expressions from photographs". In *Computer Graphics, Annual Conference Series*, pp. 75 – 84. Siggraph, July 1998.
- [22] S. M. Seize and C. R. Dyer. "View morphing". In *Computer Graphics, Annual Conference Series*, pp. 21 – 30. Siggraph, August 1996.
- [23] D. Terzopoulos and K. Waters. "Physically-based facial modelling and animation". *Journal of Visualization and Computer Animation*, vol. 1, no. 4, pp. 73 – 80, March, 1990.
- [24] S. Toelg and T. Poggio. "Towards an example-based image compression architecture for video-conferencing". In MIT Technical Report No. 1494, 1994.
- [25] Y. Wang. "A robust parser for spoken language understanding". In *Proceedings of Eurospeech'99*, 1999.
- [26] K. Waters. "A muscle model for animating three-dimensional facial expression". *Computer Graphics*, vol. 22, no. 4, pp.17 – 24, 1987.
- [27] L. Williams. "Performace-driven facial animation". In *Computer Graphics, Annual Conference Series*, pp. 235 – 242. Siggraph, August 1990.
- [28] Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley, 1997.
- [29] B. Zhang, C. Hu, Q. Cai, B. Guo, H. Shum. "E-partner: a photo-realistic conversation agent". In *Proc. of Second IEEE Pacific-Rim Conference on Multimedia (PCM)*, 2001.
- [30] Q. Zhang, Z. Liu, B. Guo, and H. Shum. "Geometry-driven photorealistic facial expression synthesis". In *Proc. of ACM Symp. on Computer Animation*, San Diego, CA, July, 2003.

Chapter 13

3D SEISMIC VOLUME VISUALIZATION

Chunyu Ma¹ and Jon Rokne²

1. Veritas GeoServices, Calgary, Alberta, Canada

2. Department of Computer Science, The University of Calgary, Calgary, Alberta, Canada

Chunyu_Ma@veritasdgc.com, rokne@cpsc.ucalgary.ca

Abstract The effective visualization of 3D seismic data-volumes is of great interest to professionals in the subsurface resource extraction industries as well as earth scientists researching earthquakes and other subsurface phenomena. Because of the steadily increased speed and storage capacities of computing devices this visualization has become a possibility, even approaching real-time performance. In this chapter we therefore discuss methods for 3D visualization of data gathered from seismic exploration programs after it has been processed into a 3D seismic datavolume. The problem of identifying subsurface structures of interest to the professionals is considered in some detail and an general overview of 3D volume visualization is provided.

Keywords: Seismic exploration, SEG-Y format, sealed seismic trace, voxel, iso-surface, texture mapping, horizon picking, fault determination, pulse skeleton, neural network, raycasting, volume rendering, opacity

13.1. Introduction

In this chapter we aim to discuss how geological structures, hidden from view, can be visualized from data obtained by seismic techniques using graphics and image processing tools.

The chapter first provides a short introduction to seismic exploration. The way seismic data is collected and processed is emphasized. The final result of a seismic exploration program with attendant post-processing of the data gathered, is a 2D seismic data set or a 3D seismic data volume depending on the type of seismic survey performed.

The main objective of this chapter is to discuss the currently available tools for the display of 3D seismic data once it has been processed into a seismic data-volume using the tools available in image processing and computer graphics.

The data volume ultimately has to be interpreted by geologists and geophysicists [36, 40]. The quality of their interpretation depends on their experience and knowledge, but it is also dependent on how the data volume is presented to them. It is this latter issue that is dealt with in this chapter.

In section 13.2 some introductory material for the understanding of seismic exploration is presented in an overview suitable for the present chapter including a discussion of the data formats used by the seismic industry. The two main methods for displaying the 3D seismic data are discussed in section 13.3 including the current state of 3D horizon picking. Some examples of 3D seismic visualizations are given.

As a conclusion to the introduction it is appropriate to note that there are many open problems in the visualization of seismic data. A quote from [31] indicates one particular problem:

Despite all of the advancements in graphic processing technology, visualization of multi-attribute seismic data in a meaningful way remains an outstanding challenge.

13.2. Seismic Exploration

13.2.1 Overview of the Seismic Exploration Tool

The interior of the earth is largely hidden from view. Some knowledge about the interior structure close to the earth's surface can be gained by drilling holes and bringing up samples through the drill stem and/or sending instrumentation down through in the hole. Further knowledge can be gained by looking at exposed rock outcrops and inferring that what can be learned from these rock might continue under ground. The information gained by drilling holes is limited and the extrapolating from outcrops can only provide general trends. Other means for exploring the earth have therefore been developed. The main tool is seismic exploration. Seismic exploration, also called seismic surveys, is an excellent tool for gaining a better understanding of the Earth's interior, especially at depths not exceeding a few kilometers.

The basic principle of seismic exploration is to create a disturbance on the surface of the earth. This disturbance is then propagated to the interior of the Earth. At various points in the interior the disturbances are reflected and transmitted. The reflections are then propagated back to the surface of the Earth where they are recorded. The disturbances may be generated by a variety of means such as explosions, banging, compressed air etc. The disturbances are then propagated as pressure waves. If the Earth were a homogeneous medium the waves would travel away from the disturbance and then slowly disperse. This is not the case since the subsurface of the Earth consists of layers of different materials. Upon encountering the interface between two different materials the waves are both propagated and reflected obeying Snell's law at the interfaces.

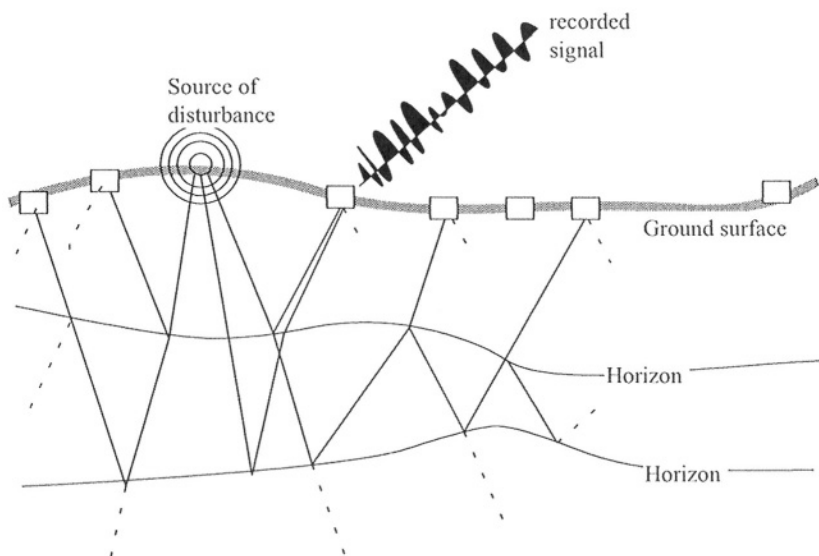


Figure 13.1. General setup for seismic exploration.

The data gathered from the seismic exploration is used to identify subsurface stratigraphy, that is, the relative positions of layers of rocks.

In practice a seismic exploration is configured as shown in Fig. 13.1 where the *source* or *shotpoint* is the origin of the disturbance, the *horizons* the interfaces between different materials and the *receivers* also known as *geophones* are instruments for receiving the reflected waves from the horizons.

In the past seismic exploration and interpreting was carried out using what is known as a 2D methodology. This meant that recorders were placed along a line from a shot-point for a distance of several thousands of meters (this is known as *common shot gather*). An alternative method was to place seismometers in a borehole distributed between the top and the bottom of the hole (*vertical seismic profile*).

The information content in a 2D seismic survey is limited to a vertical plane intersecting the seismic array and the interpretation of the geology is similarly limited to that section. In order to improve on this, repeated 2D seismic sections can be developed so that a kind of 3D interpretation becomes possible. Following this trend the notion of a 3D seismic exploration program was developed. Here the seismometers are placed in a rectangular array gathering data from a 3D volume of the earth. The aim of this chapter is to discuss the current methods for the effective display of such data volumes so that the data can be examined, analyzed and interpreted by geology and geophysics professionals.

The current state of 3D visualization is indicated in [17] where a list of 131 visualization centers is given as well as information on 40 companies providing software and equipment for 3D seismic visualization. The current trend towards volume visualization was noted in an interview in [17] where Phil Hogson stated:

It happens every day - seeing people who were skeptical about 3D visualization convert and see the light. 3D understanding is a personal and often emotional thing. Visualization is all about "seeing is believing" and many people only become converts when it is their own data that suddenly looks different.

13.2.2 Data Preparation

Seismic data are normally recorded in a special format, the most common being the SEG-Y format. After the data has been recorded, it is processed through the three main stages of deconvolution, stacking and migration [55, 15], resulting in a post-processed data volume. Deconvolution acts on the data along the time axis and increase temporal resolution by filtering and trace correction. Stacking compresses the data volume in the offset direction and yields the planes of stacked sections. Migration then moves dipping events to their true subsurface positions and collapses diffractions, thus increasing lateral resolution. After having been subjected to these processing stages, the post-processed data volumes are ready for visualization. This chapter is only concerned with the visualization of the post-processed seismic data and the processing of seismic data from the recorded data to the data for visualization is therefore assumed have been done elsewhere. It should be noted, however, there are a number of possible processing steps that might or might not be applied. The resulting data quality depends strongly on these steps being executed properly. Information present in the recorded data might easily be lost due to poor data processing. The final data volume will also only contain data that was available at the data gathering stage so that if the seismic field program is poorly executed then no postprocessing will be able to generate data that was not recorded by the seismometers. A general discussion of the processing involved is given in [15].

The SEG-Y format is one of several tape standards developed by the Society of Exploration Geophysicists (SEG) in 1973. It is the most common format used for seismic data in the exploration and production industry. SEG-Y was originally designed for storing a single line of seismic data on IBM 9-track tapes attached to IBM mainframe computers. Most of the variations in modern SEG-Y formats result from trying to overcome these limitations. The official standard SEG-Y consists of the following components:

- A 3200-byte EBCDIC descriptive reel header record which is equivalent to 40 IBM punchcards.

- A 400-byte binary reel header record containing much information about the data.
- Trace records consisting of a 240-byte binary trace header and trace data.

As noted, seismic data is acquired by generating a loud sound at one location and recording the resulting rumblings at another location using specially constructed receivers. The source or shot which generates the sound is typically an explosion or vibration at the Earth's surface (land or sea). Each shot is recorded by many receivers laid out in regular patterns. Generally a line of shots is fired. If one line is recorded, the data is a 2D survey, and if more than one line is recorded, the data is a 3D survey. The object of recording is to infer geological subsurface structure from the strength (amplitude) of the recorded signal at different times in the recording.

A trace begins is the recorded signal amplitude from one receiver. In trace, the recording is sampled at some discrete interval, typically around 4 milliseconds, and the duration of a trace is typically 4 or more seconds. After the initial recording, trace is processed in any number of ways. This processing usually changes the absolute amplitudes such that amplitude units are irrelevant, and only relative amplitudes are significant. Also the trace may reflect a logical ordering different from the original (shot, receiver) pair. But in the end, seismic data is almost always stored as a sequence of traces, each trace consisting of amplitude samples for one location (physical or logical).

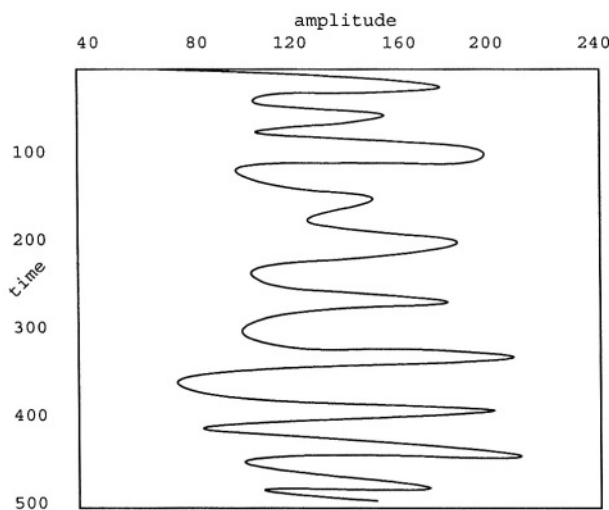


Figure 13.2. A scaled seismic trace.

To adapt SEG-Y for 3D surveys, a line number field is often added somewhere in the trace header. Programs that use values from the SEG-Y trace header usually allow the user to specify the byte location and length of the values.

The trace data is most often in 32-bit IBM floating point format. Occasionally 32-bit IEEE floating-point format is used. Fig. 13.2 shows one sample trace, where the amplitudes are scaled to 8-bit (0-255).

The distance between the source and receiver is termed offset. The point halfway between the source and receiver is the midpoint. After the raw seismic data, i.e., 3D prestack data collapses the offset axis, the result is a data volume V (*inline, crossline, time/depth*) called poststack data which is ready for visualization [32]. A volume data item consists of discrete data at a point (x_i, y_i, z_i) . The data samples are normally taken at equally spaced points in x , y , and z directions forming a *data cube*, as shown in Fig. 13.3.

At each point in the data cube there is a value, or amplitude, representing the migrated seismic data

$$\begin{aligned}x_i &= x_0 + i\Delta x, \\y_j &= y_0 + j\Delta y, \\z_k &= z_0 + k\Delta z, \\V_{ijk} &= V(x_i, y_j, z_k)\end{aligned}$$

where V_{ijk} is the scale value at grid (or voxel) x_i, y_j, z_k or vector of data in the case of multi-attribute seismic data. In seismic visualization, V_{ijk} can be the value of the reflection amplitude or it can be other attributes. Most visualizations focus on single-attribute displays however, multi-attribute visualization is of great interest and the focus of current research [31].

The 3-D array of voxel values that corresponds to equally spaced samples is called a structured data set, because the information about where each sample is located in space is implicit in the data structure.

The 3D data bases frequently contain more voxel data than can be stored in a computer memory. The data is therefore often *bricked*, that is, divided into subvolumes where each component has a size which is a power of 2 [53].

13.3. Visualizing Volume Data and Computer Graphics

Data visualization has only recently been defined as a field. It has to a large part developed as an offshoot of computer graphics as can be seen from the title of one of the new journals in the area: *IEEE Transactions on Visualization and Computer Graphics*. Initially it was focusing on the visualization of 2D data. Recently, however, an increased interest in the visualization of volume data has been witnessed which can be seen from the trends in paper submissions to the visualization conferences (see: <http://vis.computer.org/vis2003/>). Examples of

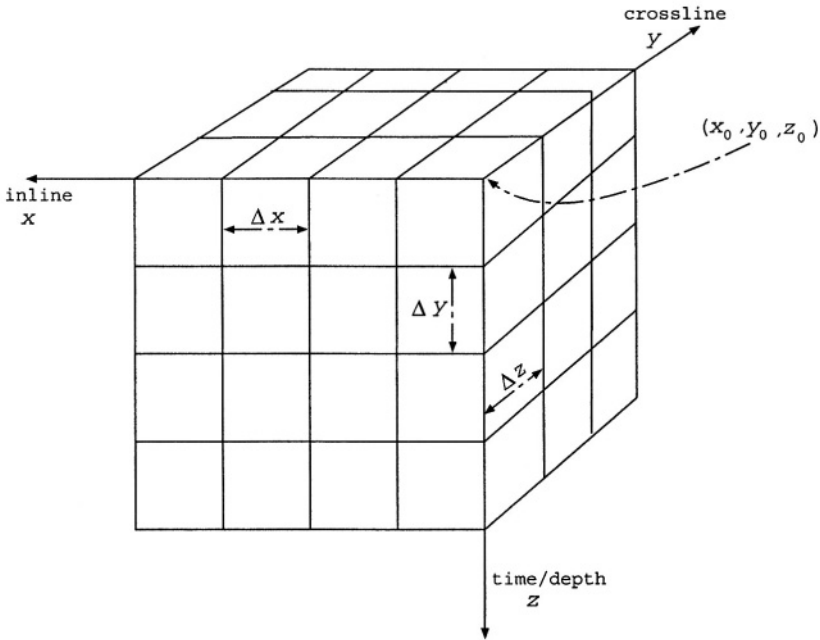


Figure 13.3. A 3D seismic structured data volume.

such data can be found in MRI (magnetic resonance imaging), CT (computer tomography) scans, fluid dynamics, combustion in addition to the 3D seismic data considered here. The commonality between these fields is that they require 3D visualization for interpreting and understanding the volume datasets. Previous techniques were only able to accomplish this to a limited extent [9].

The two main classes of techniques for 3D volume visualization are map-based techniques and volume rendering techniques. The techniques for both of these classes are useful for developing an initial understanding of the structural and stratigraphic features of a 3D seismic data volume with the aim of identifying an oil reservoir. The techniques are also applicable to oil reservoir management. As wells are drilled wireline logs, well testing results and other information becomes available. This information can be used to improve the accuracy of the 3D volume information gained from seismic exploration. Some examples of this are given in [2, 34].

13.3.1 Map-based Volume Visualization

The map-based techniques for 3D volume visualization are the currently most widely used techniques for displaying 3D seismic volume data. This is due to

the simplicity of the display algorithms resulting in fast computational speed of the displaying process. With the use of map-based techniques it possible to achieve a kind of real-time visualization. The two main map-based techniques are texture mapping of slices and iso-surface generation. The texture mapping technique consists of selecting a plane through the data volume intersecting a subset of the volume elements as well as an associated color table for displaying the data. For each vertex on the intersection plane a texture image is created according the data value. The resulting image is then displayed on the computer screen where an interpreter can evaluate the structures and strata displayed.

This technique is fast since it reduces the 3D mapping problem to a sequence of 2D problems. Furthermore, due to speed by which the slices can be generated it is possible to display a sequence of slices creating a kind of real-time display of the 3D data [48].

From the cognitive point of view a human is often better able to interpret and process 2D information as opposed to 3D information. This technique has therefore remained a valuable and popular tool for seismic visualization.

Texture mapping is a technique used widely in computer graphics [22, 14]. A texture is a pattern or an image and texture mapping consists of mapping the pattern onto the surface of a 3D object. From the point of view of 3D seismic data visualization it is only necessary to consider textures mapped from a rectangular patch onto the face of a data element. The textures are now also used to indicate data values. If the data values are scaled to 8-bit integers with values in the range 0-255, then the values can be used to determine a item in a look-up table. The color table contains 3 one-dimensional arrays $r(n)$, $g(n)$ and $b(n)$ each having 256 elements. For any scaled amplitude value n ($0 \leq n \leq 255$) the corresponding RGB value can be found from the table. These values are then assigned to the texture image array. In Figure 13.4 conventionally used color spectra and their names are displayed (courtesy of Veritas GeoServices, Calgary). A general discussion of color as applied to multi-attribute seismic data is given in [31].

When implementing the map-based technique problems occur that are due to the bricking of the data. A further issue relating to the handling of the data occurs when the data is displayed with multiple texture surfaces in one volume.

An example of the display of a textured multi-sliced volume is given in Figure 13.5.

A second map-based technique is based on iso-surface generation. An iso-surface consists of a set of points (x, y, z) in a given data volume satisfying

$$V(x, y, z) = C$$

where C is the iso-surface value and $V(x, y, z)$ is the value of the seismic response at point (x, y, z) . It is assumed that the data is available on rectangular cells stacked in a regular fashion. Conceptually the method processes one cell

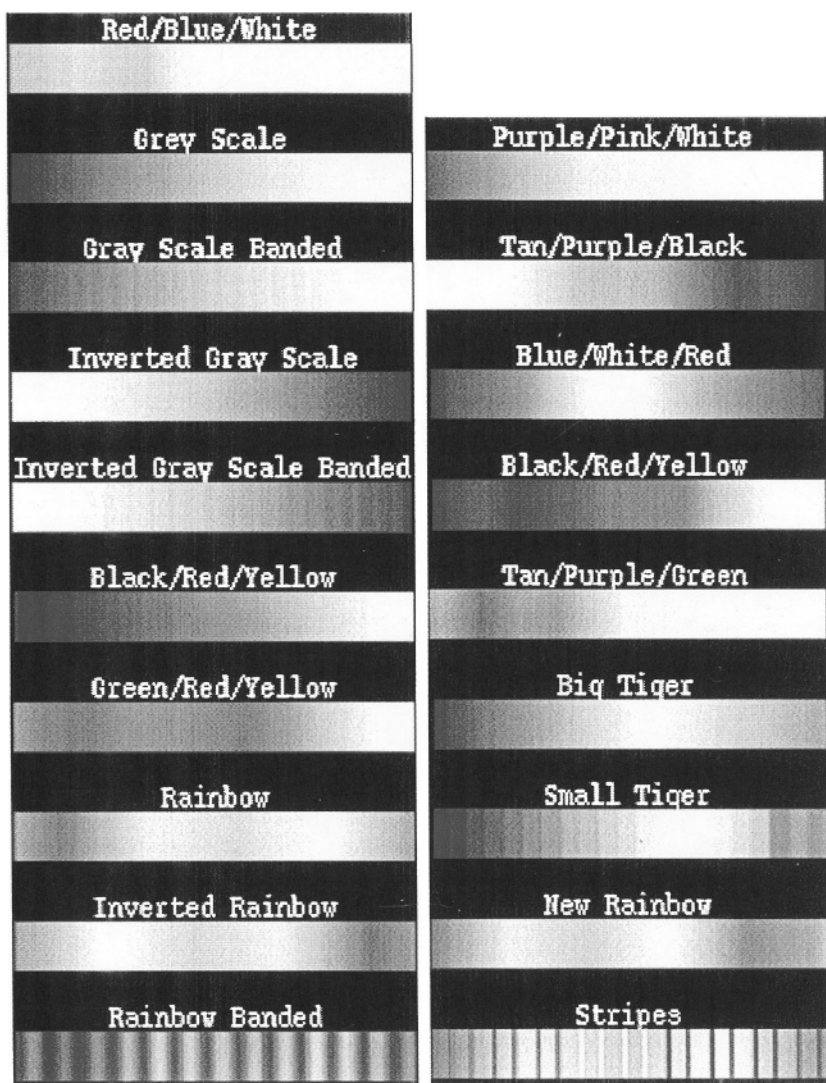


Figure 13.4. Conventionally used color spectra.

at a time. Each cell is topologically equivalent to a unit cube. The description of the method can therefore be simplified to the extraction of an iso-surface within a unit cube. From the values of the 8 cube vertices and the data at the vertices

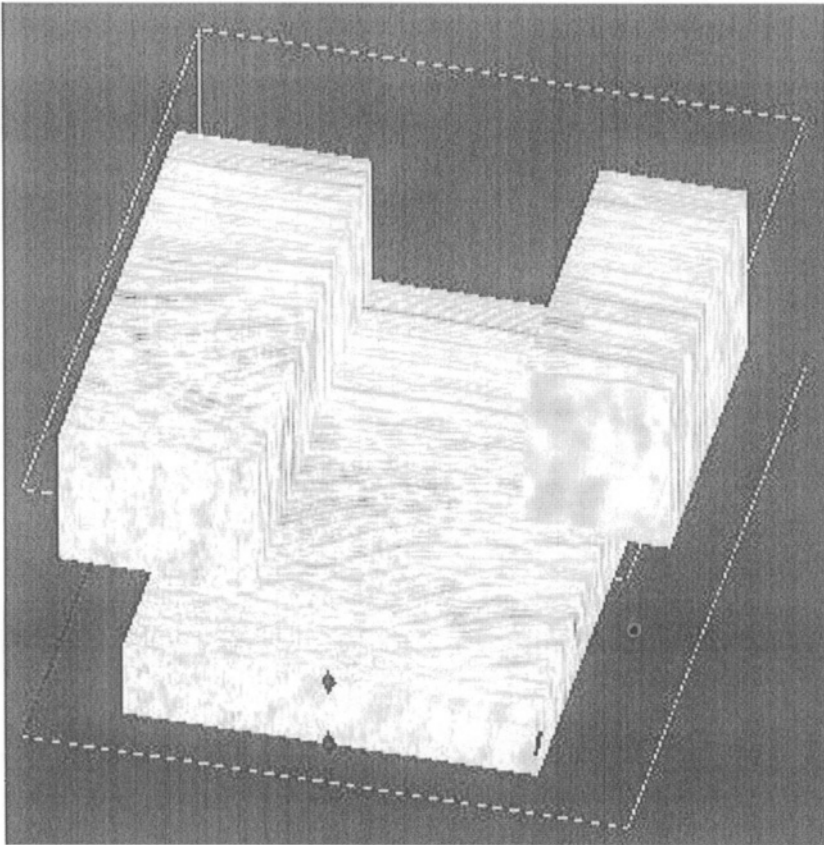


Figure 13.5. An example of sliced and textured volume.

compared with the threshold value C it follows that there are 256 scenarios for the interpolated iso-surface within the cube. This surface can be constructed by trilinear interpolation of the values at the cube vertices. The intersections of the surface with the cube edges are calculated by inverse interpolation and the values at these points can also be used to generate a triangle which is a simplistic estimate of the surface within the cube. This algorithm can be modified for seismic visualization purposes.

One of the most commonly used techniques for iso-surface generation is the marching cube algorithm [35]. It has proven to be very effective in combination with fast triangle rendering hardware as is provided on Silicon Graphics workstations, and more recently on PC graphics boards supporting OpenGL. Improvement on the marching cube algorithm has been studied by many authors.

Howie and Blake [23] suggested a mesh propagation algorithm for iso-surface construction. Itoh and Koyamada [24] made a further improvement on the mesh propagation algorithm using an extrema graph and sorted boundary cell lists for automatic iso-surface propagation. An efficiency enhanced iso-surface generation algorithm for volume visualization was proposed by Li and Agathoklis [30]. In [16] iso-surface generation using hierarchical octree data structure was used for speed-up the marching cube algorithm to achieve interactive rendering. Another algorithm, the so-called *splitting-box algorithm* by Muller and Stark [39], was used for the adaptive generation of surfaces in volume data.

By *horizon picking* is meant determining a connected set of reflections of comparable amplitude and two-way traveltimes appearing persistently from seismic trace to seismic trace [19]. In practical terms time tagged amplitude data has been obtained at a set of positions covering a specific area. The data are stored in row, according to the time-tag, with amplitude at each position being stored in the appropriate column (trace) and array (inline and crossline). The horizon to be picked is assumed to run through the volume data as a surface, and the algorithm must select the appropriate depth value for each trace in the volume data.

The application of the marching cube algorithm to seismic horizon picking is still an open problem, since it is similar to iso-surface extraction, but not exactly the same. For example, horizon picking requires extracting local maximum or local minimum for each trace, instead of an iso-value in the volume data. Therefore, we cannot apply iso-surface extraction algorithm directly for horizon picking. Extracting faults is even more complicated than horizon picking [27]. On a fault surface, all of the voxels have local maximum gradient from trace to trace. For these reasons, the marching cube algorithm is not directly adopted, however, the mesh propagation algorithm mentioned above is useful for horizon surface generation and it is simply based on the amplitudes from trace to trace with a so-called *one-step look-ahead* algorithm.

An FIFO queue is used for propagation from the central cell to its neighbor cells. Triangles are constructed as the mesh propagates forward, and the horizon surface expands from the seed cell to other cells. While all the voxels have a constant value on an iso-surface. This difference is not essential as far as the marching cubes algorithm is concerned which means that it can be used for horizon picking after having been suitably modified. The mesh propagation algorithm for iso-surface generation [23, 24] reduces the computation cost by avoiding visiting non-intersected cells. This can also be exploited for horizon picking. It is claimed that the algorithm may also work for irregular 3D grids [23]. The height of each cell is set to be the window size along traces (i.e. z-axis). The point of intersection is the local maximum on the edges along the four traces. The traversal is performed using an FIFO queue, initialized with a seed picked by an operator.

Again, the stop conditions include the threshold value of amplitude at which picking is to stop, or a volume boundary is met. The triangulation process is similar to the triangulation used in the marching cube algorithm. It is not clear, however, if the unit normal for each triangle vertex can be calculated the same as in marching cubes algorithm [35], i.e., by estimating the gradient vector at the surface using central differences along the three coordinate axes. One possibility is to calculate the normal using the coordinates of the picked voxels [11].

It should also be noted that seismic data is typically noisy and imprecise, hence horizon picking is difficult, and there is no completely satisfactory algorithm, as yet, for "fully automatic" horizon picking.

Horizons usually have a high amplitude relative to other regions in seismic section, and display lateral continuity with small variation in arrival traveltime from trace to trace except where faulting or other geophysical anomalies occur. The most basic method of horizon picking is based on detecting the characteristic high amplitude of the reflection wavelet. Amplitude thresholding of the data is useful for extracting horizons where they are easily separable from the rest of the data on an amplitude basis. The problem is that there is no obvious level at which horizon can be distinguished from noise on an amplitude basis alone in an area where signal to noise ratio becomes poor. Some improvement to thresholding was suggested by detecting amplitude maxima [19]. The amplitude maxima are obtained by finding the zero-crossings of the first order derivative with respect to time. Another improvement is to estimate a signal to noise ratio measure based on similarity or coherence in a moving window which is passed over the data. All amplitude based methods rely on the characteristic lateral continuity of horizons and their characteristic high amplitude. Figure 13.6 shows the simplest amplitude based horizon picking method, where the moving window size is 5. The center position of the moving window in the next trace is along the direction of the previous tracked horizon dip as shown by the dashed lines, according to the characteristic of lateral continuity of horizon section. We can interpolate the data in the moving window to obtain more precise horizon positions (i.e., the positions of the local maxima) in each trace. In this figure, the picking is based on searching local maximum trace by trace. Picking can also be based on local minimum or zero-crossing in each trace. If the data is free of noise, this method works well. But noisy data may mislead the path of the moving window, and make the method fail.

Amplitude based methods use only measurements of the reflection amplitudes for event picking, which often yield rather poor results when applied to field data because of the poor signal-to-noise-ratio (SNR) often encountered in seismic data. In fact, amplitude is only one of attributes of a seismic wavelet. If amplitude information is supplemented with other available information for horizon picking, we can expect better results.

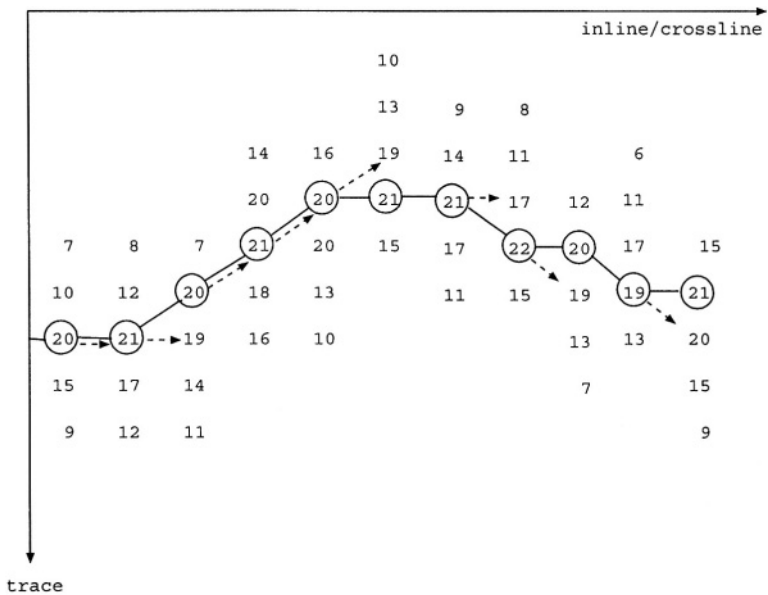


Figure 13.6. Horizon picking with moving window.

A seismic reflection wavelet can be described by many attributes including amplitude and phase information. In other words, by regarding the wavelet signature as a pattern, the horizon picking can be viewed as a pattern recognition process, and hence an ideal candidate for the application of artificial neural networks.

Figure 13.7 displays 16 parameters describing a pulse skeleton. The parameters define measurements of the main and side lobe maximum amplitudes ($|AB|$, $|CD|$, $|EF|$, $|GH|$ and $|IJ|$), half amplitude width ($|h_1|$ to $|h_5|$), and zero cross points ($|Z_1|$ to $|Z_5|$).

These parameters can be used as inputs for a neural network for seismic horizon picking based on the morphology of the wavelet as discussed in [19]. To train the network, each trace (column of training data) is shifted, point by point, across the input nodes, along with its corresponding target data. The input nodes act like a window, which is shifted down each column of data, and taking a number of data samples as the input to the network. Further studies have been made to improve the convergence of the weight matrix of the network. McCormack et al. employed a back propagation neural network (BNN) for seismic event picking and trace editing [37]. A hybrid neural network was proposed by Veezhinathan [51]. The key feature of the hybrid neural network is that it allows incorporation of geological constraints in identifying segments

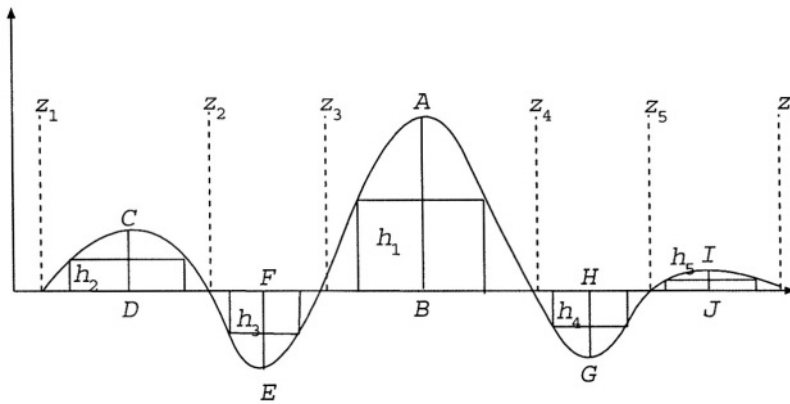


Figure 13.7. Pulse skeleton parameters (from [19]).

that form the horizon. Neural networks was first reported to be integrated with a commercial package in Landmark SeisWorks3D [28].

Using neural network, classification is based on reflection wavelet signature and shape. However, this method does not utilize information regarding traveltimes of arrivals, or the geology of the section. Traces are considered independently, so no knowledge of horizon continuity can be used. From a visual perspective, lateral continuity in traveltimes and amplitude is one of the most apparent characteristics of horizons. Neural networks have also been applied to distinguish waveforms of earthquakes from the waveforms of man-made explosions [46].

Probabilistic data association was first suggested for picking multiple targets by Reid [45]. Traditionally, the application areas of multitarget picking include ballistic missiles, aircrafts, ships and submarines. Recently, multitarget detection/picking method has been suggested for application to pavement profiling [47] and seismic event picking [20, 54, 3, 6, 4, 5, 42, 1]. Multitarget picking theory can be applied to seismic event picking since the objective is similar to the above application areas. Seismic data are obtained from a grid of receivers which record the strengths of signals bouncing off horizons below the Earth's surface. The signal strength and traveltimes depend on the material forming the horizon and its depth below the Earth's surface, and picking a horizon requires the selection the correct depth value at each grid point. This is, in essence, a similar problem to that of picking a target across a cluttered radar screen, with the target being replaced by horizon to be tracked, and the initialization of the picking being replaced by a grid reference and depth of the chosen horizon. The difference is that seismic horizons are static while the aircraft targets on the radar screen are dynamic. For seismic picking, the system dynamics is

replaced by a mathematical model of the shape of the horizon described by a first-order ordinary differential equation.

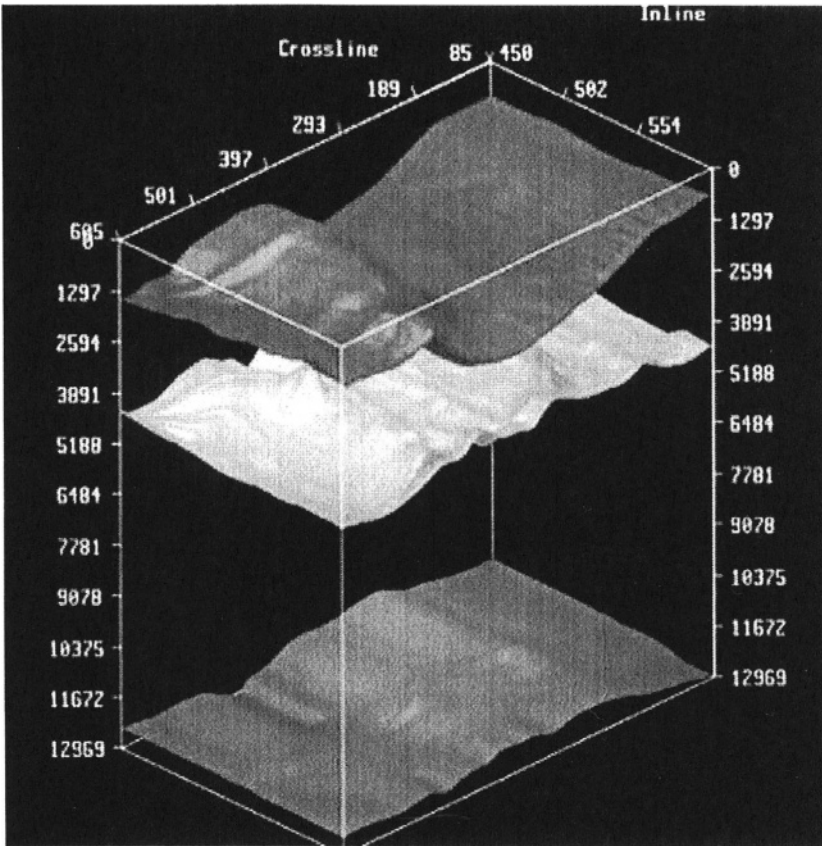


Figure 13.8. An example of a data volume rendered with three horizons courtesy Veritas GeoServices, Calgary.

The image shown in Figure 13.8 shows three horizons picked out of a 3D seismic data volume. In this example the horizons are relatively horizontal and there are no obvious faults disturbing the continuity of the of the surfaces of the horizons.

13.3.2 Volume Rendering Techniques

Volume rendering is a simple, elegant, and universal approach to visualization of scientific and engineering data [9], including geophysical models. Volume rendering combines the tools of image processing with those of 3-

D computer graphics to achieve an unprecedented versatility in display and analysis of objects or phenomena occupying three dimensional space. Volume rendering derives its power from its conceptual simplicity. The object or phenomenon of interest is sampled or subdivided into millions of cubic building blocks, called voxel (or volume elements). Each voxel carries a value of some measured or calculated property of the volume. For example, the voxels might represent mass density measurements taken throughout a computed X-ray tomography (CT) scan of a core sample, or reflectivity amplitudes obtained by seismic surveys.

Ray casting is the most commonly used image-order technique. It simulates optical projections of light rays through the dataset, yielding a simple and visually accurate mechanism for volume rendering. Rays are cast from the viewpoint through screen pixels into the volume [29, 52, 43]. The contributions of all voxels along the rays are calculated and used to find the pixel color. It is illustrated in Figure 13.9.

The basic ray casting algorithm consists of two computational pipelines:

- The visualization pipeline - the data is shaded with a color intensity $c(x)$. A local gradient approximation is used to allocate a voxel normal at each voxel in the data. This normal is substituted into a standard Phong shading model to obtain an intensity, i.e., RGB values at the voxel.
- The classification pipeline - each voxel is associated with an opacity $\alpha(x)$. The opacity is typically a function of the scalar sample value, e.g., the reflection amplitude for seismic dataset. Usually, it is also useful to include the local gradient when the opacity is calculated.

After the values of $c(x)$ and $\alpha(x)$ are calculated, they are combined to provide a final voxel intensity.

The choice of opacity values is of critical importance for the quality of the volume rendering. One system, known as the *zone system* (abbreviated ZS) is outlined in [25]. In this system the 8-bit seismic data is divided into six zones where three positive zones correspond to peak amplitudes and three negative zones to trough amplitudes. The ZS then consists of choosing the appropriate colors for each zone as well as interactively deciding on the zone boundaries that gives the best visual results. The detailed process involved in the ZS is described in [25], however, it is appropriate to quote the paper:

The key to distinguishing features with clarity is the geoscientists' ability to enhance contrast between two features. The primary components in feature enhancement are the base color scale, their color variations at different opacity levels and the background color.

Ray casting offers very high image quality, including the ability to provide visual depth cues such as shaded surfaces and shadows. But these advantages come at the price of high computational requirements. Ray-casters do not access the volume data in the order it is stored because of the arbitrary traversal

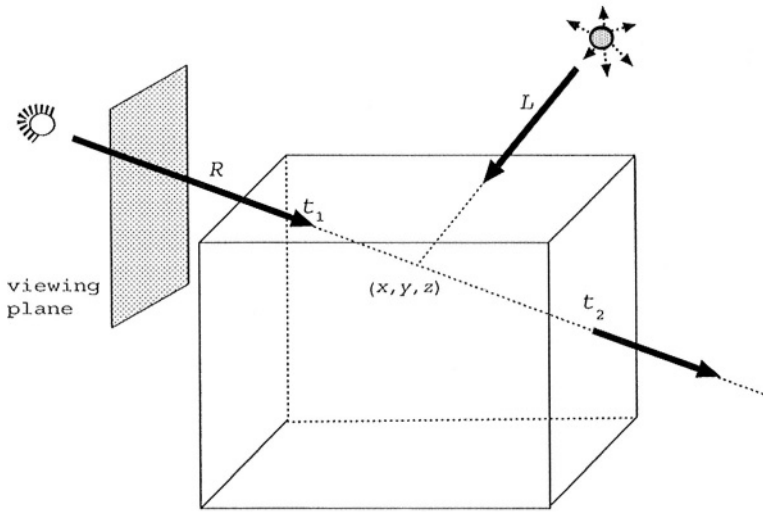


Figure 13.9. Raycasting.

direction of the viewing rays. This leads to poor spatial locality of data references and requires expensive address arithmetic to calculate new sampling locations. There are, however, a number of optimization techniques which can be used to increase the speed with which an image can be rendered [26]. Overviews of the applications of visualization to seismic data are also given in [8, 13]. The VolumePro system [44] is a commercial product created by Mitsubishi Electric in 1999 using some of these techniques, which can fit into a standard PC as a PC-board. The system is based on ray casting approach with many acceleration techniques, such as the use of shear-warp transformations to improve speed of data access [26]. A number of other companies are actively engaged in developing software for 3D seismic visualization, see for example [13, 21, 47, 33].

An example of a 3D volume rendered image using accumulated 2D textures is given in Figure 13.10. The size of the data set from an offshore seismic survey is $2401 \times 1461 \times 300$ voxels. The sea bed surface is clearly displayed in the image.

The techniques outlined above are also used to display data in the so-called *visualization caves* where the 3D data values are displayed in a closed room using data projectors.

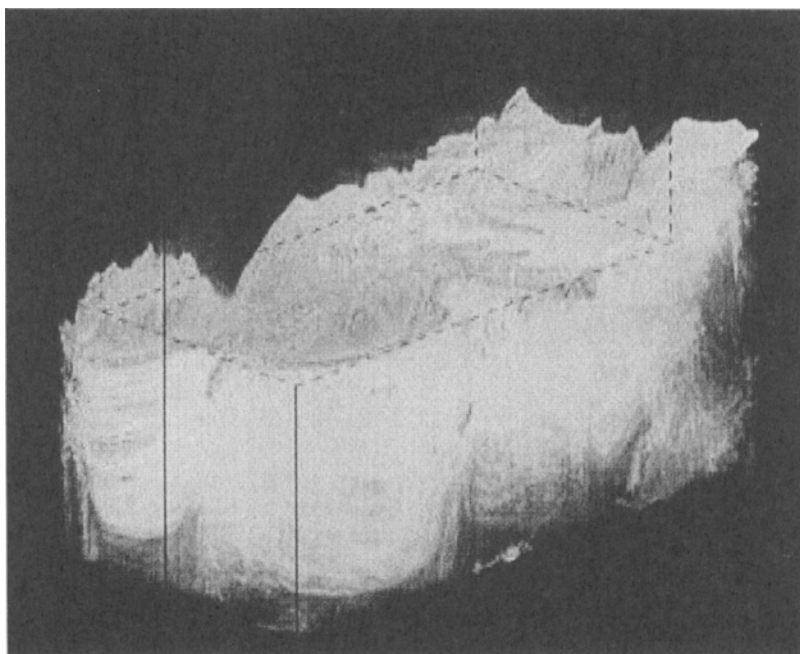


Figure 13.10. Volume rendered image courtesy Veritas GeoServices, Calgary.

13.4. Conclusion

Volume rendering for 3D seismic data volumes is a very effective tool providing an overview of the gross structural and stratigraphic environment. Variation in data quality can be identified, giving the interpreter an idea of the relative difficulty of interpretation in different areas. It is also possible to identify an initial set of seismic horizons to interpret and the manner in which those horizons should be interpreted.

Current research in the area of volume visualization includes *volume illustration* where non-photorealistic rendering is employed to enhance specific features of the data volume [49]. Other avenues of research are in the application of non-linear geometric lenses to 3D volumes, improved algorithms for horizon tracing and fault identification.

Acknowledgments

Thanks are due to Veritas GeoServices, Calgary for images used in this paper.

References

- [1] P. Alberts and K. Moorfeld. "Artificial neural networks for seismic horizon tracking across discontinuities". In: *EAGE 61st Conference and Technical Exhibition*, pp. 6-31, Helsinki, Finland, June, 1999.
- [2] H. A. Al-Faddagh and G. Tobert. *Reservoir Management Emphasizes People, Technology and Innovation*. Saudi Aramco Technology, Spring, 2003, pp. 10-31, 2003.
- [3] N. Bienati, M. Nicoli, and U. Spagnolini. "Automatic horizon picking algorithms for multidimensional data". In: *SEG Meeting, 61st EAGE*, pp. 6 – 30, 1999.
- [4] N. Bienati, M. Nicoli, and U. Spagnolini. "Automatic horizon picking algorithm for multidimensional data". *61st EAGE Meeting*, pp. 6-30, June, 1999.
- [5] N. Bienati, M. Nicoli, and U. Spagnolini. "Horizon picking for multidimensional data: an integrated approach". In: *SEG Meeting, 6th International Congress of the Brazilian Geophysical Society*, pp. SBGf359, 1999.
- [6] N. Bienati, M. Nicoli, and U. Spagnolini. "Horizon picking for multidimensional data: an integrated approach". *6th International Congress of the Brazilian Geophysical Society*, pp. SBGf359, 2000.
- [7] N. Bienati and U. Spagnolini. "Traveltime tracking in 3-D data volume". *60th EAGE Meeting*, 1998.
- [8] F. Bosquet and J. Dula. "Advanced volume visualization - new ways to explore, analyze, and interpret seismic data". *The Leading Edge*, vol. 19, no. 5, pp. 535 - 537, 2000.
- [9] K. Brodlić and J. Wood. "Recent advances in volume visualization". *Computer Graphics Forum*, vol. 20, no. 2, pp. 125 - 148, 2001.
- [10] C. Cary, editor. *OpenGL Volumizer Programmer's Guide*. Silicon Graphics, Inc., 1998.

- [11] Ma, Chunyu. *Visualization of seismic data in 3-D*. M. Sc. Thesis, University of Calgary, 2002.
- [12] G. Dorn. "Modern 3-D seismic interpretation". *The Leading Edge*, vol. 17, pp. 1262 - 1269, 1998.
- [13] G. Dorn, M. J. Cole, and K. M. Tubman. "Visualization in 3-D seismic interpretation". *The Leading Edge*, vol. 14, no. 5, pp. 1045 - 1049, 1995.
- [14] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin and S. Worley. *Texture and Modeling. A Procedural Approach*. Morgan Kaufmann, 2003.
- [15] M. G. Gadallah. *Reservoir Seismology. Geophysics in Nontechnical Language*. PennWell Books, 1994.
- [16] G. Greiner and R. Grosso. "Hierarchical tetrahedral-octahedral subdivision for volume visualization". *The Visual Computer*, vol. 16, pp. 357 - 369, 2000.
- [17] *Visualization Reference for the Oil and Gas Industry*. Gulf Publishing Company, 2003.
- [18] B. A. Hardage. *Vertical Seismic Profiling, Part A: Principles*. Geophysical Press, 1985.
- [19] E. Harrigan. *Seismic Event Tracking*. PhD Thesis, University of Strathclyde, UK, 1992.
- [20] E. Harrigan and T.S. Durrani. "Automatic horizon picking by multiple target tracking". *53rd EAGE Meeting*, pp. 440 - 441, 1991.
- [21] D. Harron. "Horizon autopicking". *The Leading Edge*, vol. 19, no. 5, pp. 491 -492, 2000.
- [22] R S. Heckbert. "Survey of texture mapping". *IEEE Computer Graphics & Applications*, pp. 56 - 67, November, 1986.
- [23] C. T. Howie and E. H. Blake. "The mesh propagation algorithm for isosurface construction". *Computer Graphics Forum*, vol. 13, no. 3, pp. C65 - C47, 1994.
- [24] T. Itoh and K. Koyamada. "Automatic isosurface propagation using an extrema graph and sorted boundary cell lists". *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 4, pp. 319-327, 1995.
- [25] G. Kidd. "Fundamentals of 3-D seismic volume visualization". *The Leading Edge*, vol. 18, no. 6, pp. 702-709, 1999.
- [26] P. Lacroute and M. Levoy. "Fast volume rendering using a shear-warp factorization of the view transformation". In *Proc. of SIGGRAPH 94*, pp. 451 - 458, 1994.
- [27] J. Lee. "Constructing faults from seed pick by voxel tracking". *The Leading Edge*, vol. 18, no. 3, pp. 338-340, 1999.
- [28] M. Leggett, W. Sandham, and T. Durrani. "3D horizon tracking using artificial neural networks". *First Break*, vol. 14, no. 11, pp. 413 - 418, 1996.
- [29] M. Levoy. "Display of surfaces from volume data". *IEEE Computer Graphics & Applications*, vol. 8, no. 5, pp. 29-37, 1988.
- [30] J. Li and P. Agathoklis. "An efficiency enhanced isosurface generation algorithm for volume visualization". *The Visual Computer*, vol. 13, pp. 391 - 400, 1997.
- [31] I-Ching Lin. *Visualization of Multiple Seismic Attributes: Mapping 3-D Multi-attribute Seismic Data into HLS Color Space*. M. Sc. Thesis, Department of Computer Science, University of Houston, 2003.
- [32] C. L. Liner. *Elements of 3-D Seismology*. PennWell Publishing, 1999.
- [33] L. Liro, K. Cline, and M. Kadri. "A new paradigm for data integration and collaboration using 3-D visualization technology". *The Leading Edge*, vol. 20, no. 12, pp. 1372 - 1376, 2001.

- [34] L. Liro, M. Lahr, K. Cline, and et. al. "Application of 3-D visualization to integrated geophysical and geologic model building: a prestack, subsalt depth migration project, deepwater Gulf of Mexico". *The Leading Edge*, vol. 19, no. 5, pp. 466-472, 2000.
- [35] W. E. Lorensen and H. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". *Computers and Graphics*, vol. 21, no. 4, pp. 163-169, 1987.
- [36] D. E. Meyer, E. L. Harvey, T. E. Bulloch, J. C. Vonnannon and T. M. Sheffield. "Use of seismic attributes in 3-D geovolume interpretation". *The Leading Edge*, vol. 20, no. 12, pp. 1377-1381, 2001.
- [37] M. D. McCormack, D. E. Zucha, and D. W. Dushek. "First-break refraction event picking and seismic data trace editing using neural networks". *Geophysics*, vol. 58, no. 1, pp. 67-78, 1993.
- [38] M. Midttun, R. Helland, and E. Finnstrom. "Virtual reality - adding value to exploration and production". *The Leading Edge*, vol. 19, no. 5, pp. 538-544, 2000.
- [39] H. Muller and M. Stark. "Adaptive generation of surfaces in volume data". *The Visual Computer*, vol. 9, pp. 182-192, 1993.
- [40] D. Neff, J. Singleton, J. Layton, and E. Keskula. "Seismic interpretation using true 3-D visualization". *The Leading Edge*, vol. 19, no. 5, pp. 523-525, 2000.
- [41] M. Nicoli. "Hidden Markov model for multidimensional wavefront tracking". *IEEE Trans. Geoscience and Remote Sensing*, vol. 40, no. 3, 2002.
- [42] M. Nicoli, V. Rampa, and U. Spagnolini. "Multitarget detection/tracking based on hidden Markov models". *Proc. IEEE ICASSP*, pp. 3196-3199, 2000.
- [43] H. Pfister. *Architectures for Real-Time Volume Rendering*. PhD thesis, State University of New York at Stony Brook, US, 1997.
- [44] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. "The volumepro real-time ray casting system". In *Proc. of SIGGRAPH 99*, pp. 251-260, Los Angeles, CA, USA, 1999.
- [45] D. B. Reid. "An algorithm for tracking multiple targets". *IEEE Transaction on Automatic Control*, vol. 24, no. 6, pp. 843-854, 1979.
- [46] Y. Shimshoni and N. Intrator. "Classification of seismic signals by integrating ensembles of neural networks". *IEEE Trans. Signal Processing*, vol. 46, no. 5, pp. 1194-1201, 1998.
- [47] U. Spagnolini and V. Rampa. "Multitarget detection/tracking for monostatic ground penetrating radar: application to pavement profile". *IEEE Transaction on Geoscience and Remote Sensing*, vol. 37, no. 1, pp. 383-394, 1999.
- [48] T. J. Stark. "Surface slice generation and interpretation". *The Leading Edge*, vol. 15, no. 7, pp. 818-819, 1996.
- [49] N. A. Svakhine and D. S. Ebert. "Interactive volume illustration". *Pacific Graphics 2003*, to appear.
- [50] G. M. Treece, R. W. Prager, and A. H. Gee. "Regularized marching tetrahedron: improved iso-surface extraction". *Computers and Graphics*, vol. 23, no. 4, pp. 583-598, 1999.
- [51] J. Veezhinathan, F. Kemp, and J. Threet. "A hybrid of neural net and branch and bound techniques for seismic horizon tracking". In *Proc. ACMISIGGRAPH Symposium on Applied Computing*, vol. 58, pp. 173-178. ACM, March, 1993.
- [52] A. H. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley Publication, 1992.

- [53] M. Woo, J. Neider, T. Davis, and D. Shreiner, editors. *OpenGL Programming Guide*. Addison Wesley, third edition, 1999.
- [54] C. Woodham, W. Sandham, and T. Durrani. "3-D seismic tracking with probabilistic data association". *Geophysics*, vol. 60, no. 4, pp. 1088-1094, 1995.
- [55] O. Yilmaz. *Seismic Data Processing, volume 2 of Investigations in Geophysics*. Society of Exploration Geophysicists, P. O. Box 702740, Tulsa, OK 74170-2740, 1994.

Chapter 14

GRAPHICAL REPRESENTATION OF FINGERPRINT IMAGES

Jie Zhou ¹, David Zhang ², Jinwei Gu ¹ and Nannan Wu ¹

1. Department of Automation, Tsinghua University, Beijing 100084, China

2. Department of Computing, The Hong Kong Polytechnic University, Hong Kong

jzhou@tsinghua.edu.cn, csdzhang@comp.polyu.edu.hk, gujinwei98@mails.tsinghua.edu.cn, wunannan99@mails.tsinghua.edu.cn

Abstract Fingerprint recognition is very important in automatic personal identification. In conventional fingerprint recognition algorithms, fingerprints are represented graphically with a simple set of minutiae and singular points, but the information in this kind of representation is not adequate for large-scale applications or where fingerprint quality is poor. A complete and compact representation of fingerprints is thus highly desirable. In this chapter, we focus on research issues in the graphical representation of fingerprints. We first introduce minutiae-based representation and provide some models for the graphical representation of orientation fields. Latterly, we deal with the generation of synthetic fingerprint images and conclude with a discussion of how to establish a complete fingerprint representation.

Keywords: Fingerprint recognition, orientation field, minutiae, synthesis, graphical model, approximation

14.1. Introduction

In recent years, fingerprint identification has received increasing attention. Among the various biometric techniques used for automatic personal identification, automatic fingerprint identification systems are the most popular and reliable. Nonetheless, while the performance of fingerprint identification systems has reached a high level, it is still not satisfactory when applied to large databases or to poor-quality fingerprints [1,2].

A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. Figure 14.1 (a) depicts a fingerprint in which the ridges are black and the valleys

are white. Its *orientation field*, defined as the local orientation of the ridge-valley structures, is shown in Figure 14.1 (b). The *minutiae*, ridge endings and bifurcations, and the *singular points* are also shown in Figure 14.1 (a). Singular points can be viewed as points where the orientation field is discontinuous. They can be classified into two types: a *core*, the point of the innermost curving ridges, and a *delta*, the center of triangular regions where three different directional flows meet. Fingerprints are usually partitioned into six main classes according to their macro-singularities, i.e., arch, tented arch, left loop, right loop, twin loop and whorl (see Figure 14.2). Fingerprints can be represented by a large number of features, including the overall ridge flow pattern (i.e. orientation field), ridge frequency (i.e. density map), location and position of singular points, and the type, direction and location of minutiae points. All of these features contribute to fingerprint individuality.

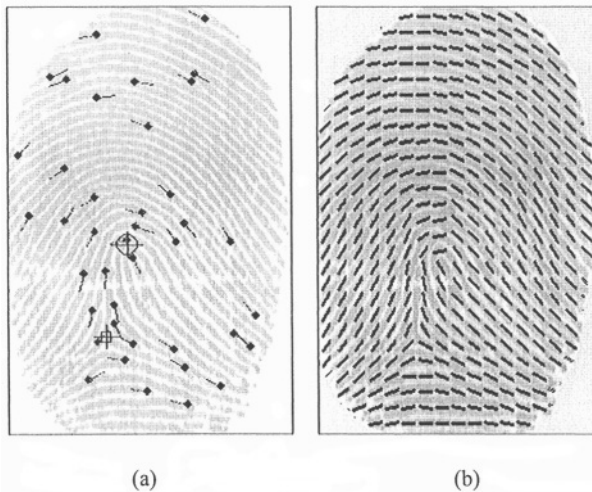


Figure 14.1. Example of a fingerprint: (a) singular points and minutiae with its direction; (b) orientation field shown with unit vector.

The performance of a fingerprint identification system depends mainly on the kind of fingerprint representation it utilizes. Most classical fingerprint recognition algorithms [1,2,3] represent the fingerprint using the minutiae and the singular points, including their coordinates and direction, as the distinctive features. That means fingerprints are first graphically represented with a set of minutiae and singular points are then compared with the template set. If the matching score exceeds a predefined threshold, two fingerprints can be regarded as belonging to the same finger.

Obviously this kind of representation does not make use of every feature that is available in fingerprints and therefore cannot provide enough information for

large-scale fingerprint identification tasks [4]. Certainly, a better representation for fingerprints is needed.

In this chapter, we mainly address the topic of graphical representation of fingerprints. First we introduce the conventional representation of fingerprints, the minutiae-based representation. We then establish a complete representation with a compact form in which much more information other than minutiae features (such as the orientation field) can be taken into account.

The remainder of the chapter is organized as follows. In Section 14.2, we will introduce the minutiae-based representation of fingerprints. In Section 14.3, we will provide four models of fingerprint orientation fields. In Section 14.4, we will describe how to synthesize the fingerprint images utilizing the orientation field models. In Section 14.5, we will discuss the establishment of a complete and compact fingerprint representation. The conclusions are drawn in the last section.

14.2. Minutiae-based Representation Fingerprint

Most conventional fingerprint recognition algorithms are based on a minutiae-based representation, which has a rather low storage cost. A reliable minutiae

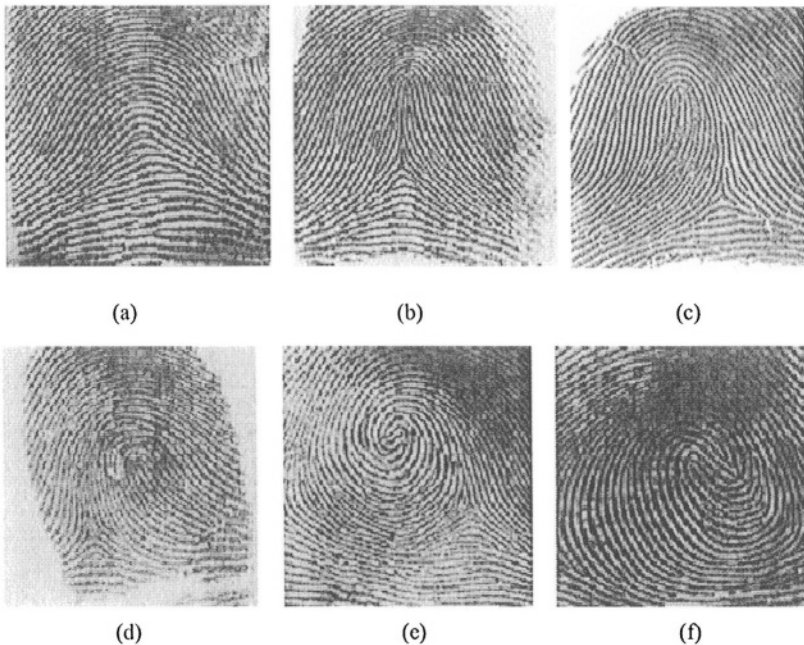


Figure 14.2. One fingerprint for each of six main classes: (a) arch; (b) tented arch; (c) left loop; (d) right loop; (e) whorl and (f) twin loop.

estimation, ridge extraction or enhancement, ridge thinning and minutiae extraction. Figure 14.3 provides a flowchart of conventional minutiae extraction and minutiae matching algorithms.

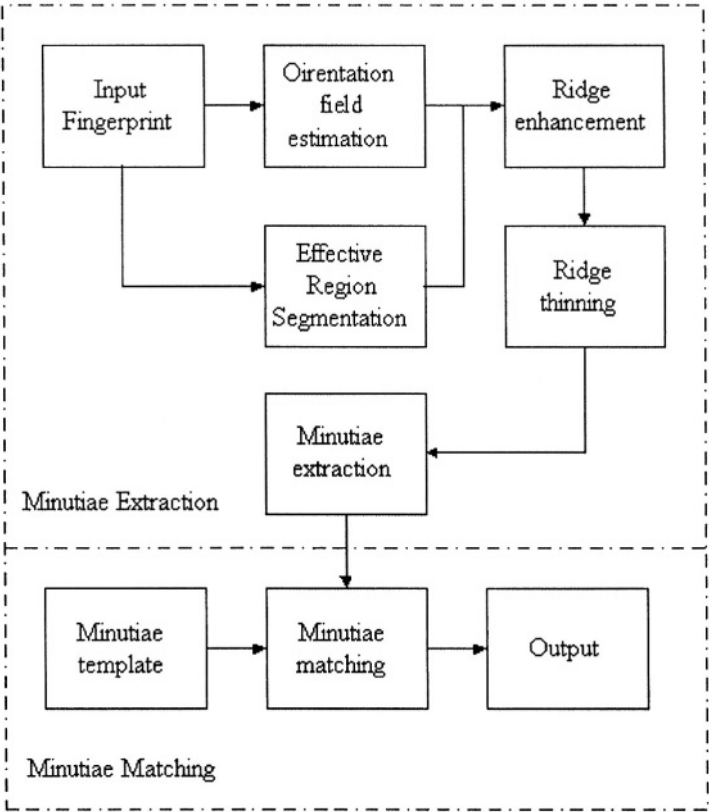


Figure 14.3. Flowchart of minutiae extraction and matching.

By quantifying the amount of information available in minutiae-based representation, a correspondence can be established between two fingerprints. Yet between the minutiae-based representations of two arbitrarily chosen fingerprints belonging to different fingers it is also possible to establish a false correspondence. For example, the probability that a fingerprint with 36 minutiae points will share 12 minutiae points with another arbitrarily chosen fingerprint with 36 minutiae points is 6.10×10^{-8} [3]. Fig 14.4 provides an example of such a false correspondence. Because of noise in the sensed fingerprint images, errors in

locating minutiae, and the fragility of the matching algorithms, the observed matching performance of the state-in-the-art fingerprint recognition systems is several orders of magnitude lower than their theoretical performance. Because minutiae-based representations use only a part of the discriminatory information present in fingerprints, it may be desirable for the purposes of automatic matching to explore additional complementary fingerprint representations. Taking Fig 14.4 as an example, the orientation fields of these two fingerprints are clearly different in some places, such as at the bottom of the fingerprints and the left-down part from the core. Including orientation information in the matching step would greatly reduce opportunities for this kind of false match.

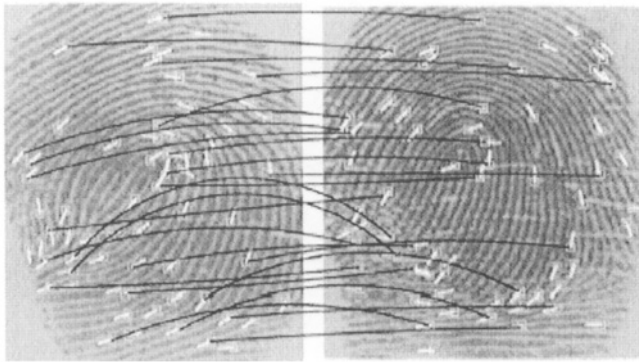


Figure 14.4. A false match between two different fingerprints: 64 minutiae are detected in the left image, 65 in the right image, with 25 “false” correspondences [3].

14.3. Modeling Orientation Fields

As a global feature, an orientation field describes one of the basic structures of a fingerprint and is thus quite important in the modelling and representation of the entire fingerprint. Orientation field variation is low frequency so it is robust with respect to various noises. It has been widely used for minutiae extraction and fingerprint classification. In this section, we focus on the modelling of the orientation field. Our purpose is to represent the orientation field in a complete and compact form so that it can be accurately reconstructed with several coefficients. This work is significant in three ways. (1) It can be used to improve the estimation of orientation field, especially when fingerprint quality is poor; therefore it will be of benefit in the extraction of minutiae for conventional fingerprint identification algorithms. (2) The coefficients of the orientation field model can be saved for use in the matching step. As a result, information on the orientation field can be utilized for fingerprint identification. By combining it with the minutiae

information, we can expect a much better identification performance. (3) We can synthesize the fingerprint by using information on the orientation field, minutiae and the density between the ridges. This makes it possible to establish a complete representation for the fingerprint by combining the orientation model with other information.

14.3.1 Zero-Pole Model

Sherlock and Monro [5] proposed a so-called zero-pole model for the orientation field based on singular points, which takes the core as zero and the delta as a pole in the complex plane. The influence of a core, z_c , is $\frac{1}{2}\arg(z-z_c)$ for the point, z , and that of a delta z_d , is $-\frac{1}{2}\arg(z-z_d)$. The orientation at z , is the sum of the influence of all cores and deltas, i.e.

$$\theta(z) = \frac{1}{2} [\sum_i \arg(z - z_c^i) - \sum_j \arg(z - z_d^j)] \quad (14.1)$$

where z_c^i and z_d^j are the i -th core and j -th delta. Figure 14.5 depicts the influence of a core and delta. They can roughly describe the structure near these singular points.

14.3.2 Piecewise Linear Approximation Model

Vizcaya and Gerhardt [6] had made an improvement using a piecewise linear approximation model around singular points to adjust the behaviour of the zero and pole. First, the neighbourhood of each singular point is uniformly divided into eight regions and the influence of the singular point is assumed to change linearly in each region. An optimization implemented by gradient-descend is then performed to get a piecewise linear function.

These two models, the zero-pole model and the piecewise linear model, cannot deal with fingerprints belonging to the plain arch class (i.e. without singular points). Furthermore, they do not take into account the distance from singular points, but as the influence of a singular point is the same as any point on the same central line, whether near or far from the singular point, serious errors arise in the modelling of those regions that are far from singular points. As a result, these two models cannot be used to accurately approximate a real fingerprint's orientation field.

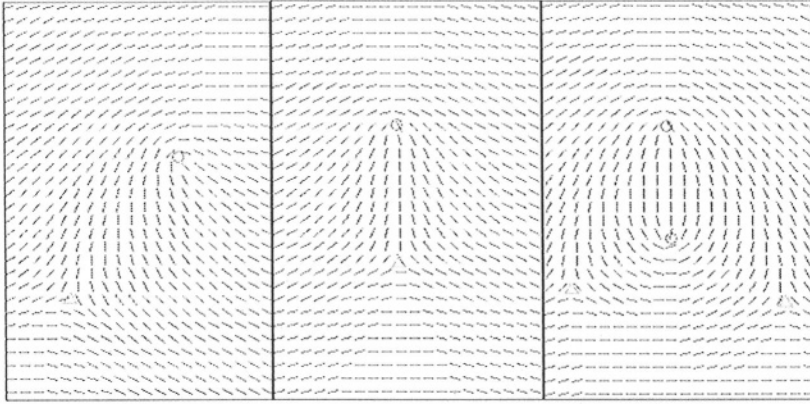


Figure 14.5. Illustration of zero-pole model.

14.3.3 Rational Complex Model [7]

Denote the image plane as a complex space, C . For any $z \in C$, the value of a fingerprint orientation, $\theta(z)$, is defined within $[0, \pi)$, so it can be regarded as half the argument of a complex number, i.e. $\theta(z) = \frac{1}{2} \arg U(z)$. As we know, the orientation pattern of a fingerprint is quite smooth and continuous except at the singular points (including cores and deltas), so a rational complex function may be utilized here to represent the function, $U(z)$, in which the known cores and deltas act as zeros of the numerator and the denominator, respectively. Thus, the model for the orientation field can be defined as

$$\phi(z) = \frac{1}{2} \arg \left[\frac{f(z)}{g(z)} \cdot \frac{P(z)}{Q(z)} \right], \quad (14.2)$$

where

$$P(z) = \prod_{i=1}^{s_0} (z - z_c^i), \quad Q(z) = \prod_{j=1}^{s_1} (z - z_d^j), \quad \{z_c^i\}_{1 \leq i \leq s_0} \text{ and } \{z_d^j\}_{1 \leq j \leq s_1} \text{ are the}$$

cores and deltas of the fingerprint in the known region. The zeros of $f(z)$ and $g(z)$ should be outside the known region. Actually, these zeros are always sparsely located on real fingerprints. All the zeros of $f(z)$, $g(z)$, $P(z)$ and $Q(z)$ define the nature of the model.

Obviously, the zero-pole model proposed in [5] can be regarded as a special example of the rational complex model, where $f(z)$ and $g(z)$ are both set to a constant, such as 1. It also should be noted that our model is suitable for all types of fingerprints, even for "plain arch" fingerprints, which do not have singular points.

How to compute the coefficients of the model? From the mathematical theorem of complex variables, we know that a rational function can be approximated using a polynomial function in a closed region. So, to simplify the computation of the rational complex model, the model can be written as

$$\phi(z) = \frac{1}{2} \arg[f(z) \cdot \frac{P(z)}{Q(z)}]. \quad (14.3)$$

We want to find a function, $f(z)$, to minimize the difference between $\{\phi(z)\}$ and the original orientation field, $\{\theta(z)\}$. Denote $\psi(z) = \frac{1}{2} \arg[\frac{P(z)}{Q(z)}]$ and $\omega(z) = \frac{1}{2} \arg[f(z)]$, and now what we need to do is to compute $f(z)$ by minimizing the difference between $\{\omega(z)\}$ and $\{\theta(z) - \psi(z)\}$.

It is unsuitable for us to directly compute this minimum. A solution to this problem is to map the orientation field to a continuous complex function by using

$$U(z) = \cos 2[\theta(z) - \psi(z)] + i \sin 2[\theta(z) - \psi(z)], \quad (14.4)$$

Then, instead of computing the minimal difference between $\{\omega(z)\}$ and $\{\theta(z) - \psi(z)\}$, we compute the function, $f(z)$, by minimizing $\sum_z |f(z) - U(z)|^2$. Since $\theta(z)$, $P(z)$ and $Q(z)$ (corresponding to the original orientation field, cores and deltas) are known when we deal with a fingerprint image, it is easy to solve this problem by using Least Square Error principle.

When we choose $f(z)$ from the set of polynomials of an order less than n , only $n+1$ parameters need to be computed and saved (many fewer than the model in [6]). In the experiments, n is usually set as 6. Due to the global approximation, the rational complex model has a robust performance against noise.

An experiment was carried out on more than 100 inked fingerprints and live-scanned fingerprints. These fingerprints were of different types: loop, whorl, twin loop, and plain arch without singular points. They also varied in different qualities. Three orientation models are evaluated on the database, zero-pole model,

piecewise linear model and rational complex model. All of them used the same algorithm for singular points extraction and orientation estimation algorithms. On all these fingerprint images, the performance of the rational complex model was quite satisfying. The average approximation error of the rational complex model was about 6 degrees, which is much better than using the other two models (respectively 14 degrees and 11 degrees). This shows that the rational complex model's more effectively than the previous models.

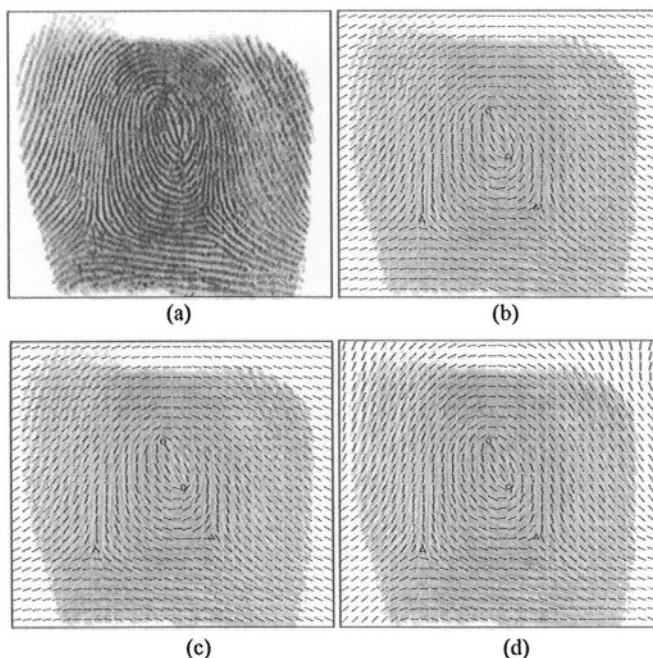


Figure 14.6. Comparative result of the orientation field constructed by using three models: (a) original fingerprint image; (b) zero-pole model; (c) piecewise linear model and (d) rational complex model.

Figure 14.6 provides an example for comparison, where (a) is the original fingerprint, and (b), (c), and (d) are the reconstructed orientation fields, respectively using the zero-pole, piecewise linear, and rational complex models. The reconstructed orientation field is shown as unit vectors upon the original fingerprint. As shown, the zero-pole model and piecewise linear model perform badly far from the singular points, easily observable in the top-left and the top-right part in (c) and (d). In contrast, the rational complex model describes the orientation of the whole fingerprint image precisely.

14.3.4 Combination Model [8]

Since the orientation of fingerprints is quite smooth and continuous except at singular points, a polynomial model can be applied to approximate the global orientation field. At each singular point, the local region is described using a point-charge model similar to the zero-pole model. Then, these two models are smoothly combined through a weight function.

Since the value of the orientation of a fingerprint is defined within $[0, \pi]$, it seems that this representation has an intrinsic discontinuity. (In fact, the orientation 0 is the same as the orientation π in a ridge pattern). As a result, we cannot model the orientation field directly. A solution to this problem is to map the orientation field to a continuous complex function. Define $\theta(x,y)$ and $U(x,y)$ as respectively the orientation field and the transformed function. The mapping can be defined as

$$U = R + iI = \cos 2\theta + i \sin 2\theta \quad (14.4)$$

where R and I denote respectively the real and the imaginary parts of the complex function, $U(x,y)$. Obviously, $R(x,y)$ and $I(x,y)$ are continuous with x, y in those regions. The above mapping is a one-to-one transformation and $\theta(x,y)$ can be easily reconstructed from the values of $R(x,y)$ and $I(x,y)$.

To globally represent $R(x,y)$ and $I(x,y)$, two bivariate polynomial models are established, which are denoted by PR, PI respectively. These two polynomials can be formulated as:

$$PR(x, y) = \begin{pmatrix} 1 & x & \cdots & x^n \end{pmatrix} \cdot P_1 \cdot \begin{pmatrix} 1 & y & \cdots & y^n \end{pmatrix}^T, \quad (14.5)$$

and

$$PI(x, y) = \begin{pmatrix} 1 & x & \cdots & x^n \end{pmatrix} \cdot P_2 \cdot \begin{pmatrix} 1 & y & \cdots & y^n \end{pmatrix}^T, \quad (14.6)$$

where n is the polynomials' order and the matrixes, $P_i \in \mathbb{R}^{n \times n}$, $\forall i = 1, 2$.

Near the singular points, the orientation is no longer smooth, so it is difficult to model with a polynomial function. A model named 'Point-Charge' (PC) is added at each singular point. Compared with the zero-pole model, Point-Charge uses different quantities of electricity to describe the neighbourhood of each singular point instead of the same influence at all singular points, while the influence of a certain singular point at the point, (x,y) , varies with the distance between the point and the singular point. The influence of a standard (vertical) core at the point, (x,y) , is defined as

$$PC_{Core} = H_1 + iH_2 = \begin{cases} \frac{y-y_0}{r}Q - i\frac{x-x_0}{r}Q, & r \leq R \\ 0, & r > R \end{cases}, \quad (14.7)$$

where (x_0, y_0) is this core's position. Q is the quantity of electricity, R denotes the radius of its effective region, and $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. The radius of a standard delta is:

$$PC_{Delta} = H_1 + iH_2 = \begin{cases} -\frac{y-y_0}{r}Q - i\frac{x-x_0}{r}Q, & r \leq R \\ 0, & r > R \end{cases}. \quad (14.8)$$

In a real fingerprint, the ridge pattern at the singular points may have a rotation angle compared with the standard one. If the rotation angle from standard position is ϕ ($\phi \in (-\pi, \pi]$), a transformation can be made as:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x-x_0 \\ y-y_0 \end{pmatrix}. \quad (14.9)$$

Then, the Point-Charge model can be modified by taking x' and y' instead of x and y , for cores in Eq. (14.7) and deltas in Eq. (14.8), respectively.

To combine the polynomial model (PR , PI) with Point-Charge smoothly, a weight function can be used. For Point-Charge, the weighting factor at the point (x, y) is defined as:

$$\alpha_{PC}^{(k)}(x, y) = 1 - \frac{r^{(k)}(x, y)}{R^{(k)}}, \quad (14.10)$$

where $(x_0^{(k)}, y_0^{(k)})$ is the coordinate of the k -th singular point, $R^{(k)}$ is the radius of the effective region. and $r^{(k)}(x, y)$ is set as $\min(\sqrt{(x-x_0^{(k)})^2 + (y-y_0^{(k)})^2}, R^{(k)})$. For the polynomial model, the weighting factor at the point, (x, y) , is:

$$\alpha_{PM}(x, y) = \max \left\{ 1 - \sum_{k=1}^K \alpha_{PC}^{(k)}, 0 \right\}, \quad (14.11)$$

where K is the number of singular points. The weight function guarantees that for each point, its orientation follows the polynomial model if it is far from the singular points and follows the Point-Charge if it is near one of the singular points.

Then, the combination model for the whole fingerprint's orientation field can be formulated as:

$$\begin{pmatrix} R(x,y) \\ I(x,y) \end{pmatrix} = \alpha_{PM} \cdot \begin{pmatrix} PR \\ PI \end{pmatrix} + \sum_{k=1}^K \alpha_{PC}^{(k)} \cdot \begin{pmatrix} H_1^{(k)} \\ H_2^{(k)} \end{pmatrix}, \quad (14.12)$$

with the constraint of

$$R^2(x,y) + I^2(x,y) = 1, \quad (14.13)$$

where PR and PI are respectively the real and the imaginary parts of the polynomial model, and $H_1^{(k)}$ and $H_2^{(k)}$ are respectively the real and the imaginary parts of the Point-Charge model for the k -th singular point. Obviously, the combination model is continuous with x and y . The coefficient matrices of the two polynomials, PR and PI , and the electrical qualities, $\{Q_1, Q_2, \dots, Q_K\}$, of the singular points will define the combination model.

Obviously, the combination model can be regarded as a generalized method of the other three models introduced above, so it can represent the orientation field more accurately. It does, however, have more parameters than the zero-pole and rational complex models, and this will produce some limitations when it is utilized in synthetic fingerprint generation.

To compute its parameters, the coarse orientation field and singular points need to be estimated from the original fingerprint image. After that, two bivariate polynomials can be computed using the Weighted Least Square (WLS) algorithm. The coefficients of the polynomial are obtained by minimizing the weighted square error between the polynomial and the values of $R(x,y)$ and $I(x,y)$ computed from the real fingerprint. As pointed out above, the reliability, $W(x,y)$, can indicate how well the orientation fits the real ridge. The higher the reliability $W(x,y)$, the more influence the point should have. Then $W(x,y)$ can be used as the weighting factor at the point (x,y) .

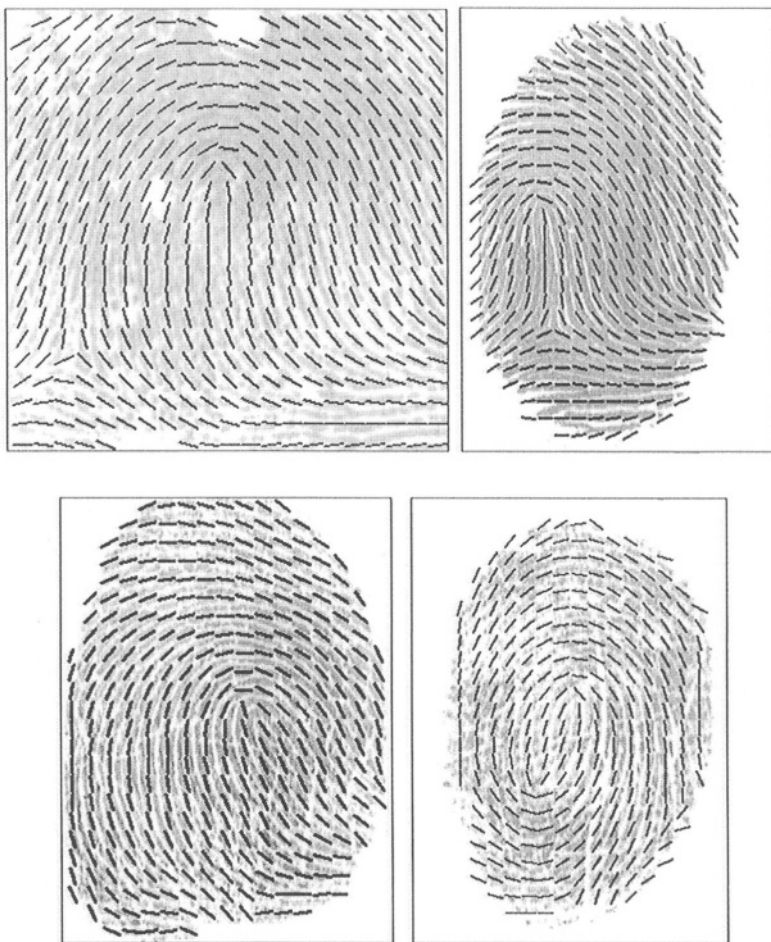


Figure 14.7. Some examples of approximation results for the combination model.

After the computation of polynomials, the coefficients of the Point-Charge Model at singular points can be obtained in two steps. First, two parameters are estimated for each singular point: the rotation angle, ϕ , and the effective radius, R (which can also be chosen in advance). Second, charges of singular points are estimated by optimization.

As we know, a higher order polynomial can provide a better approximation, but at the same time it will result in a much higher cost of storage and computation. Moreover, a high order polynomial will be ill-behaved in numerical approximation. As to a lower order polynomial, however, it will yield lower approximation accuracy in those regions with high curvature. As a trade-off, 4-order ($n=4$)

polynomials can be chosen for the global approximation. The experimental results showed that they performed well enough for almost all real fingerprints, while the cost for storage and computation remained low. In Figure 14.7, some of the results are presented, in which the reconstructed orientation fields are shown as unit vectors upon the original fingerprint. As shown, the results are rather accurate and robust for these fingerprints, although there is a lot of noise in these images.

The combination model has many more parameters than the zero-pole and rational complex models, making it, in comparison with those two models, much more difficult to use in synthesizing fingerprint images

14.4. Generation of Synthetic Fingerprint Images

The main topic of this section is the generation of synthetic fingerprint images. Such images can not only be used to create, at zero cost, large databases of fingerprints, thus allowing recognition algorithms to be simply tested and optimized, they would help us to better understand the rules that underpin the biological process involved in the genesis of fingerprints. The effective modelling of fingerprint patterns could also contribute to the development of very useful tools for the inverse task, i.e. fingerprint feature extraction.

The generation method sequentially performs the following steps [9]: (1) orientation field generation, (2) density map generation, (3) ridge pattern generation and (4) noising and rendering.

Orientation field generation: There are three ways to generate the orientation field for synthetic fingerprints. 1) Using the zero-pole model in [6], a consistent orientation field can be calculated from the predefined position of the cores and deltas alone, (see Figure 14.5) but the generated orientation is less accurate than real fingerprints. 2) The rational complex model in [7] produces a more lifelike result. From Eq. (14.2), we know that the rational complex model is determined by the zeros of $f(z)$, $g(z)$, $P(z)$ and $Q(z)$, in which the zeros of $P(z)$ are cores and those of $Q(z)$ are deltas. The position of cores and deltas can be predefined according to the fingerprint's class, then, several points are randomly and sparsely selected from outside the print region as the zeros of $f(z)$ and $g(z)$. 3) We can first choose several real fingerprints and compute the parameters of the rational complex model by minimizing the approximation error between the model and the fingerprint image' orientation [7]. Then these parameters can be changed a little randomly to produce different orientation fields.

Density map generation [9]: This step creates a density map on the basis of some heuristic criteria inferred from the visual inspection of several real fingerprints. The visual inspection of several fingerprint images, leads us to immediately discard the possibility of generating the density map in a completely random way. In fact, we noted that usually in the region above the northernmost

core and in the region below the southernmost delta the ridge-line density is lower than in the rest of the fingerprint. So, the density-map can be generated as follows: 1) randomly select a feasible overall background density; 2) slightly increase the density in the above-described regions according to the singularity locations; 3) randomly perturb the density map and performs a local smoothing.

Ridge pattern generation [9]: In this step, the ridgeline pattern and the minutiae are created through a space-variant linear filtering; the output is a very clear near-binary fingerprint image. Given an orientation field and a density map as input, a deterministic generation of a ridgeline pattern, including consistent minutiae is not an easy task. One could try a priori to fix the number, type and location of the minutiae, and by means of an explicit model, generate the gray-scale fingerprint image starting from the minutiae neighbourhoods and expanding to connect different regions until the whole image is covered. Such a constructive approach requires several complex rules and tricks to be implemented in order to deal with the complexity of fingerprint ridgeline patterns. A more "elegant" approach could be based on the use of a syntactic approach that generates fingerprints according to some starting symbols and a set of production rules. The method here proposed is very simple, but at the same time surprisingly powerful: by iteratively enhancing an initial image (containing one or more isolated spikes) through Gabor-like filters adjusted according to the local orientation and density, a consistent and very realistic ridge-line pattern "magically" appears; in particular, fingerprint minutiae of different types (terminations, bifurcations, islands, dots, etc.) are automatically generated at random positions. Formally, the filter is obtained as the product of a Gaussian by a cosine plane wave. A correction term is included to make the filter DC free:

$$f(\mathbf{v}) = \frac{1}{\sigma^2} e^{-\frac{\|\mathbf{v}\|^2}{2\sigma^2}} \left[\cos(\mathbf{k} \cdot \mathbf{v}) - e^{-\frac{\sigma\|\mathbf{k}\|^2}{2}} \right], \quad (14.14)$$

where σ is the variance of the Gaussian and $\mathbf{k}$ is the wave vector of the plane wave. The parameters σ and $\mathbf{k}$ are adjusted using local ridge orientation and density. Let $\mathbf{z}$ be appoint of the image where the filters have to be applied, then the vector $\mathbf{k} = [k_x, k_y]^T$ is determined by the solution of the two equations:

$$D(\mathbf{z}) = (k_x^2 + k_y^2)^{1/2} \quad \text{and} \quad \tan(O(\mathbf{z})) = -\frac{k_x}{k_y}. \quad (14.15)$$

The parameter σ , which determines the bandwidth of the filter, is adjusted in the time domain according to $D(\mathbf{z})$ so that the filter does not contain more than three effective peaks. The filter is then clipped to get a FIR filter. The filter should

be designed with the constraint that the maximum possible response is larger than 1. When such a filter is applied repeatedly, the dynamic range of the output increases and becomes numerically unstable, but the generation algorithm exploits this fact. When the output values are clipped to fit into a constant range, it is possible to obtain a near-binary image. The above filter equation satisfies this requirement without any normalization.

In Figure 14.8, an example of the iterative ridgeline generation process is shown.

Noising and rendering [9]: In this step, some specific noise is added and a realistic gray-scale representation of the fingerprint is produced. During fingerprint acquisition several factors contribute to the deterioration of the original signal, thus producing a gray-scale noisy image: 1) irregularity of the ridge and differences in their contact with the sensor surface; 2) the presence of small pores within the ridges; 3) the presence of very-small-prominence ridges; 4) gaps and cluttering noise due to non-uniform pressure of the finger against the sensor or due to excessively wet or dry fingers. So, the noising and rendering approach sequentially performs the following steps: 1) isolate the valley white pixels into a separate layer by copying the pixels brighter than a fixed threshold to a temporary image; 2) add noise in the form of small white blobs of variable size and shape; 3) smooth the image; 4) superimpose the valley layer to the image obtained. In the above steps, steps 1 and 4 are necessary to avoid an excessive overall image smoothing.

14.5. Complete Representation of Fingerprints

Synthesizing a fingerprint using an orientation field and a density map means the orientation field and density map will constitute a complete fingerprint representation. The minutiae primarily originate from the ridgeline disparity produced by local convergence/divergence of the orientation field and by density changes. As stated above, a complete representation of fingerprints will help us to choose an appropriate feature set for the matching. From this it would seem that these two features, an orientation field and a density map, are all enough for the purpose of fingerprint matching. Unfortunately this is not so.

We tested this with two experiments. In one we constructed a fingerprint from the original fingerprint image using the synthesis method. In the other we compared two synthetic fingerprints using different starting points.

In the first experiment, the orientation field and density map should be computed from the original fingerprint image. Many algorithms have been proposed for the computation of the orientation field of a real fingerprint image. Of these, we prefer the algorithm proposed in [10]. To compute the density map, we first take some steps similar to the ridge detection method used in [4], then use

a median filter to remove the noise. After that, it is a simple matter to choose a threshold to segment the ridges adaptively. Along the direction normal to the local ridge orientation, the width of the ridge at this position can be measured. After all pixels are processed, the density map of the fingerprint image has been obtained. After obtaining the orientation field and density map, the method described in last section is taken to produce a new image. Unfortunately, the experimental results are not satisfactory and the reconstructed image is not similar to the original. Another experiment is conducted using different starting points and comparing the synthetic fingerprints. The result shows that starting from different points will result in evident changes in the final synthetic image. One example is provided in Figure 14.9.

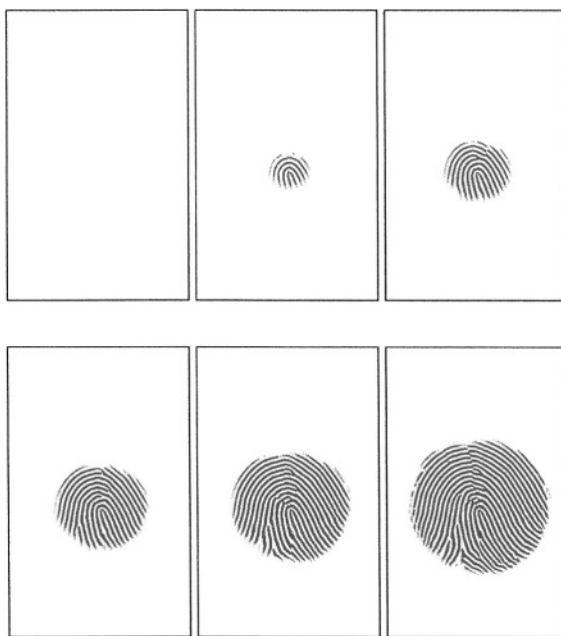


Figure 14.8. An example to illustrate the image-generation process.

These experiments are not conclusive but they do raise some questions and issues: (1) The proposed synthesis algorithm is an iterative method but is the iterative process convergent? If convergent, is the converged image the same as the original noise-affected fingerprint? (2) From the point of view of numerical analysis, the computational error will greatly influence the final result, so the main cause of the experiment's failure may be inaccurate computation of the orientation field and density map. It also implies that a representation method that uses only an orientation field and a density map is not adequate for recognition tasks. One

simple way to reduce the computational errors is to use more constraints, i.e., more features in our task, such as minutiae points.

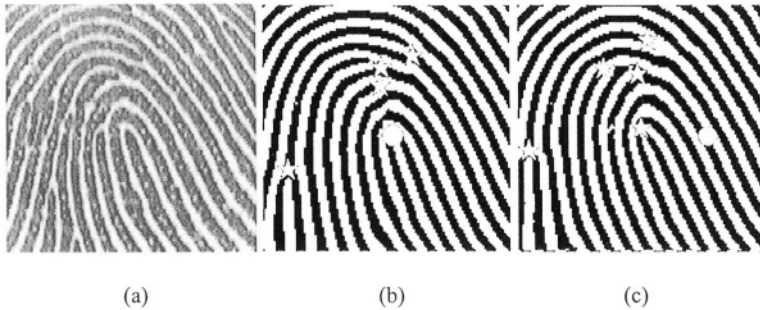


Figure 14.9. An example of reconstruction: (a) a real fingerprint image; (b) and (c) are two reconstructed images, in which the circles denote the starting points and the star symbols denote the minutiae points. Twenty-five iterations were carried out.

14.6. Summary

Fingerprint recognition applications seek a complete but compact representation of fingerprints. In this chapter, we introduced issues related to the graphical representation of fingerprints and some approaches to these issues. Since orientation field is an important feature in the description of the global appearance of fingerprints, we have focused on orientation field models and have described methods based on them for generating synthetic fingerprint images. Finally, we discussed the reconstruction of a noise-less fingerprint image from an original fingerprint image. We summarise our conclusions as follows:

- Conventional fingerprint recognition algorithms rely on minutiae-based representations of fingerprints. As a minutiae-based representation uses only a part of the discriminatory information present in fingerprints, further exploration of additional complementary representations of fingerprints for automatic matching is needed.
- An orientation field can be well represented by either a rational complex model or a combination model. In the rational complex model, the orientation field of fingerprints is expressed as the argument of a rational complex function. In the combination model, it is represented by bivariate polynomials globally and is rectified locally by several point-charge models. As a comparison, the rational complex model is more compact while the combination model approximates more effectively. Both of these models

make it feasible to utilize the orientation information into the matching stage.

- Using a feature set consisting of an orientation field and a density map, we can synthesize a new fingerprint, which shows that this feature set constitutes a complete fingerprint representation. However, in order to obtain a robust representation for real applications, minutiae information is still very helpful, so for effective fingerprint matching it may be desirable to use features of minutiae, orientation fields, and density maps

Acknowledgments

This work is supported by National Science Foundation of China (Grant No. 60205002) and National Hi-Tech Development Program of China. Also, the authors would like to express their gratitude to Prof. Zhaoqi Bian and Prof. Gang Rong for their contributions on technical discussions.

References

- [1] A.K. Jain, R. Bolle, S. Pankanti (Eds.). *BIOMETRICS: Personal Identification in Networked Society*. Kluwer, New York, 1999.
- [2] D. Zhang. *Automated Biometrics: Technologies and Systems*. Kluwer Academic Publisher, USA, 2000.
- [3] S. Pankanti, S. Prabhakar, and A. K. Jain. "On the individuality of fingerprints", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1010-1025, 2002.
- [4] A. Jain and L. Hong. "On-line fingerprint verification", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.19, no. 4, pp. 302-314, 1997.
- [5] B. Sherlock and D. Monro. "A model for interpreting fingerprint topology", *Pattern Recognition*, vol. 26, no. 7, pp.1047-1055, 1993.
- [6] P. Vizcaya and L. Gerhardt. "A nonlinear orientation model for global description of fingerprints", *Pattern Recognition*, vol. 29, no.7, pp. 1221-1231, 1996.
- [7] J. Zhou and J. Gu. "Modeling orientation field of fingerprints with rational complex functions". *Pattern Recognition*, 2003, in press.
- [8] J. Gu, J. Zhou, and D. Zhang. "A combination model for orientation field of fingerprints", *Pattern Recognition*, 2003, in press.
- [9] R. Cappelli, A. Erol, D. Maio and D. Maltoni. "Synthetic fingerprint-image generation". *Proc. of ICPR*, pp. 471-474, 2000.
- [10] J. Zhou and J. Gu. "A model-based method for the computation of fingerprints' orientation field", in revised form, submitted to *IEEE Trans. on Image Processing*, June, 2003.

This page intentionally left blank

Chapter 15

IMAGE BASED MODELING AND ANALYSIS OF TEXTILE MATERIALS

Jinlian Hu and Binjie Xin

*Institute of Textile & Clothing, The Hong Kong Polytechnic University,
Kowloon, Hong Kong
tchujl@polyu.edu.hk*

Abstract In this chapter, we describe the modeling and analysis of fabric appearance using image techniques and state-of-the-art of objective evaluation of textile surfaces for quality control in textile industry. The chapter contains the analysis of three major textile surface appearance attributes, namely, pilling, wrinkling and polar fleece fabric appearance and the modeling methods for different textile materials with different surface features including template matching for pilling modeling, morphological fractal for polar fleece fabrics and photometric stereo for 3D wrinkling,

Keywords: Pilling, wrinkling, polar fleece, fabric, objective evaluation, subjective evaluation, Gaussian function, template matching, grade rating, fractal, morphological fractal, modeling, photometric stereo

15.1. Introduction

Appearance is one of the most important attributes for the quality evaluation of fabrics or garments. A fabric product's acceptability might very much be influenced only due to the unpleasant appearance factors like pilling, wrinkle and hairiness. Currently, fabric appearance testing mainly relies on manual and subjective assessment and thus the accuracy and reliability is rather doubtful. The only solution to this lies in the invention of an objective evaluation system for textile appearance and this becomes the subject of this chapter. The system proposed is based on image techniques.

Also presented in this chapter is the analysis of three major textile surface appearance attributes (pilling, wrinkling and polar fleece fabric), and the modeling

methods for each, viz. template matching for pilling, morphological fractal for polar fleece fabrics and photometric stereo for 3D wrinkling,

The study of fabric appearance started from 1950s. Without exception, the pioneers usually captured grey images using CCD camera and processed with computer and image technologies. Unfortunately, this approach met great difficulty when assessing the patterned fabrics. Hence improvement on the fabric image capture is required to reduce the disturbance of fabric patterns. Laser scanner and structural lighting were thus adopted since they can enable the extraction of three-dimensional profile of fabric surface but these two approaches still suffer the limitation like high cost and the low accuracy of structural lighting.

15.2. Modeling of Fabric Pilling

15.2.1 Introduction

Pilling can spoil considerably the appearance of a fabric and even leads to the rejection of a fabric product. The generation of pilling usually begins with a migration of fibers to the external part of yarns so that fuzz emerges in the fabric surface. Due to friction, fuzz gets entangled forming pills, which remain suspended from fabric, by long fibers. Therefore, the formation of a pill can be divided into four stages: fuzz formation, entanglement, growth, and wear-off [28].

Traditional pilling evaluation is usually very subjective, which is based on rating the processed fabric samples according to the standard pilling photos. Apparently, the result obtained from this method is rather subjective and thus lacks inconsistency and accuracy.

Image analysis technique was thus introduced and presented prominent improvement for the characterization and inspection of textile materials [5,8,9,16,17,24,27,29]. Laser-scanning technique successfully conquers the difficulty in identifying the pills on a patterned fabric by acquiring 3D fabric image. However, the scanning process of this technique slows down the process of data acquisition. Video camera together with effective algorithm to identify pills was thus introduced. Konda et al. [1] used video camera and an almost tangential illumination to capture the samples and Hsi et al. [9] found that diffuse light source is much more suitable for pilling identification than collimated lighting, so the influence of fabric texture can be eliminated. Wager [16] successfully used a derived algorithm consisting pilling area and major axis breadth/major axis length ratio to identify pills. Hector et al. [8]' method of identifying pilling was based on image analysis. And Xu [5] first introduced the concept of "template" into pilling identification and demonstrated a big success for the solid color fabric.

"Template Matching" can simulate well the visual perception of human beings and thus was adopted in our work. The success of this method lies in the choice of an effective and suitable template. The work reported here involves the

development of a practical algorithm named "Pill Template Training", which can be used to generate a suitable template b . Also described is the method of applying "Pill Template Matching" to identify pills from common fabric texture and building the statistic model of grade rating system.

15.2.2 Theory

Figure 15.1 illustrates the procedures followed in image analysis. The whole flow takes about 10s.

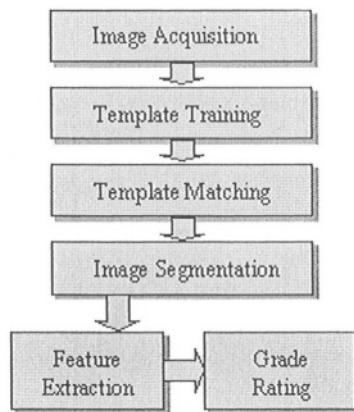


Figure 15.1. Flow diagram of pilling.

Pill Template Training. The difficulty of pill detecting lies in the great variety of fabric structure. However, considering the fact that pills are spaced relatively far apart and elevated from fabric surface, this difficulty might be well overcome, viz. pills are much brighter than the ground of a fabric when illuminating a fabric obliquely. In image analysis, pills can be assumed as an elliptical or circular object containing a centered white circle surrounded by black pixels. A two-dimensional Gaussian function thus is chosen as the template for pill detection.

Firstly, crop a typical pill image and then fit each pill image using two-dimensional Gaussian function. This is followed by "Pill template training", namely using "actual pill image" to construct the pill template. The template constructed this way apparently is very adaptive to fabric textures since it was synthesized from actual fabric image.

Two-Dimensional Gaussian Fit Theory. One can measure a circular or elliptical object such as a pill in an image by fitting a two-dimensional Gaussian surface through the image. The equation for the two-dimensional Gaussian is

$$z_i = Ae^{\left[-\frac{(x_i - x_0)^2}{2\sigma_x^2} - \frac{(y_i - y_0)^2}{2\sigma_y^2} \right]} \quad (15.1)$$

where A is the amplitude, (x_0, y_0) the position, and σ_x and σ_y the standard deviations in the two directions. The equation can also be changed into

$$z_i \ln(z_i) = \left[\ln(A) - \frac{x_0^2}{2\sigma_x^2} - \frac{y_0^2}{2\sigma_y^2} \right] z_i \quad (15.2)$$

$$+ \frac{x_0^2}{\sigma_x^2} [x_i z_i] + \frac{y_0^2}{\sigma_y^2} [y_i z_i] + \frac{-1}{2\sigma_x^2} [x_i^2 z_i] + \frac{-1}{2\sigma_y^2} [y_i^2 z_i]$$

whose matrix form is:

$$Q = CB \quad (15.3)$$

where Q is an N -by- 1 vector with elements

$$q_i = z_i \ln(z_i) \quad (15.4)$$

C is a five-element vector composed entirely of Gaussian parameters

$$C^T = \left[\ln(A) - \frac{x_0^2}{2\sigma_x^2} - \frac{y_0^2}{2\sigma_y^2} \quad \frac{x_0^2}{\sigma_x^2} \quad \frac{y_0^2}{\sigma_y^2} \quad \frac{-1}{2\sigma_x^2} \quad \frac{-1}{2\sigma_y^2} \right] \quad (15.5)$$

and B is an N -by-5 matrix with i th row

$$[b_i] = [z_i \quad z_i x_i \quad z_i y_i \quad z_i x_i^2 \quad z_i y_i^2] \quad (15.6)$$

The C matrix can thus be computed by Q and B , and we can recover the Gaussian parameters from it by

$$\begin{aligned}\sigma_x^2 &= \frac{-1}{2c_4} & \sigma_y^2 &= \frac{-1}{2c_5} \\ x_0 &= c_2\sigma_x^2 & y_0 &= c_3\sigma_y^2\end{aligned}\quad (15.7)$$

and

$$A = e^{\left[c_1 + \frac{x_0}{2\sigma_x^2} + \frac{y_0}{2\sigma_y^2} \right]} \quad (15.8)$$

Only a five-by-five matrix must be inverted, regardless of N, the number of points used in the fit.

Template Training. σ_x and σ_y are the most important parameters to generate pill template. If $\sigma_x = \sigma_y$, the shape of pill template will be circular, otherwise, the shape will be elliptical. We also find that the values of σ_x and σ_y have a negative influence on the contrast of pill template. Figure 15.2 is an actual pill images cropped from fabric images while Figure 15.3 shows the fitted Gaussian templates determined from the raw image.

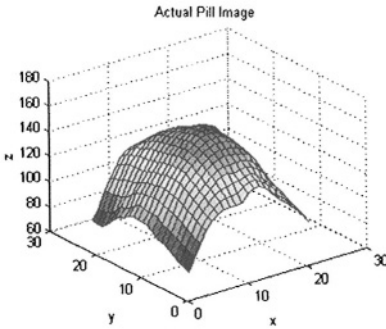


Figure 15.2. Actual pill image.

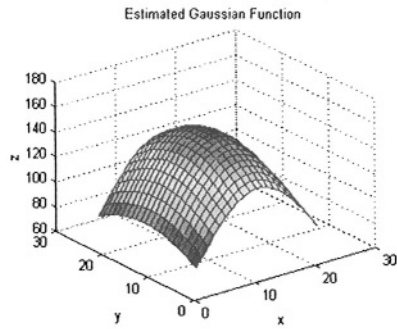


Figure 15.3. Fitted gaussian template.

Template Matching. Template matching is the process of moving the template over the entire image and calculating the similarity between the template and the covered window on the image. It is implemented through two-dimensional convolution, in which the value of an output pixel is computed by multiplying elements of two matrices and the summation of the results. One of these matrices represents the image itself, while the other matrix is the template.

If the original image and the template are denoted as $f(x, y)$ and $t(x, y)$, the enhanced image after convolution is denoted as $g(x, y)$, the convolution may be abbreviated by:

$$g(x, y) = f(x, y) \circ t(x, y) \quad (15.9)$$

where $\circ$ is used to indicate the convolution of two functions.
And discrete convolution is give by

$$g(i, j) = \sum_{m=1}^M \sum_{n=1}^N t(m, n) \cdot f(i - m, j - n) \quad (15.10)$$

where $M(N)$ is the size of the template.

The size of the template is mainly governed by the average size of texture elements, so FFT technique is adopted to calculate the frequencies of wale and course of plain knit fabric [28]. Ideal template size according experience is 1.5 loops, smaller or larger size might blur the pill image.

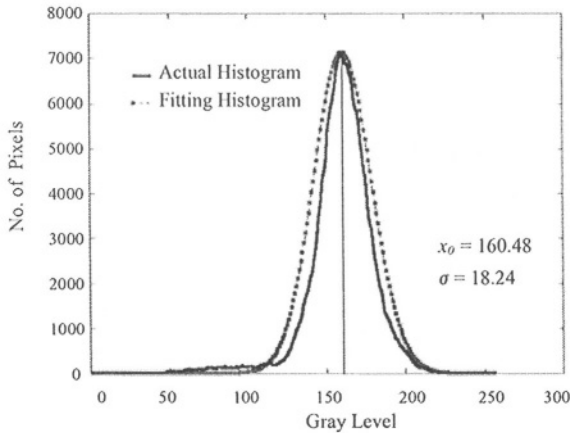


Figure15.4. Histogram of the filtered image and its fitting curve.

15.2.3 Image Segmentation

Template matching enables pill evaluation by generating an enhanced fabric image, in which a bright area indicates a pill. And image segmentation is the way to produce a binary image highlighting the pills. The success of image segmentation is threshold.

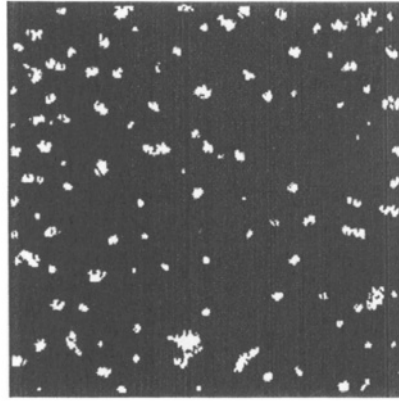


Figure 15.5. The binary image.

Figure 15.4 illustrates the histogram of the image generated from template matching, in the center of which a single peak can be observed. Model the histogram curve with a 1-D Gaussian distribution (dashed line), x_0 is the position of peak center and σ is the standard deviation of the Gaussian function, the threshold t can be worked out through equation:

$$t = x_0 + \lambda \cdot \sigma \quad (15.11)$$

where λ is a coefficient and when $\lambda=3$, Figure 15.5 shows the obtained result.

15.2.4 Feature Extraction

In the subjective evaluation, judges tend to rate the pilling appearance of a fabric by comparing pill properties such as number, area (size), contrast and density. All these properties can be measured in the binary fabric images objectively using image analysis techniques.

Number. Pill number n can be measured by counting the number of white objects in the binary image. Note that very small objects with area less than 4 pixels should be considered as noise objects and thus need to be removed.

Area (Size). Both d_i , the equivalent diameter in pixels and s_i , the area of one pill, can be used to describe the size of a pill, while s is the mean area of all pills and S denotes the total area of all pills.

Contrast. Pill contrast is a measure of how much the gray level of a pill differs from the gray level of the base fabric in fabric image. It is calculated by the following expression:

$$C_i = \frac{g_{p-i}}{g_b} \quad (15.12)$$

where g_{p-i} is the mean gray level of pill i , g_b is the mean gray level of background. $\bar{C}$ is the mean value of C_i .

Density. Pill density is another important factor influencing pilling appearance; a reasonable estimator of pill density can be referred to Ref. [2, 3].

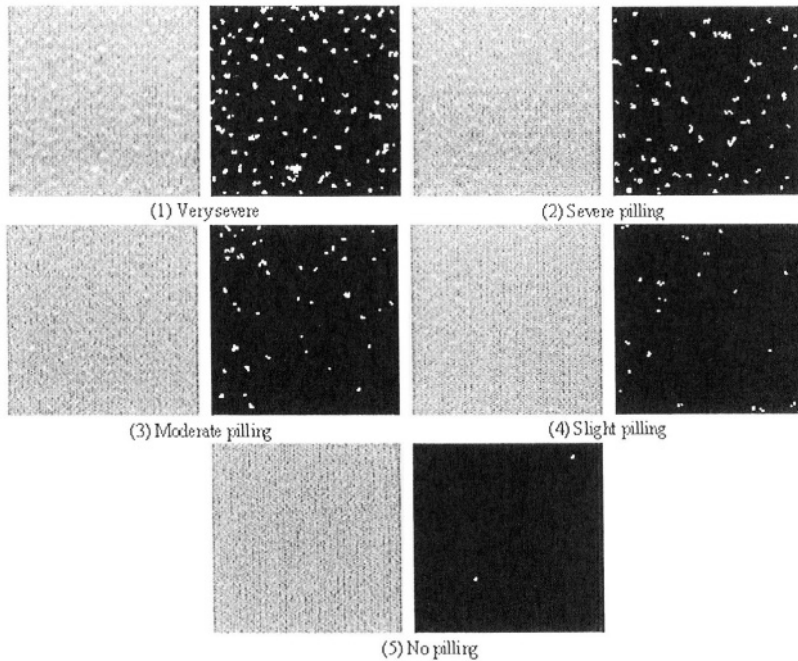


Figure 15.6. Standard Pilling Images.

15.2.5 Grade Rating

Figure 15.6 shows a comparison of five standard fabric images with their binary images through image segmentation. It is found that pill number n , mean area s ,

total area S and density D will decrease with the drop in subjective rating, while no apparent correlation could be found between mean contrast $\bar{C}$ and subjective rating. In addition, linear regression analysis implies that total area S is the most important and thus chosen to build the rating formula. Also revealed is a good correlation between the objective grading and subjective rating which confirms the proposed method is quite a success. It is also found that judges are insensitive to the mean area of pills but more sensitive to pill number n , total area S and density D so n , S and D are more important for subjective evaluation.

15.3. Modeling of Polar Fleece Fabric Appearance

15.3.1 Introduction

Polar fleece designates a knitted fabric with very high specific volume. Bulkiness is the main feature of fabrics in this class, but the fibers on the surface of the polar fleece tend to bunch up and form regular beards on the fabric surface after wearing or washing. Evaluating wearing appearance of polar fleece fabrics objectively, quickly and reliably has become an urgent task for both the purchaser and the supplier to achieve quality agreement.

Traditionally, evaluating fabric appearance still relies on subjective evaluation and thus lacks inconsistency, accuracy and reliability. Image analysis technique is thus adopted to conquer the problems.

A new fractal analysis method named Extended Morphological Fractal Analysis is a good solution to evaluate polar fleece fabric appearance while Multiscale Fractal Vector enable the characterization of various fabric textures by collecting all the fractal dimensions at different scales.

15.3.2 Theory

Fractal Model. Fractal geometry [7] characterizes the ability of an n -dimensional object to fill the $n+1$ dimensional space where the relationship of a measure M with the topological dimension n , and the scale ϵ is expressed as

$$M(\epsilon) \propto \frac{1}{\epsilon^r}, 0 \leq r \leq 1 \quad (15.13)$$

where the quantity $r+n$ (denoted by D) is called the fractal dimension or Hausdorff-Besicovich dimension and characterizes the degree of erratic behavior. For a real object, the measure M is independent of the scale ϵ and hence $n=D$. Thus, a fractal object can be defined as a set for which the fractal dimension is greater than the topological dimension.

Morphological Fractals. Digitized fabric image is represented as a surface whose height represents the gray level at each point. The surface area at different scales is estimated using a series of dilations and erosions of this surface.

Mathematical morphology developed by Matheron and Serra [15] extracts the impact of a particular shape on image via the concept of structuring element (SE). The SE encodes the primitive shape information. In a discrete approach, the shape is described as a set of vectors with respect to a particular point, the center. During morphological transformation, the center scans the whole image and shape matching information is used to define the transformation. The transformed image is thus a function of the SE distribution in the original image. In particular, dilation of a set X with a SE Y is given by the expression

$$X \oplus Y = \{x : Y^x \cap X \neq \emptyset\} \quad (15.14)$$

and erosion of a set X with an SE Y can be expressed as

$$X \ominus Y = \{x : Y^x \subset X\} \quad (15.15)$$

where Y^x indicates the translation of set Y with x . Hence the surface area of a compact set X with respect to a compact convex SE Y is given by:

$$S(X, Y) = \lim_{\varepsilon \rightarrow 0} \frac{V(\partial X \oplus \varepsilon Y - \partial X \ominus \varepsilon Y)}{2\varepsilon} \quad (15.16)$$

Where ∂X is the boundary of set X , $\oplus$ denotes the dilation of the boundary of set X by the SE Y scaled by a factor ε , $\ominus$ denotes the erosion of the boundary of set X by the SE Y scaled by a factor ρ . $V(X)$ gives the volume of set X .

It thus can be seen that dilating by ρY hides all structures smaller than ρY and therefore is equivalent to looking at the surface at scale ρ .

If the object is regular, the surface area will not change with ε . For a fractal object, $S(\partial X, Y, \varepsilon)$ is increasing exponentially with decreasing ε . By taking the logarithm, we got

$$\log(S(\partial X, Y, \varepsilon)) = \log(K) - r \log(\varepsilon) \quad (15.17)$$

$$D = 2 + r \quad (15.18)$$

where K is the proportionality constant. The value of D can be estimated by plotting $\log(S(\partial X, Y, \varepsilon_i))$ vs. $\log(\varepsilon_i)$ for a given set of scale factors ρ_i , $i = 1, 2, \dots, N$ and calculating the gradient of the line that fits the plot. Fractal analysis

is thus adopted to evaluate polar fleece fabric appearance since surface roughness and particle size are the most important features for quality evaluation.

Extended Morphological Fractal Analysis. Choose the rhombus element Y as the basic structuring element to perform a series of dilation and erosion operation on the fabric images. Figure 15.7 representations the structuring elements whose scale range from $\epsilon=1$ to $\epsilon=7$, and the area of these structuring elements is 5,13,25,41,61,85,113 pixels.

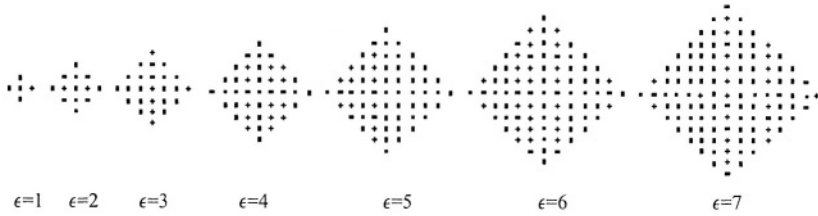


Figure 15.7. Structuring Elements at different scales.

The series of dilation or erosion of X by ϵY required for the above computation can be further reduced to dilation or erosion by the unit element Y by observing that if $X_Y^\epsilon = X \oplus \epsilon Y$ or $X_Y^\epsilon = X \ominus \epsilon Y$, then

$$X_Y^{\epsilon+1} = X \oplus (\epsilon + 1)Y = (X \oplus \epsilon Y) \oplus Y = X_Y^\epsilon \oplus Y \quad (15.19)$$

$$\text{or, } X_Y^{\epsilon+1} = X \ominus (\epsilon + 1)Y = (X \ominus \epsilon Y) \ominus Y = X_Y^\epsilon \ominus Y \quad (15.20)$$

The surface area of a set X at scale ϵ can be worked out through.

$$S(\partial X, Y, \epsilon) = \frac{V(\partial X \oplus \epsilon Y - \partial X \ominus \epsilon Y)}{2\epsilon} \quad (15.21)$$

The surface area $S(X, Y, \epsilon)$ can be iteratively calculated as follows: Define the image X as the set of triplets $\{f(x, y); x = 1, 2, \dots, N; y = 1, 2, \dots, M\}$ and the structuring element be given as a set of triplets $\{x_i, y_i, z_i, i = 1, 2, \dots, P\}$.

The ε th dilate $f_\varepsilon^u(x, y)$ is calculated as:

$$f_\varepsilon^u(x, y) = \max\{f_{\varepsilon-1}^u(x + x_i, y + y_i) \cdot z_i, i = 1, 2, \dots, P\} \quad (15.22)$$

The ε th erosion $f_\varepsilon^l(x, y)$ is calculated as:

$$f_\varepsilon^l(x, y) = \min\{f_{\varepsilon-1}^l(x + x_i, y + y_i) \cdot z_i, i = 1, 2, \dots, P\} \quad (15.23)$$

Figure 15.8 shows a comparison of the images of five standard polar fleece fabrics after erosion and dilation operations.

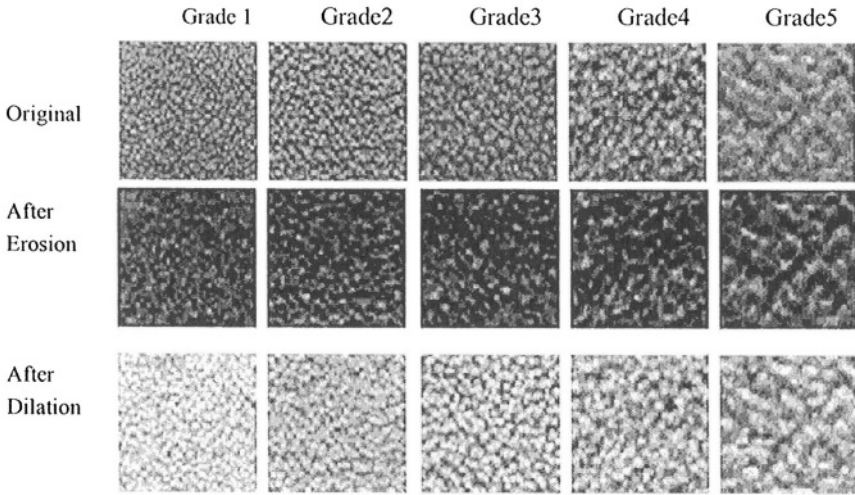


Figure 15.8. Image of five standard fabrics after erosion and dilation.

The initial condition $f_0(x, y)$ is set to $f(x, y)$. The surface area at each step can be then calculated as follows:

$$\begin{aligned} S(X, Y, \varepsilon) &= \frac{V(X, Y, \varepsilon)}{2\varepsilon} \\ &= \frac{\sum_{x \in [1, N], y \in [1, M]} (f_\varepsilon^u(x, y) - f_\varepsilon^l(x, y))}{2\varepsilon} \end{aligned} \quad (15.24)$$

The value of D can be estimated by plotting $\log(S(X, Y, \varepsilon_i))$ vs. $\log(\varepsilon_i)$ for a given set of scale factors ε_i , $i = 1, 2, \dots, N$ and calculating the gradient of the line that fits the plot. The natural logarithm of the surface area $S(\varepsilon)$ versus natural logarithm of scale ε for fractal surface of five standard images can be found in Figure 15.9, which indicates that it is difficult to get a good curve fitting using the straight-line equations.

The curve-fitting process was applied using only three pairs of data $\{\log(1/\varepsilon), \log(S(\varepsilon))\}$ each time. This was done to highlight the variability in the "local slope" in the log-log data and to focus on the dimension estimates for different polar fleece fabrics with small, medium, and large pilling particles. Table 1 shows the estimated fractal dimensions at different scales of five standard polar fleece fabric images.

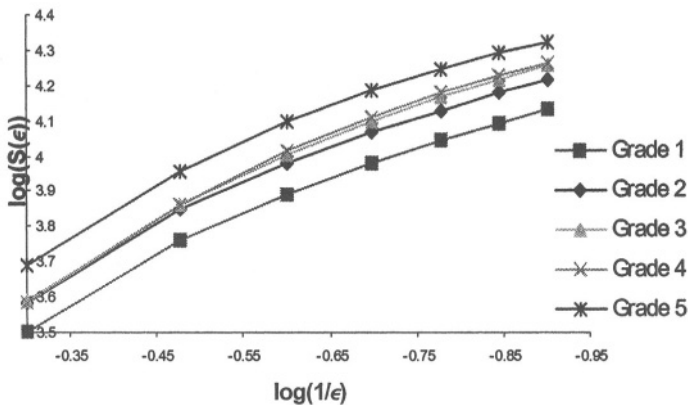


Figure 15.9. Logarithmic plot of surface area versus scale for five standard images from Grade 1 to Grade 5.

Figure 15.10 shows the correlation between fractal dimensions estimated and its relative scale, according to which it is evident that the real surfaces found in fabric images are not perfect fractal surfaces, and thus the classification of real surfaces cannot only depend on a single value for the fractal dimension. As a result, a new fractal concept is introduced named Extended Morphological Fractal Analysis. The principle of Extended Morphological Fractal Analysis is to treat fractal dimension as a function of measuring scale. A fractal vector V_f , $V_f = [V_f, V_1, V_2, \dots, V_n]$ where V_i is the fractal dimension estimated at scale i , the fractal dimensions calculated at different scales, is thus adopted to represent the fabric textures. The Multiscale Fractal Vector is apparently superior to a single fractal dimension since it contains more texture roughness information.

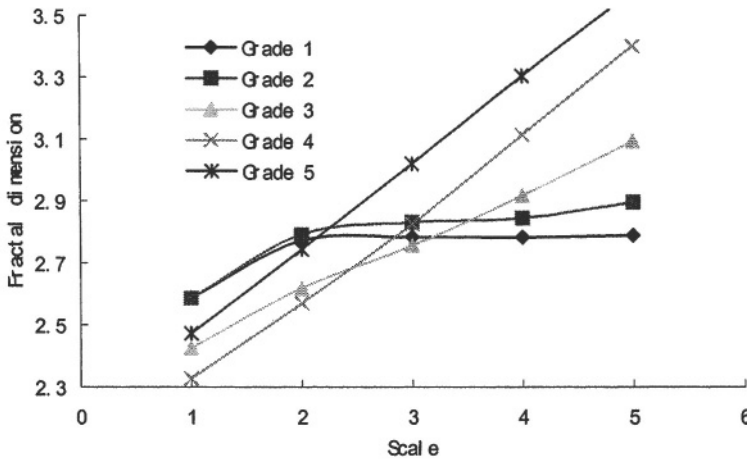


Figure 15.10. Correlation between fractal dimension estimated and its relative scale.

15.3.3 Grade Classification

The obtainment of the Multiscale Fractal Vector is the first step for objective fabric image assessment and this is followed by System Training accomplished using standard fabric images. While the grade of a fabric image is defined as the grade of the closest standard image in the feature space.

To account for the closeness between one image and five grade clusters, a simplified Bayes distance is considered. Assumed that the features are independent and Gaussian and the Bayes distance provides the maximum likelihood (ML) classification, the likelihood function for a feature vector $\vec{v}$ belonging in fabric appearance grade l is

$$p(\vec{V} = \vec{v} | \vec{L} = \vec{l}) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_{i,l}} e^{-((v_i - u_{i,l})^2 / 2\sigma_{i,l}^2)} \quad (15.25)$$

where v_i is the element of a feature vector, and n the total number of features. The grade classification is achieved by choosing the grade l , which minimizes the simplified Bayes distance function

$$d_l = \sum_{i=1}^n [2 \ln \sigma_{i,l} + (\frac{v_i - u_{i,l}}{\sigma_{i,l}})^2] \quad (15.26)$$

Five fractal dimensions at different scales can be used to make up a fractal vector, and the Bayes distance function can be further simplified as below:

$$d_l = \sum_{i=1}^n (v_i - u_{i,l})^2 \quad (15.27)$$

Table 15.1. Bayes distance between 5 testing specimens and 5 standard grade clusters.

No.	D ₁	D ₂	D ₃	D ₄	D ₅	Estimated Grade	Subjective Grade
1	1.041	0.794	0.507	0.159	0.003	5	4.5
2	0.104	0.076	0.011	0.240	0.616	3	2.6
3	0.122	0.206	0.271	0.804	1.438	1	2.6
4	0.357	0.215	0.148	0.136	0.152	4	3.9
5	0.122	0.050	0.048	0.222	0.411	3	3.4

Table 15.1 is an example illustrating a comparison between the proposed objective grading system and subjective grading on 5 fabric samples. Apparently, the generated result of Bayes Classification Method is very reliable.

15.4. Modeling of Fabric Wrinkling

15.4.1 Introduction

Wrinkle is another important factor which can badly deteriorate the quality of a textile product. It is by nature a three-dimensional crease, forming when fabrics are forced to develop high levels of double curvature. The study of wrinkle assessment starts from the early 1950s. The traditional method of wrinkle assessment is also very subjective, i.e. allowing expert observers to compare fabric specimens with a set of six three-dimensional replica plates, and then assigning a grade according to their similarity.

Many attempts have also been done to automate this characterization process using imaging technology[2, 12, 14, 18]. Laser probe is one way of acquiring surface from a fabric specimen to measure surface height variation [14,18], which proposes obvious physical meaning and not influenced by color and pattern. However, the slow point-scanning process and high costs limit its popularization in industrial applications. Video camera with common lighting system is another consideration[12]. The advantage of video camera lies in that it can acquire good resolution fabric images quickly, but it also suffers the problem of high sensitivity

to fabric colors and patterns. Hence it is mainly used to assess fabrics without patterns or designs. Line laser profilometer is also a good attempt to improve the detecting efficiency, but its problems are that line profiles cannot replace whole fabric surface strictly [2] and too many fabrics are needed to produce a reliable result. Recently, a shape measurement method called Photometric Stereo receives much attention for quality control and inspection of objects in engineering and industrial fields [26]. The idea of photometric stereo is to vary the direction of incident illumination between successive images, keeping the viewing direction constant. This method can provide sufficient information to determine surface orientation at each image point. The technique is photometric because it uses the radiance values recorded at a single image location, in successive views, rather than the relative positions of displaced features. Finally, the 3-D shape of fabric surface is recovered from the surface normal at each image point. Photometric stereo is thus a reliable technique for wrinkle assessment since wrinkle measurement is not affected by color and pattern of fabrics.

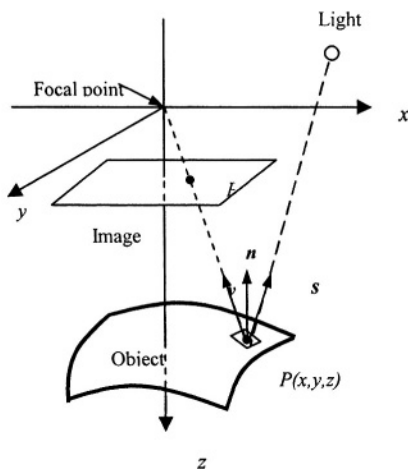


Figure 15.11. Surface model and observation system.

15.4.2 Theory of 3-D Surface Reconstruction

Fabric Surface Reflectance Model. When a light ray strikes the surface of fabric, specular and diffuse reflections take place. These reflections are very much governed by surface microstructure, incident wavelength, and the direction of incidence [23]. However, it is acceptable to visualize most of fabrics' surface as Lambertian surfaces, which scatter incident light equally in all directions and appear equally bright from all directions.

According to Lambert's cosine law ^[8], the intensity of an image element P' corresponding to a Lambertian reflecting surface is given by the relationship $I(x, y) = c(x, y) \cos \theta$, where $c(x, y)$ is the reflective parameter of corresponding surface element P and θ the incident angle at this element. As shown in Figure 15.11, P , n , s , v are respectively a surface element of an object, normal vector of P , and vector of P , incident vector of P , and vector of sight of P . $\cos \theta$ can be expressed by $\cos \theta = n \cdot s$. Apparently that different colors or patterns propose different $c(x, y)$ so the influence of color and pattern can thus be eliminated if $c(x, y)$ can be calculated.

15.4.3 Lighting System

Four parallel light sources illuminating fabric specimens from four different directions with the same radiance intensity E_0 are used as incident light as shown in Figure 15.12, in which l and w denote the length and width of the light source, α the illuminating angle of four parallel light sources and R_l , R_m , R_r the distances between light source and left, middle, right parts of fabric surface separately. According to photometry theory [9], irradiance of one surface element $P(x, y)$ is

$$E(x, y) = \frac{E \cos \alpha}{R^2(x, y)} \quad (15.28)$$

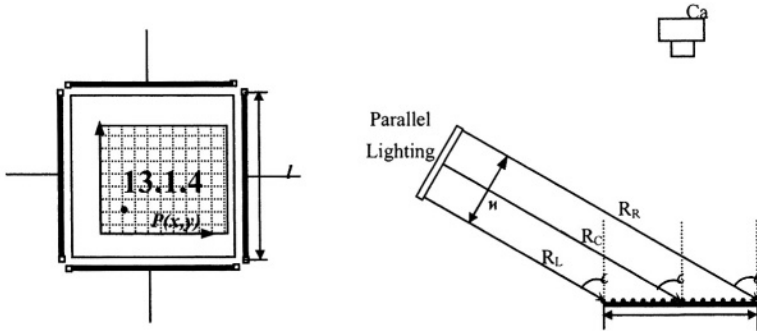


Figure 15.12. Lighting System.

$R(x, y)$ is the distance between light source and P , which can be calculated from x, y easily. When p and q are the first partial derivatives of z with respect to x and y , the normal vector of a surface element is given as

$$n = \frac{(p, q, -1)}{\sqrt{p^2 + q^2 + 1}} \quad (15.29)$$

Figure 15.13 illustrates a visual image of each lighting vector. Supposing I_e, I_w, I_s, I_n are image pixels of the same surface element $P(x,y)$ captured under four different illuminating sources sequentially, we can get four equations as below.

$$\begin{cases} I_e(x, y) = E_e(x, y) \cdot c(x, y) \cdot \cos \theta_e \\ I_w(x, y) = E_w(x, y) \cdot c(x, y) \cdot \cos \theta_w \\ I_s(x, y) = E_s(x, y) \cdot c(x, y) \cdot \cos \theta_s \\ I_n(x, y) = E_n(x, y) \cdot c(x, y) \cdot \cos \theta_n \end{cases} \quad (15.30)$$

$E_e(x, y), E_w(x, y), E_s(x, y), E_n(x, y)$ are irradiances of this surface element $P(x,y)$ under four different lighting sources separately and the expressions of $\cos \theta_e, \cos \theta_w, \cos \theta_s, \cos \theta_n$ are

$$\begin{cases} \cos \theta_e = \frac{\sin \alpha + \cos \alpha \cdot p}{\sqrt{p^2 + q^2 + 1}} \\ \cos \theta_w = \frac{\sin \alpha - \cos \alpha \cdot p}{\sqrt{p^2 + q^2 + 1}} \\ \cos \theta_s = \frac{\sin \alpha - \cos \alpha \cdot q}{\sqrt{p^2 + q^2 + 1}} \\ \cos \theta_n = \frac{\sin \alpha + \cos \alpha \cdot q}{\sqrt{p^2 + q^2 + 1}} \end{cases} \quad (15.31)$$

From the above equations, we can get the surface normal gradients p, q and $c(x,y)$ by cross multiplications and transpositions.

$$\begin{cases} p = \frac{I_e E_w - I_w E_e}{I_e E_w + I_w E_e} \cdot \operatorname{tg} \alpha \\ q = \frac{I_n E_s - I_s E_n}{I_n E_s + I_s E_n} \cdot \operatorname{tg} \alpha \\ c = \frac{I_e \cdot \sqrt{p^2 + q^2 + 1}}{\sin \alpha + \cos \alpha \cdot p} \end{cases} \quad (15.32)$$

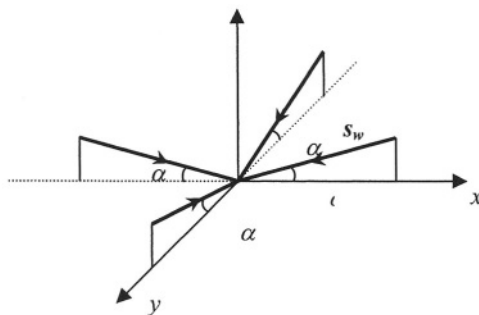


Figure 15.13. Four direction lighting.

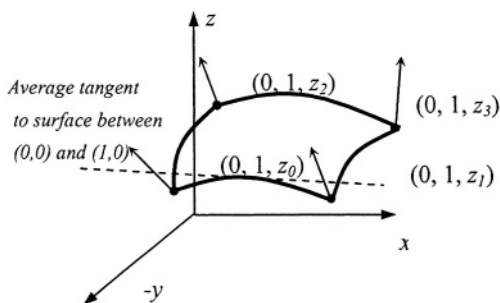


Figure 15.14. Surface patch and normal vectors. Approximation to surface between points $(0,0)$ and $(1,0)$ can be made by using the average tangent line if points are sufficiently close.

15.4.4 Reconstruction

The final step in generating the actual surface is the conversion from surface normal to depth information. That is, for every (x, y) point and normal vector N at (x, y) , a z value with respect to the image plane must be computed.

Assuming a surface patch with known surface normal N_0, N_1, N_2, N_3 at the points $(0,0), (1,0), (0,1), (1,1)$ as shown in Figure 15.15, a starting z value at point $(0,0)$ is thus either chosen or known. The task left thus becomes to choose a function to compute z values at the remaining three points. Considering what dealing with is the distance between pixels, the points $(0,0)$ and $(1,0)$ are thus very close. As a result, the curve between these points can be approximated by its average tangent line. Given the following normal vectors:

$$N_0 = (n_{0x}, n_{0y}, n_{0z}) \text{ AT } (0,0)$$

$$N_1 = (n_{1x}, n_{1y}, n_{1z}) \text{ AT } (1,0)$$

$$N_2 = (n_{2x}, n_{2y}, n_{2z}) \text{ AT } (0,1)$$

$$N_3 = (n_{3x}, n_{3y}, n_{3z}) \text{ AT } (1,1)$$

z at (1,0) can be expressed as

$$ax + b(z(1,0) - z(0,0)) = 0 \quad (15.33)$$

where

$$a = (n_{0x} + n_{1x}) / 2 \quad (15.34)$$

$$b = (n_{0z} + n_{1z}) / 2 \quad (15.35)$$

This gives $z(1,0) = z(0,0) - x(a / b)$ with $x=1$.

Similarly, approximation along the y axis to find z at (0,1) gives

$$z(0,1) = z(0,0) - y(a / b) \text{ with } y=1 \quad (15.36)$$

$$\text{Here} \quad a = (n_{0y} + n_{1y}) / 2 \quad (15.37)$$

$$b = (n_{0z} + n_{1z}) / 2 \quad (15.38)$$

And $z(1,1)$ can be expressed as $z(1,1) = (z(1,0) + z(0,1)) / 2$

If z at (1,1) is known, the values of z at other three points can be computed along the negative x and y direction.

The algorithm for depth conversion can start with choosing an arbitrary z value for the point in the center of the image. z values can thus be determined at all points along the x and y axis passing through this center point as shown in Figure 15.15 (a) z values are thus computed for the remaining points in each quadrant in the order as shown in Figure 15.15 (b). The reconstructed 3-D image of fabric sample (Grade 1) is shown in Figure 15.16.

15.4.5 System Set-up

The 3-D wrinkling measurement system consists of a color digital camera, one lighting box, a frame grabber and a personal computer. The highest resolution of digital camera can be 1600 pixel $\times$ 1200 pixel, the parallel lighting can be

controlled in four directions in the special lighting box, and the personal computer contains an image analysis software.

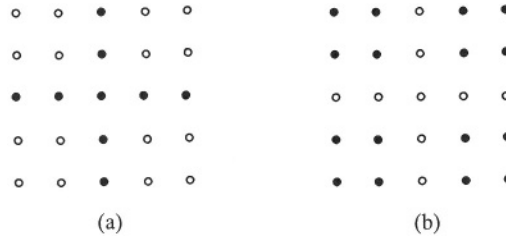


Figure 15.15. Illustration of depth conversion order for surface points.

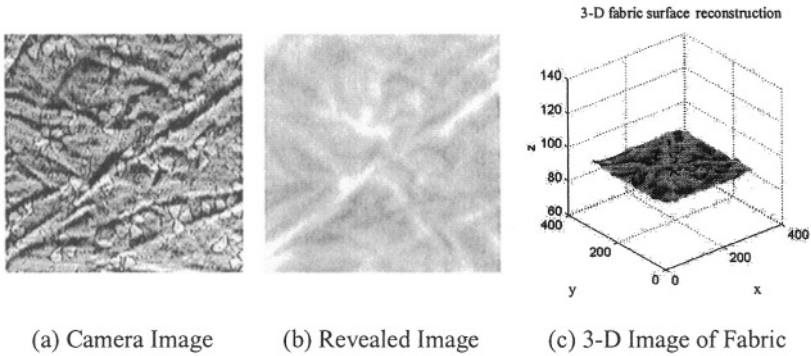


Figure 15.16. Fabric surface reconstruction.

15.4.6 Feature Extraction

Supposing a surface element is flat, its normal vector is thus $(0, 0, -1)$. For the surface element of wrinkling parts in fabric surface, its absolute value of p , q will be larger than other regions. The distributions of p of different fabric wrinkling grades are shown in Figure 15.17.

P and Q can be used to describe the wrinkling status of fabrics, where

$$P = \frac{1}{N} \sum_{i=1}^N |p(i)| \quad (15.39)$$

$$Q = \frac{1}{N} \sum_{i=1}^N |q(i)| \quad (15.40)$$

$P(i)$, $q(i)$ are the first partial derivatives of z with respect to x and y of surface element i and N the number of surface elements of each image. P describes the

wrinkling in x direction, Q the wrinkling in y direction and $P+Q$ represents the wrinkling of whole fabric surface.

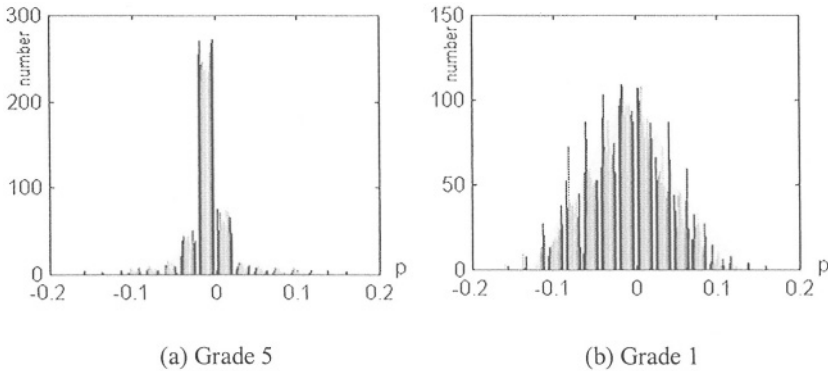


Figure 15.17. Distribution of p .

15.4.7 Grade Rating

Table 15.2 shows examples of the proposed method together with a comparison with subjective evaluation, in which sample A, B differ in patterns and colors. Apparently, there is a high correlation between $P+Q$ and the subjective grading result. Moreover, P of B1 is higher than B2, but $P+Q$ of B1 is lower than B2, which implies that it is better to describe fabric wrinkling of whole surface using $P+Q$ instead of P or Q .

15.5. Summary and Conclusions

In this chapter, we describe the modeling and analysis of fabric appearance using image techniques and state-of-the-art of objective evaluation of textile surfaces for quality control in textile industry. Analysis of three major textile surface appearance attributes (pilling, wrinkling and polar fleece fabric) is given together with a detailed explanation of the modelling methods for their objective assessment. The conclusions reached include:

- The objective assessment of fabric pilling can be successfully accomplished through pill templates trained from "the actual pill image" using "Pill Template Training" and "Pill Template Matching". However, the current pill template is only applicable to solid-colored fabrics. For printed fabrics with many colors, a new pill template is needed which contains information of both the pill shapes and the colors.

Table 15.2. Examples of objective and subjective measurements.

Fabric Code		P+Q	P	Q	Subjective Grading
Sample A	A1	0.035971	0.020097	0.015874	5
	A2	0.041352	0.022775	0.018577	3.6
	A3	0.044955	0.024703	0.020252	3
	A4	0.059857	0.035655	0.024202	2.1
Sample B	B1	0.024463	0.012993	0.01147	5
	B2	0.024621	0.011665	0.012956	4.2
	B3	0.028088	0.012697	0.015391	4.1
	B4	0.038737	0.018134	0.020603	2.2

- Extended morphological fractal method can be applied to analyze the appearance of polar fleece fabrics since it eliminates the problem that the statistical features of real surfaces cannot be fully represented by a single value of fractal dimension. The roughness of fabric surface can be used to describe the fractal vector while the grading is accomplished by Bayes classification method. A good correlation was observed between the output of the proposed objective evaluation method and subjective grading result.
- The evaluation of fabric wrinkling is conducted using Photometric Stereo. Based on the 3D image extracted from four camera images taken from different illuminating directions, an effective feature $P+Q$ of 3D image is used to describe fabric wrinkling. The results indicate a great success of the application of photometric stereo in objective evaluation of fabric wrinkling.

Acknowledgments

The authors would like to express their appreciations to the financial support of the Hong Kong Polytechnic University and the editing effort made by Dr Xiuying Jiang to this chapter.

References

- [1] Konda. "Evaluation of pilling by computer image analysis". *J. Textile March. Soc.* Japan, vol. 36, no. 3, pp. 96-101, 1990
- [2] B. Horn, R. Sjoberg. "Calculate the reflectance map". *Artificial Intell. Lab.*, M.I.T., AIM-498, 1978
- [3] B. Xin and X. Yu. "Automatic measurement of the yarn density of woven fabric". *Journal of China Textile University*, vol. 25, no. 3, pp. 34- 37, 1999.

- [4] B. Xu and D.F. Cuminato, N.M. Keyes. "Evaluating fabric smoothness appearance with a laser profilometer". *Textile Res. J.*, vol. 68, no. 12, pp. 900-906, 1998.
- [5] B. Xu. "Instrumental evaluations of fabric pilling". *J.Text.Inst.*, vol. 88, pp. 488-500, 1997.
- [6] B. Xu, J.A. Reed. "Instrumental evaluation of fabric wrinkle recovery". *J.Textile Inst.*, vol. 86, no. 1, pp. 129-128, 1995.
- [7] B.B. Mandebrot. *Fractals: Form, Chance and Dimension*. San Francisco: Freeman, 1977.
- [8] C. A. Hector, M. S. Millan, Y. Torres, and R. Navarro. "Automatic method based on image analysis for pilling evaluation in fabrics". *Optical Engineering*, vol. 37, no. 11, pp. 2937-2947, 1998.
- [9] C.H. His, R.R. Bresee, and P.A. Annis. "Characterizing fabric pilling by using image-analysis techniques, Part I: Pill detection and description". *J.Textile Inst.* Vol. 89, no. 1, Part I, pp. 80-95, 1998.
- [10] C.J. Krebs. *Ecological Methodology*. Harper & Row, New York, NY, USA, 1989.
- [11] D.D. Thibodeaux. "Cotton fiber maturity by image analysis". *Textile Res. J.*, vol. 56, pp. 130-139, 1986.
- [12] E. H. Kim. "Objective evaluation of wrinkle recovery". *Textile Res. J.*, vol. 69, no. 11, pp. 860-865, 1999.
- [13] F. Thorr, J.Y. Drean, and D. Adolphe. "Image analysis tools to study nonwovens". *Textile Res. J.*, vol. 69, no. 3, pp. 162-168, 1999.
- [14] J. Amirbayat and M.J. Alagha. "Objective assessment of wrinkle recovery by means of laser triangulation". *J. Textile Inst.*, vol. 87, no. 2, Part I, pp. 349-354, 1996.
- [15] J. Serra. *Image Analysis and Mathematical Morphology*. vol. 1. Academic Press, 1982.
- [16] J.R. Wagner & Associates. "Development of an image analysis algorithm for assessing pilling". *1999 Nonwovens Conference, TAPPI Proceedings*, pp. 305-311, 1999.
- [17] L. Andrew. "Contribution concerning the detection of defects on small-patterned textiles". *Melliand Textilber*, vol. 72, pp. 981-984, 1991.
- [18] M.G. Dobb, S.J. Russell. "A system for the quantitative comparison of wrinkling in plain fabrics". *Journal of the Textile Institute*, vol. 86, no. 3, pp. 495-497, 1995.
- [19] M.L. Smith, G. Smith, and T. Hill. "Gradient space analysis of surface defects using a photometric stereo derived bump map". *Image and Vision Computing Journal*, vol. 17, no. 3-4, pp. 321-332, 1999.
- [20] N. Youngjoo and B. Pourdeyhimi. "Assessing wrinkling using image analysis and replicate standards". *Textile Research Journal*, vol. 65, no. 3, pp. 149-157, 1995.
- [21] Pourdeyhimi, B. Xu, and J. Sobus. "Evaluating carpet appearance loss: surface intensity and roughness". *Textile Res. J.*, vol. 63, no. 9, pp. 523-535, 1993.
- [22] Pourdeyhimi, B. Xu, and L. Wehrel. "Evaluating carpet appearance loss: periodicity and tuft placement". *Textile Res. J.*, vol. 64, no. 1, pp.21-22, 1994.
- [23] R., Ed. Kingslake. *Applied Optics and Optical Engineering*. vols. I and II. New York, Academic, 1965.

- [24] R.B. Ramgulum. "The objective assessment of fabric pilling, Part I: methodology". *Textile Inst.*, vol. 84, no. 2, pp.221-223, 1993.
- [25] R.H. Gong and A. Newton. "Image-analysis techniques, Part II: the measurement of fiber orientation in nonwoven fabrics". *J.Text.Inst.*,vo. 87, no. 2, pp. 371-388, 1996.
- [26] T.J. Kang, D.H. Cho, H.S. Whang, "A new objective method of measuring fabric wrinkles using a 3-D projecting grid technique". *Textile Research Journal*, vol. 69, no. 4, pp. 261-268, 1999.
- [27] U. Roster. "Defect detection of fabrics by image process". *Melliand Textilber*, vol. 73, pp. 635-636, 1992.
- [28] W.D. Cooke. "Pilling attrition and fatigue". *Text. Res. J.*, vol. 55, pp. 409-414, 1985.
- [29] Y.F. Zhang and R.R. Bresee. "Fabric defect detection and classification using image analysis". *Textile Res. J.*, vol. 65, no. 1, pp. 1-9, 1995.
- [30] Z. Fazekas, J.Komilves, I. Renyi and L. Surjan. "Towards objective visual assessment of fabric features". *Image Processing and its Applications*, Conference Publication No.465 ©IEE 411-416, 1999.

This page intentionally left blank

Chapter 16

VIRTUAL PRODUCT PRESENTATION BASED ON IMAGES AND GRAPHICS

Zhigeng Pan ^{1,2}, Mingmin Zhang ¹ and Tian Chen ¹

1. State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China

2. VR and Multimedia Research Center, HZIEE, Hangzhou, China

{zgpan,zmm,chentian}@cad.zju.edu.cn

Abstract In this chapter, we discuss the research issues and state-of-the-art of virtual product presentation based on image and graphics. Through reviewing and comparing several online product presentation web sites embedded 3D visualization technology, especially the current virtual shopping malls web sites which integrate Virtual Reality into E-commerce, this chapter presents both the advantages and disadvantages of two kinds of product showing methods: 2D image based and 3D model based presentation method. Subsequently, take our EasyMall (virtual shopping mall) and EasyShow (virtual presentation of textile product) as showcases, the techniques for virtual product presentation based on image and graphics are described in detail. Our developing practice shows that diversified product presentation methods should be available in different cases.

Keywords: Product Presentation, product customisation, virtual reality, E-commerce, texture mapping, texture smooth, texture morphing, intelligent sales assistant, image-based rendering

16.1. Introduction

The wide-range applications of the Internet, computer graphics, man-machine interface and the expressiveness of multimedia offer more and more opportunities for developing electronic commerce (E-commerce). E-commerce and online shopping applications are expected to become one of the fastest growing fields in the Internet market [1].

Nevertheless, most of the current E-commerce platforms only provide users simple, 2D image-based and text-based interfaces or some flash animations to

access the products. Such kind of tedious environment neither provides users the same shopping experience as they may get in an actual store or shopping mall, nor allows them to customize some personalized products which meet their taste. Furthermore, it also fails to make consumers enjoy fun and evoke an appetite for purchasing something.

As a result, the functions of most commerce websites for production presentation and customization are impaired somewhat. Practically they just play a role of introducing products by indicating their apparent properties, such as the color, size, appearance, etc.

16.1.1 Motivation

Virtual Reality (VR) is a new and attractive human-computer interactive interface technology, which is becoming one of the hottest research and development areas in computer industry today. VR technology is being applied in wide domains and touching on many new fields. Also, VR offers a high potential for product presentation: instead of regarding flat, static pictures, configurable and animated 3D models embedded in entertaining environments provide a new way of product presentation. How to emerge VR technology in building the E-commerce platform, so as to exhibit products effectively and provide customers powerful function of customizing products conveniently is an important problem to solve. In the same time, we have to agree on that there are still many advantages of using 2D images to represent and customize products, especially in some special cases.

Regarding our current developing work related to this field, we have developed EasyShow system for virtual presentation of textile products, which is based mainly on images. In addition, we implemented a virtual mall system (called EasyMall), integrating product presentation and customization function. In EasyMall, we focus on analyzing the usage of 3D model for setting up the virtual purchase environment.

16.1.2 Organization of This Chapter

The remainder of the chapter is organized as follows. In Section 16.2, work related to this research is first reviewed and compared. And both the advantages and disadvantages of two kinds of product exhibition methods: 2D image based and 3D model based presentation methods are discussed.

In Section 16.3, we present EasyShow, a virtual product presentation system for textile product. And detailed techniques for image-based product presentation are discussed.

In Section 16.4, with our EasyMall as a showcase, the architecture of the system and the development technologies, especially the graphics-based presentation methods are analysed. Finally, based on our practice and current

website situation, we discuss how to improve the online service quality to lead to more bargains through the variable, convenient and realistic product presentation methods.

16.2. Related Work

E-Commerce involves individuals as well as companies engaging in a variety of electronic business transactions using computer and telecommunication networks. Traditionally, the definition of E-Commerce focused on Electronic Data Interchange (EDI) as the primary means of conducting business electronically between companies which have pre-established contractual relationship. Recently, however, due to the WWW's surge in popularity and the acceptance of the Internet as a viable transport mechanism for business information, the definition of E-Commerce has been broadened to encompass business conducted over the Internet and includes individuals and companies not previously known to each other [2].

The first researchers, Hoffman proposed a framework for examining the commercial development of the Web in 1995[3]. After that, E-commerce has been developed rapidly. From the studies of the usability of current E-commerce sites, it is a general problem that buyers fail to find what they are looking for, or abandon the purchase even though they have found the product [4]. Most reports suggest that promoters are disappointed with the current level of online sales [5]. Nevertheless, E-commerce is undoubtedly the best media and tool for presenting product online.

However, the aim of online presentation is not only for exhibiting product, but for increasing the purchase rate by letting the customer know products in more detail. Therefore, one of the challenges of E-commerce is the design on web sites which effectively present products and are convenient and enjoyable for buyers to use.

16.2.1 VR and Interactive Virtual Experience

VR was defined originally as a technological advance of that involves human senses (e.g., vision, hearing, touch) through input and output devices. As Heim pointed out, VR has "three I's" – immersion, interactivity, and information intensity [6]. Immersion is usually achieved through a head mounted display (HMD) or a CAVE environment [7]. A desktop virtual reality usually is equipped with stereoscopic glasses and maneuvering device (e.g., data glove, joy stick or space mouse). Currently, VR devices are still not very popular because of its expensive price. And most of them are used in some research universities and military departments. However, the immersive feeling also can be produced only if the object is expressed with the simple Virtual Reality Modeling Language (VRML).

Interactivity makes it possible that human feel the virtual experience just like the actual direct experience to objects or an environment. Lundh defined such an experience as an event or process that can occur spontaneously or voluntarily within everyday situations but always involve the internal awareness of something taking place [8].

Biocca et al [9] defined two conventional types of experience when consumers evaluate products: direct experience (for example, product trial) and indirect experience (for example, looking at product brochure). A direct product experience involves all the senses including 'visual, auditory, taste-smell, haptic (touch sense) and orienting' [10]. The virtual experience was defined as "psychological and emotional states that consumers undergo while interacting with products in a 3-D environment" [9]. Interactivity makes the virtual environment be capable of providing human different feedbacks in response to different actions performed by them.

In most cases, virtual experience derived from interactive media such as 3-D virtual environments is richer than indirect experience obtained from traditional advertising or others. With the vivid and media-richen interactivity, many 3-D virtual products and shopping malls are not just a representation of physical products and malls; instead they are simulations of the consumption experience.

16.2.2 Integrating E-commerce with VR

Studies on VR interfaces emerged into E-commerce sites began appearing on the Internet several years ago. Matsushita uses VR for product presentation in his Virtual Kitchen project [11], a retail application set up in Japan to help people choose appliances and furnishings for the rather small kitchen apartment spaces in Tokyo. Users bring their architectural plans to the Matsushita store, and a virtual copy of their home kitchen is programmed into the computer system.

In 1994, Fraunhofer IAO and the British software company Division presented the Cooperative-Interactive Application Tool (CIA-Tool) which consists of a VR based system for immersive placement and surface adjustment of interior design objects in offices [12].

In 1997 at the IAA motorshow in Frankfurt, Germany, Mercedes-Benz introduced the "Virtual Car" Simulator to display its A-class model. This simulator allows users to hold a screen in their hands to make selections for colours etc. The Virtual Design Exhibition was presented in [13], which founded by Fraunhofer IAO in collaboration with the Milanese Design and Architecture bureau Studio De Lucchi. The goal of the project was to offer a new way of presenting products of interior design furniture manufacturers. The Virtual Design Exhibition consists of an exhibition part and of a set of tools for interactive product configuration. However, according to statements presented in [14] immersive VR applications still lack an easy-to-use interface. Currently, there are

many existing 3D virtual shopping malls. Let's review several typical ones in the following section.

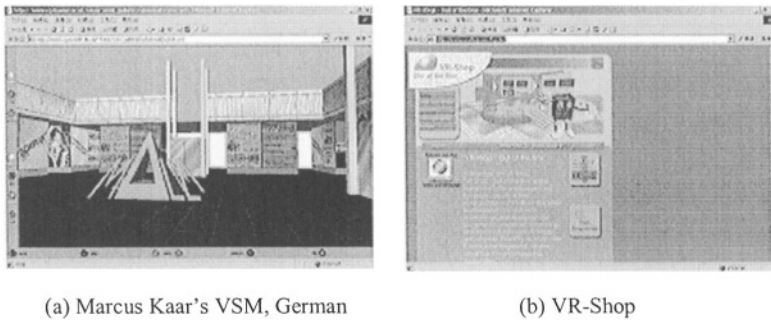


Figure 16.1. Snapshots of several existing 3D virtual shopping malls.

Virtual Shopping Mall (VSM), as a prototype system, allows users to choose their own figures from several simple avatars during navigation [15], shown in Figure 16.1 (a). It just like VNet, realizes a shared virtual environment based on VRML and Java. Before connecting to the server the user has to specify his name and select a custom or one of the built-in avatars. The function is very like the former one. After the user connects to the server, he is able to chat with other users, to move around in virtual space and to watch the avatars of other users.

VR-Shop[16], as a demo installation of a virtual shopping environment within a 3D online community, is based on Blaxxun VRML multi-user technology and is capable of providing advanced visualizations in return for a rather lengthy plug-in download, shown in Figure 16.1 (b). VR-Shop enables companies to offer a complete and efficient service solution for their customers for a faster and better way to communicate and enter the marketplace.

From Figure 16.2 (a), we can see a snapshot of Active Worlds, an excite net of interlinked, three-dimensional virtual environments. Unlike other similar systems, Active Worlds provides toolkits to build interactive virtual community. @mart in Active Worlds is an exemplary world which shows how the toolkit can be used to build a 3D virtual shopping mall. Users can get 2D information on objects by clicking them [17]. Based on the project of ATLAS, CDSN Lab, Information and Communications Univ., Korea, developed a 3D shared virtual shopping mall [18]. The main goal of this project is to design and implement a scalable network framework for large distributed virtual environments. A screen capture is shown in Figure 16.2 (b).

The Agent-aided Collaborative Virtual Environment (CVE) over HLA/RTI was presented by MCRLab, Univ. of Ottawa[19]. Figure 16.2 (c) shows a snapshot of Client's View of the running application. Their CVE for E-commerce bridges the

gap between the Industrial Training and E-commerce. It provides the user with a much enjoyable experience while shopping online. Under their virtual environment, the users, represented by avatars, can join a Virtual World and then manipulate and interact with the objects as in the real world.

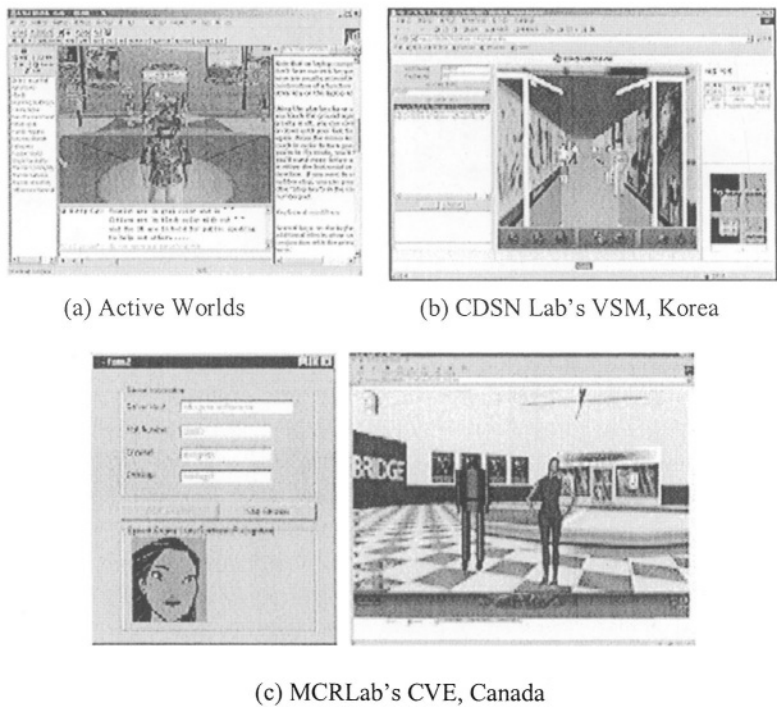
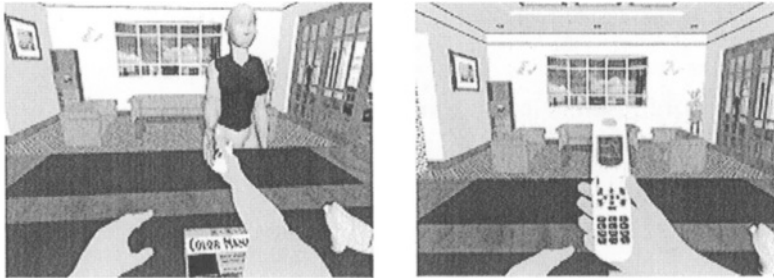


Figure 16.2. Snapshots of several existing 3D virtual shopping malls (continued).

At VR Lab, POSTECH, Korea, researchers developed an OpenGL-based customization and presentation system for hand phone (called PhoneShow), integrating virtual human into product presentation [20,21]. Users can customize each component, the position, the texture color, the model style etc. Figure 16.3 (a) shows the screenshot of the user getting the hand phone from Cally, an assistant Avatar for exhibition. In Figure 16.3 (b), user rotates, manipulates the phone with 'his own hand' after getting it. Thus, it makes the product to be embedded in a lively environment.

In addition, some other web sites on the Internet are also making attempts at providing users with 3D user interfaces (Examples are Cybertown Shopping Mall at www.cybertown.com, Eluxury online shopping site at www.eluxury.com, FAO

schwarz virtual playroom at <http://www.fao.com>), allowing them to explore a VR representation of the store.



(a) The user getting the phone from Cally. (b) The user manipulates the phone.

Figure 16.3. Snapshots of PhoneShow system.

16.2.3 Product Presentation Methods

Broadly speaking, there are only two kinds of presentation methods: 2D image based presentation method and 3D model based presentation method. Then one question is which kind of method would be better to present product in E-commerce?

Integrating VR into E-commerce, the world of E-commerce is entering a new realm previously 'thought impossible' [22]. Since most products are 3D objects that are experienced with the senses, the use of dynamic and compelling 3D visualization in E-commerce is increasing as companies seek to give users an innovative experience of the product. A 3D model can offer varying degrees of viewing, but a standard 2D image can only appeal to the visual sense. 3D model based method to represent products enables consumers to 'interact' with products on the Internet rather than just look at them.

However, a different view is that of Neilson [23], he states 2D is better than 3D, and virtual shopping malls are just a gimmick. In [24], Hurst also points out that the '3D product sites are implementing a feature just because they can. He expands on this to say that online shoppers do not need high-tech gadgetry and that it simply complicated matters. Today, the use of 2D image based method to represent products in E-commerce is still commonplace. Thousands of E-commerce sites have used this method to represent their products to consumers. There are still many advantages of using 2D images to represent products. Among them, cost is an important factor as it is relatively cheap to obtain a picture of a product with a digital camera, which is available everywhere

Regarding of the present situation, it seems that we can not distinguish which kind of presentation method is better. Both of these two kinds of presentation methods are important and valuable. In different cases, we need different methods to present different kinds of products.

16.3. Image-based Virtual Presentation of Products

EasyShow is an image based presentation and customisation system for bed clothing and apparel. It employs the 2D image based presentation method. As to the customisation, the product matching is mainly considered, including the colour matching, the texture matching, the style matching etc. For example, when a consumer chooses a shirt, he may want to choose a piece of tie to match his shirt. Another case, after a consumer chooses the style of his bedspread, he may want to check the showing effect of different textures, different material, also, to make sure if it matches his bed.

In order to produce the 3D effect, firstly we need to pre-process the ordinary picture of cloth, including adjusting colour, light, etc. Secondly, we provide the interactive interface, the guiding tools, to help user select the region the texture should be attached, where the Snake technique is applied. Thirdly, the system offers variable degrees of depth through texture deformation (e.g. regular disturbing, regular shade), joint and smooth transition, offers the pleat vision through the fusion of colour and light. In addition, different light models for different kinds of material, the cotton, the silk, the leather, are considered.

As we know, textile emulation in visual presentation is a difficult problem for many years, therefore the experts domestic and abroad have focused on this technique for long time. There are many algorithms presented until now. But most of these algorithms have to construct textile models. For example, the algorithm presented by Weil constructs a textile model with droop feeling through geometry calculation [25]. Terzopoulos et al. presented a textile model in 1987[26-28]. The model simulated the behavior of textile through a differential equation based on the elasticity theory.

A numerical code has been developed to model the complex behavior of woven fabric panels subjected to ballistic impact by Roylance et al in 1995[29]. Chen et al. presented an energy method to construct textile model [30]. It is more difficult to use these algorithms in the textile visual presentation. First it takes long time to build the result image. For instance, for the result image created by the 30×30 grid, Chen's energy algorithm needs 10 to 15 minutes. Second, the feature of the textile is indispensable to textile modeling. Moreover, the modeling step takes a long time and needs participation of experts.

Wang et al simulated the presentation by building a grid in the target area and adjusted the texture coordinates of grid vertexes by their lightness. But this algorithm is only suitable for the garment simulation [31].

To address this technical problem, we design a method based on image processing to simulate the realistic textile, achieve the realistic effects of the textile according to a new texture morphing method. This method is integrated into EasyShow system, and it is real-time, and is very suitable for the visual textile presentation of E-commerce.

16.3.1 Algorithm Overview

Here we give some definitions before introducing the algorithm.

Definition 16.1. The **U (V) texture vector** of a surface is the direction through which the horizontal (vertical) line in the texture image is mapped. The angle of the texture vector is between $(-\pi, \pi)$. And the surface is called as **texture block**.

Definition 16.2. The **connected texture region** includes some texture blocks connected with each other. They must have a same texture vector, the **base texture vector**. And the vector is close to the direction of common borders (the difference between them is smaller than a given threshold). If the $U (V)$ texture vector is the base vector, it calls $V (U)$ connected texture region. We call two blocks **U (V) connected** when they are in the same $U (V)$ connected region and connect with each other.

Definition 16.3. The **adjacent connected texture regions**: if there are two blocks belong to two $U (V)$ connected regions, R_1, R_2 , respectively and these blocks are $V(U)$ connected, we say that R_1 and R_2 are adjacent.

The new algorithm can be described as several step.

The first step of the algorithm is to divide the target area into several texture blocks. The second step is texture morphing. After that, a texture connection step is necessary to ensure the continuity of the texture. When the texture is mapped, an adjustment to the depth of field is used to achieve more realistic feeling.

16.3.2 Texture Morphing

Texture morphing is to recalculate the texture coordinates of the frame vertexes of a texture block to satisfy the following condition: for each two pixels in the block, if they are on a line parallel to the $U (V)$ texture vector, they have the same $y (x)$ texture coordinate. The depth of field should also be considered. According to the optic theory, the near object looks bigger while the object far away looks smaller.

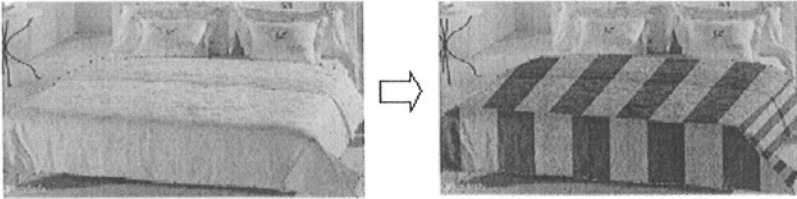
For these reasons the axes projection method is used to calculate texture coordinates. For a pixel A in a texture block, which world coordinate is (x, y) , the angles of the block's U, V texture vectors are θ_u, θ_v , its texture coordinate (tx, ty) is calculated as Eq. (16.1) and (16.2). The result looks as Figure 16.4 shows.

$$ty = \begin{cases} \frac{y - \tan(\theta_u) \times x}{\tan(\theta_v) - \tan(\theta_u)} \times \sqrt{1 + \tan(\theta_v)^2} & \theta_v \in (0, \pi/2) \\ y - \tan(\theta_v) \times x & \theta_v = \pi/2 \\ \frac{\tan(\theta_u) \times x - y}{\tan(\theta_v) - \tan(\theta_u)} \times \sqrt{1 + \tan(\theta_v)^2} & \theta_v \in (\pi/2, \pi) \end{cases} \quad (16.1)$$

$$tx = \begin{cases} \frac{y - \tan(\theta_v) \times x}{\tan(\theta_u) - \tan(\theta_v)} \times \sqrt{1 + \tan(\theta_u)^2} & \theta_u \in [0, \pi/2), \theta_v \neq \pi/2 \\ x \times \sqrt{1 + \tan(\theta_u)^2} & \theta_u = \pi/2 \\ \frac{\tan(\theta_v) \times x - y}{\tan(\theta_u) - \tan(\theta_v)} \times \sqrt{1 + \tan(\theta_u)^2} & \theta_u \in (-\pi/2, 0), \theta_v \neq \pi/2 \end{cases} \quad (16.2)$$

16.3.3 Smoothing Operation

Because of the flexible feature of textile materials, our algorithm can provide smoothing effects for the texture presentation. The texture smoothing operation is necessary for the adjacent blocks when the angle between their non-base texture vectors is larger than a given threshold.



The original image

After texture morphing

Figure 16.4. Texture morphing.

Definition 16.4. As Figure 16.5 shows, AB is the common border between B_1 and B_2 . B and l_1 are the **base dot** and **base line** of AB on B_1 . Here l_1 is parallel with the $V(U)$ vector of B_1 and has a distance value δ from B . The area between l_1

and AB is divided into some polygons, which called the **smoothing polygons** of AB on B_1 .

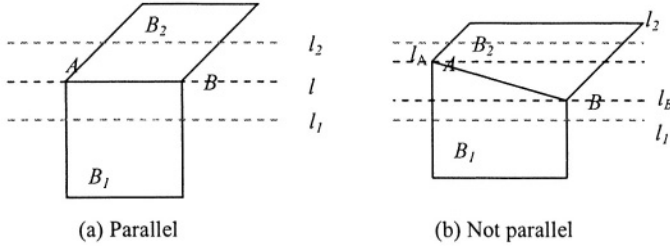


Figure 16.5. Illustration of The base line.

The smoothing polygons are used to smooth the texture effect. For a texture block, if one of its common borders needs to be smoothed, construct smoothing polygons and adjust texture coordinates. The texture image is mapped upon the block, and then the smoothing polygons.

Definition 16.5. As Figure 16.6 shows, A is the base dot, $A_1'(A_2')$ is on the base line of $B_1(B_2)$. $AA_1'(AA_2')$ is parallel with the non-base texture vector of $B_1(B_2)$. $A_1'A_2'$ is the smoothing vector of B_1 and B_2 when the angle between AA_1' and AA_2' is smaller then a given threshold (Figure 16.6 (a)). Otherwise, OA_1' is the smoothing vector of B_1 and $A_2'O$ is the smoothing vector of B_2 (Figure 16.6 (b)). O is the middle point of AM .

Here we use $\angle I$ related to AB to adjust the texture coordinate of the vertexes on smooth polygons, which is calculated as Eq. (16.3), here I is a vertex of smooth polygons.

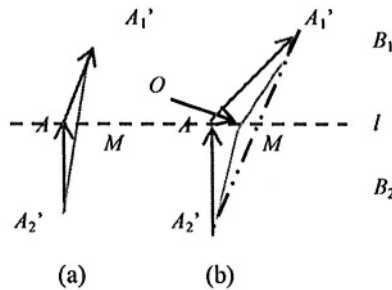


Figure 16.6. Smooth vector.

$$\Delta I = \begin{cases} \delta_A & I = A \\ \delta_B & I = B \\ \delta_{Di} & I \in (D_1, D_2, D_4, D_5) \\ 0 & I \in (M, N) \\ \frac{|y_{D1} - y|}{|y_{D1} - y_A|} \times \delta_A & I \text{ between } D_1 \text{---} A \\ \frac{|y_{D2} - y|}{|y_{D2} - y_B|} \times \delta_B & I \text{ between } B \text{---} D_2 \end{cases} \quad (16.3)$$

Here, y_K is the original texture coordinate of the vertex K , y is the y texture coordinate of I ,

$$\delta_{Di} = \begin{cases} 0 & i = 1, 2 \\ \delta_A & i = 4, 5 \text{ and } y_A = y_B \\ \frac{|y_A - y|}{|y_A - y_B|} \times \delta_B & i = 4, 5 \text{ and } y_A \neq y_B \end{cases} \quad (16.4)$$

δ_A, δ_B are basic adjust values of A, B , which are calculated as Eq. (16.5).

$$\delta = \begin{cases} \left(\cos(\theta_1 - \theta) - \frac{\sin(\theta_1 - \theta)}{\tan(\theta_1' - \theta)} \right) \times |\delta'| & \text{Figure 16.7 (a)} \\ \left(\frac{\sin(\theta_1 - \theta)}{\tan(\theta_1' - \theta)} - \cos(\theta_1 - \theta) \right) \times |\delta'| & \text{Figure 16.7 (b)} \\ \left(\frac{\sin(\theta - \theta_1)}{\tan(\theta - \theta_1')} - \cos(\theta - \theta_1) \right) \times |\delta'| & \text{Figure 16.7 (c)} \\ \left(\cos(\theta - \theta_1) - \frac{\sin(\theta - \theta_1)}{\tan(\theta - \theta_1')} \right) \times |\delta'| & \text{Figure 16.7(d)} \end{cases} \quad (16.5)$$

where $\theta, \theta_1, \theta_1'$ are angles of base texture vector, non-base texture vector and smooth vector respectively. δ' is the difference between texture coordinates of the base dot and dots on the base line. And we have:

$$\delta' = \begin{cases} K_d \times (\max(ty_1) - \min(ty_2)) & \text{if } K_d \times (\max(ty_1) - \min(ty_2)) \leq K_c \times k_y \\ K_c \times k_y & \text{else} \end{cases} \quad (16.6)$$

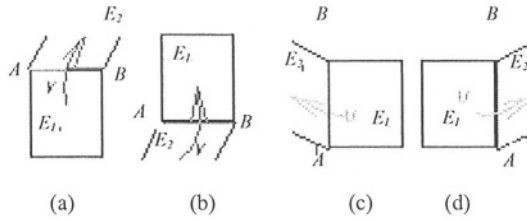


Figure 16.7. Cases for common borders: (a) at the top of the block; (b) at the foot of the block; (c) at left and (d) at right.

It is the projection of δ (as Definition 4 shows) in the texture space.

$$\delta = \delta' / k_y \quad (16.7)$$

where K_c and K_d are constant values, which are 8.0 and 0.05. In Eq. (16.8), ty_1 stands for the y texture coordinate of a vertex of B_1 and ty_2 represents y texture coordinate of a vertex of B_2 . Let M, N are the vertexes which y texture coordinates are the greatest and smallest among the vertexes of the block. Δl is the distance of V texture vector between the lines that are parallel with U texture vector and pass through M and N .

$$k_y = \frac{\max(ty_1) - \min(ty_1)}{\Delta l} \quad (16.8)$$

Then calculate the new texture coordinate as $y' = y + \Delta_1$. For D_1AD_4M , another adjust value Δ_1' is necessary, because the smooth polygon is related to another common border AC . New texture coordinate is $x' = x + \Delta_1'$.

Texture distortion often happens at the area where two surfaces of the target object are connected. Therefore there will be such a case that the common border is not parallel with the base texture vector. Then special processing is required (See Figure 16.5 (b)).

16.3.4 Experimental Results of EasyShow

The EasyShow system presented above is implemented under MS Window2000, VC++ 6.0, PIII800, 128M memory. Some experimental results are showed. In Figure 16.8, simulations of garment are presented. The direction and shape of the texture pattern are fitted to the contorted cloth. In Figure 16.9, a beautiful bed cover is presented on which the texture pattern is natural and smoothed. It should

be noted that in black-white printing, the results will not be as good as with colour printing.

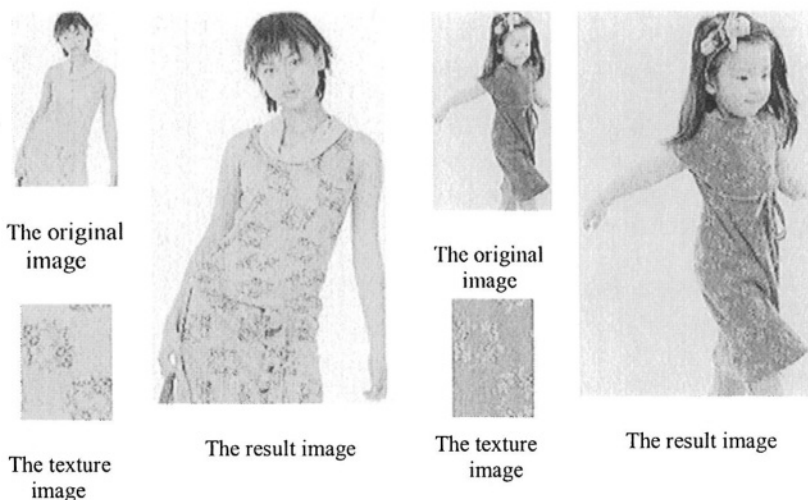


Figure 16.8. Snapshots of EasyShow system.

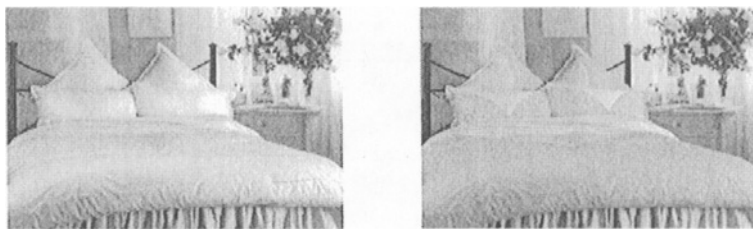


Figure 16.9. Snapshots of EasyShow system (continued).

16.4. Graphics-based Virtual Presentation of Products

Based on the standard of VRML 2.0, ActiveX controller and some other E-commerce related techniques, we implemet our virtual shopping mall EasyMall. EasyMall supports multi-user interaction and manipulation, leverages existing E-commerce solutions through an immersive 3D environment based on VRML and Java.

One of our goals for developing EasyMall is to make it easier for consumers to experience virtual shopping and, as such, to make our virtual store more natural and consistent with the shopper's previous physical experience through providing intelligent guide service. Another goal is to offer a customizable platform to change the attributes of products, such as shape, style, color, size, etc to satisfy consumers' personality requirements.

16.4.1 Overview of EasyMall

In EasyMall, both two kinds of presentation methods mentioned above are used. As to the structure of the mall, we employ the 3D model based presentation method. It could be said that 3D visualized environment can bring a whole new dimension to the way people learn and purchase products online. Thus, consumers can virtually 'interact' with products in EasyMall. Also, in order to let user test some simple manipulation functions of product, we also apply 3D model based presentation method. However, as to the main aim of matching effect, such as the fashion match, the clothing match, etc, in this case, it might be better to provide 2D images instead of or in addition to a 3D model (as it is done in EasyShow system).

The EasyMall consists of four modules for different classes of products, the presentation and customization module for furniture used at home or office [32], the presentation and customization module for ceramic products, which are commonly used in everyday life [33], the presentation and customization module for kitchen product, and the presentation and customization module for electrical apparatus.

The technologies we explored and employed in our EasyMall are described briefly as the following four levels of implementation:

- Basic VRML modeling and behavior;
- VRML communication to Java through Script Node;
- VRML communication to Java applets through EAI;
- JSP to support input, saving, and processing of forms, new world files and file uploads.

In addition, we apply VNET+ to ensure the data share of our Agent guiding system. The EasyMall system which is based on multi-server, employs Blaxxun to realize the interaction between customer and virtual environment. The system includes a web server which maintains customer information, user avatar server

and agent data server. The EasyMall mainly employs VRML for scene representation.

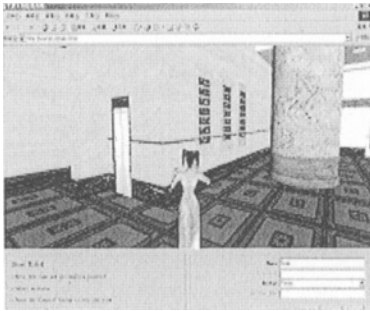


Figure 16.10. Avatar guiding in EasyMall.

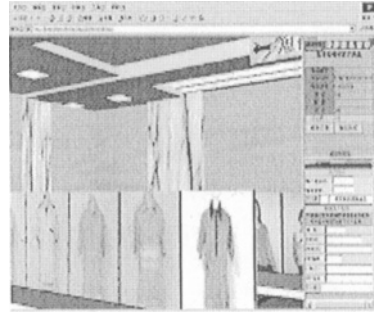


Figure 16.11. Recommended T-shirts.

Figure 16.10 shows that a shopping assistant was guiding the customer enter EasyMall. There is a guiding marker beside the elevator, and the customer may arrive at the corresponding floor on which the category of products is located through pushing the button. Figure 16.11 shows that the customer got a group of T-shirts recommended by the shopping assistant in the apparel presentation hall. The presentation is based on the image (photo) of the shirts. The recommended items are based on the consumer's personal information, the purchase action as well as the interactive communication between the consumer and the seller. Then the customer can inspect them through rotating, zooming and manipulating them one by one through popping up another window. Therefore, they may perform the whole purchase process, such as look into the detailed attribute of products, place orders, decide the paying method, etc. Nevertheless, a consumer may, instead of navigating through the virtual shopping mall, perform an intelligent search of an item based, on parameters such as price, color texture, etc. to save his purchase time.

16.4.2 Example on Virtual Presentation and Customisation

The system also supports the function of virtual presentation and customisation based on 3D models. In the system, client side can recall the template house designed in advance, and the customer can choose or change the material and dimension of the home-used product according to their requirement. This is implemented in the server side by selecting the corresponding texture from four texture libraries and putting it onto the chosen home product.

In this example there are two kinds of navigation modes which are controlled by command buttons and by keyboard. Both modes can attain the same purpose of

navigation. When customer browses the virtual rooms on the web, he can navigate, rotate, browse and purchase as much as he like, which is controlled by buttons on the interface of the system or keyboard according to his own operation habit. For example, navigation mode includes *go ahead*, *return back*, *left forward*, *right forward* etc. Rotation mode includes *turn left* and *turn right* with any angle. Browse mode includes *look closely*, *look up* and *look down*. Thus, customer can get a whole aspect, many-sided information of the product by observing from different angles.

To illustrate the functions of our EasyMall, we describe an example for virtual presentation and customisation of facilities in a virtual kitchen. The customer can choose or change the material and dimension of the home-used product according to their requirements (as shown in Figure 16.12 (b), the original scene is shown in Figure 16.12 (a)).

The customer also can make use of this system to rearrange home-used product in 3D scene by the function of rotating, moving and deleting the model. The system can judge whether it intervenes with entity around according to the constraint and arrange the virtual product logically, as shown in Figure 16.12 (c). Manufacturers and businessmen can publish the virtual scene on web, and customers can browse the 3D effect of home product, navigate and customize in the 3D scene, as shown in Figure 16.12 (d).

It should also be noted that, if the images in Figure 16.12 are printed in black-white other than colour, then the texture replacement effect may not be very noticeable. The reason is that for some colours, the printing in black-white is very similar. This is also the same reason or Figure 16.8 and Figure 16.9.

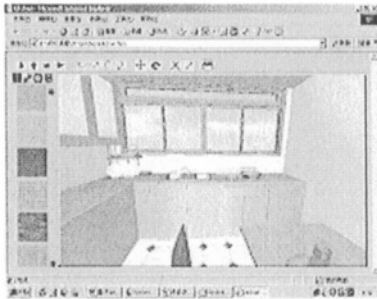
16.5. Summary and Conclusions

In this chapter, we review the research and develop works related to online product presentation and customisation, especially the development of virtual shopping mall. It shows that integrating E-commerce with VR may give the consumer virtual experience and benefit to lead to a bargain. However, there are still some disadvantages using 3D model based product presentation method even though 2D image based presentation method is limited.

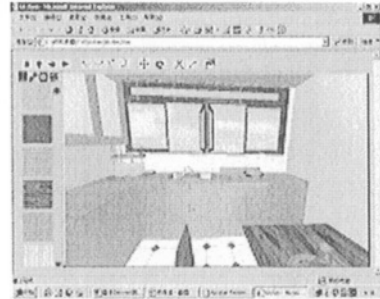
With the Easy-Show for presenting and customizing textile products, the 2D image based method is mainly applied, and In our EasyMall, we employ the mixed presentation method (most are 3D model-based, and some are image-based). A possible future work is to integrate the functions of the EasyShow into EasyMall.

According to [22], there is over 40% of buying attempts failed in the shopping web site embedded with 3D visualization technology. However, we cannot draw a conclusion that some presentation method does not work. There are many research organizations are engaging in the related work to overcome current difficulties.

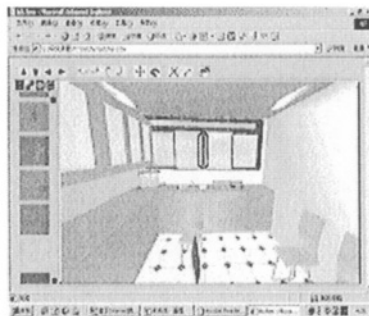
For instance, the company of VR Interactive is setting out with the goal of making the power of VR web presentation more readily accessible to web developers and e-tailors.



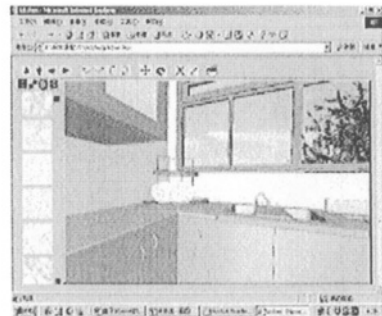
(a) Original scene



(b) Texture of the window frame is changed



(c) Interact with scene
(a table is deleted).



(d) Navigate in the room.

Figure 16.12. Experimental result of EasyMall.

Acknowledgments

This work is supported by National Science Foundation of China (grant no: 600330101), the Education Ministry Excellent Young Teacher Awarding Funds, Excellent Personnel Project of Natural Science Foundation in Zhejiang Province (grant no: RC40008). The authors want to express thanks to these organizations.

Also, the authors would like to express their gratitude to Dr. Fangsheng Wu, Mr. Bin Xu, Mr. Peng Wang, Mr. Xiaocao Wei, Miss Tieying Tang from State Key Lab of CAD&CG at Zhejiang University for their contributions on technical discussions, system implementation, and materials preparation for this chapter.

Authors also would like to thank Prof. David Zhang from The Hong Kong Polytechnic University for invitation on this contribution.

References

- [1] U. Assarsson, T. Moller. "Optimized view frustum culling algorithms for bounding boxes". *Journal of Graphics Tools*, vol. 5, no. 1, pp. 9-22, 2000.
- [2] Arndt Kraft, Stefan Pitsch, Michael Vetter. "Agent-driven online business in Virtual communities". *33rd Hawaii International Conference on System Sciences*, vol. 8, January, 2000.
- [3] Pankaj Kamthan. "E-commerce on the WWW: a matter of trust". http://tech.irt.org/articles/js158/#ec_www.
- [4] D. L. Hoffman, T. P. Novak, and P. Chatterjee. "Commercial scenarios for the Web: opportunities and challenges". *Journal of Computer Mediated Communication*, Special Issue on Electronic Commerce, vol. 1, no. 3, 1995.
- [5] L. Chittaro and R. Ranon. "Virtual reality stores for 1-to-1 commerce". *Proceedings of the CHI2000 Workshop on Designing Interactive Systems for 1-to-1 E-Commerce*, The Hague, The Netherlands, 2000. <http://www.zurich.ibm.com/~mrs/chi2000/>, 2003.
- [6] S. E. Phelan. "Internet marketing: is the emphasis misplaced?" Presented at the *Annual Meeting of the Australian & New Zealand Academy of Management*, Australia, Wollongong, NSW. 1996.
- [7] M. Heim. *Virtual Realism*. New York, Oxford, Oxford University Press, 1998.
- [8] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, et al. "The CAVE: audio visual experience automatic virtual environment". *Communications of the ACM*, vol. 35, no. 6, pp. 65-72, 1992.
- [9] L. G. Lundh. "Introspection, consciousness, and human information processing". *Scandinavian Journal of Psychology*, vol. 20, pp. 223-38, 1979.
- [10] Hairong Li, Daugherty Terry and Biocca Frank. "Characteristics of virtual experience in electronic commerce: a protocol analysis". *Journal of Interactive Marketing*, vol. 15, no. 3, pp. 13-30, 2001.
- [11] The senses considered as perpetual systems. Boston Houghton Mifflin, 1996.
- [12] <http://www.wired.com/wired/archive/2.07/kitchen.html>
- [13] A. Hinkenjann, O. Riedel. "CIA-Tool: a tool for cooperative-interactive planning in virtual environments". *Human Computer Interface (HCI) International*, Yokohama, July 1995. <http://ls7-www.informatik.unidortmund.de/~hinkenja/-publications.html>, 2003.
- [14] J. Dauner, J. Landauer, E. Stimpfig, etc. "3D product presentation online: the virtual design exhibition". *VRML 98*, pp.57-62. ACM Press, Monterey, CA, USA, 1998.
- [15] M. Mine, F. Brooks and C. Sequin. "Moving objects in space: exploiting proprioception in virtual environment interaction". *Proc.SIGGRAPH' 97*, Los Angeles, pp. 19-26,1997.

- [16] Marcus Kaar. "A multi-user environment for the World Wide Web - The virtual shopping mall". PhD thesis, Vienna University of Technology - The Institute of Computer Graphics, October 1998.
- [17] <http://www.vr-shop.iao.fhg.de>, 2003.
- [18] <http://www.activeworlds.com>, 2003.
- [19] S. Han, M. Lim, D. Lee, etc. "Scalable network support for 3D virtual shopping mall". *VSM2002*, September, 2002. Korea. The paper is available at the Web site: <http://vega.icu.ac.kr/~cats/publications.html>, 2003.
- [20] J.C.Oliveira, X.Shen and N.D.Georganas. "Collaborative virtual environment for industrial training and e-commerce". *Proc. Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems, IEEE Globecom'2000 Conference*, Nov.-Dec. 2000, San Francisco, <http://www.discover.uottawa.ca/publications/>, 2003.
- [21] Jayoung Yoon, Gerard J. Kim. "An intergrated VR platform with 2D images and 3D models". Master's Thesis. Dept of CSE, POSTECH, 2002.
- [22] Sangyoon Lee, Tian Chen, Jongseo Kim, Gerard J. Kim, Sungho Han, Zhigeng Pan. "Using virtual reality for evaluating affective properties of product design". Submitted to *JCISE*, 2003.
- [23] Walsh E. A. and M. Bourges-Sevenier. *Core Web 3D*. Prentice Hall, USA, 2001.
- [24] J. Alterbox Neilson. "2D is better than 3D". The paper is available at the Web site: <http://www.useit.com/alertbox/981115.html>.
- [25] M. Hurst. "Why 3D shopping makes no sense-measuring the online customer experience" [online] cited at <http://www.goodexperience.com>, 2003.
- [26] J. Weil. "The synthesis of cloth object". *Computer Graphics*, vol. 20, no. 4, pp. 49-54, 1986
- [27] D. Terzopoulos, J.C. Platt, A.H. Barr. "Elastically deformable models". *Computer Graphics*, vol. 21, no. 4, pp. 205-214, 1987.
- [28] D. Terzopoulos, K. Fleischer. "Deformable models". *The Visual Computer*, vol. 4, no. 6, pp. 306-331, 1988.
- [29] D. Terzopoulos, K. Fleischer. "Modelling inelastic deformation: viscoelasticity, plasticity, fracture". *Computer Graphics*, vol. 22, no. 4, pp. 269-278, 1988.
- [30] D. Roylance, P. Chammas, J. Ting, H. Chi, and B. Scott. "Numerical modelling of fabric impact". *Proceedings of the National Meeting of the American Society of Mechanical Engineers (ASME)*, San Francisco, October 1995.
- [31] Wu Chen, Naiping Xu. "Realistic cloth simulation using energy method". *Journal of Software*, vol. 12, no. 2, pp. 305-308, 2001.
- [32] Jin Wang, Zhigeng Pan, Chengqing Ye. "A natural texture mapping algorithm". *Journal of Computer-aided Design & Computer Graphics*, vol. 12, no. 3, pp.161-165, 2000.
- [33] Fangsheng Wu, Zhigeng Pan, Tian Chen, etc. "Distributed prototype system of furniture's 3D displaying and customization". *Computer Applications*, vol. 23, no. 3, pp. 78-81, 2003.

- [34] Tian Chen, Guofu Yin, Zhigeng Pan. "Interactive design and presentation of ceramic sanitary products". SPIE Proceeding of the Third International Conference on Virtual Reality and Its Application in Industry, Hangzhou, P. R. China., vol. 4756, pp. 259-264, April, 2002

This page intentionally left blank

Chapter 17

3D IMAGING AND APPLICATIONS

Xiaoyi Jiang¹ and Hanspeter Bieri²

1. Department of Computer Science, University of Münster, Germany

2. Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland
xjiang@math.uni-muenster.de, Hanspeter.Bieri@iam.unibe.ch

Abstract Modeling complex objects/environments represents an important task of computer graphics and poses substantial difficulties to traditional synthetic modeling approaches. Today's 3D imaging technologies provide an attractive alternative. In this chapter we give an introduction to 3D imaging. We outline the various steps of the whole modeling process and describe the techniques available. Also, applications are discussed to highlight the usefulness of these technologies.

Keywords: Range sensor, active stereo vision, time-of-flight, triangulation, registration, ICP, reconstruction, α -shapes, power crust, radial basis function, view planning, geometry processing

17.1. Introduction

Traditionally, computer graphics and computer vision have been considered to be at opposite ends of the spectrum. While the graphics field considers the visualization of models, the vision field involves the processing of sensor data for the purpose of image understanding. From today's point of view, however, this strict dichotomy is becoming less accurate. Computer graphics and computer vision are coming closer and their interaction starts to produce interesting results. 3D imaging represents such an example.

The computational power of current even inexpensive personal computers enables to display complex simulations of the reality. This facility unavoidably leads to the need of modeling realistic contents, ranging from complex real-world objects to entire environments, which is beyond the level achievable by the synthetic modeling approaches of computer graphics. At this place, 3D imaging techniques provide an alternative and have already demonstrated their potential in several application fields.

The dream 3D imaging equipment would allow one to place it in front of the object/environment of interest, press a key, and have a complete digital 3D description of that object/environment in short time. The reality is that the model building process consists of a number of distinct steps, some of which may not be fully automatic and need human interaction. The modeling of large-scale complex objects/environments is still faced with time-consuming data acquisition and non-trivial processing of the acquired data.

The main steps of the model building process are:

- 3D sensor calibration: Determination of internal and external parameters of the 3D sensor in use.
- Data acquisition: The object/environment of interest is imaged. Typically, a number of 3D images are acquired at different viewpoints in order to capture the complete geometry.
- Data fusion: The different views are merged within a single coordinate frame.
- Model generation: A model representation is generated from the fused 3D data.

This chapter gives an introduction to the involved techniques and algorithms. We start with a description of 3D imaging principles, followed by a short review of 3D sensors on the market and in research labs. Then, data fusion, model generation, and some other related issues are discussed. Finally, we highlight several applications of 3D imaging to show the success of this modeling technique already achieved in various fields and its huge potential for future applications.

17.2. 3D Imaging Principles

A *range sensor* is any device that senses 3D positions on an object's surface. The sensing output may be either *structured* or *unstructured*. In a structured manner a *range image* contains a grid of 3D points on the sensed surface, expressed typically in either Cartesian coordinates or cylindrical coordinates. On the other hand, unstructured range data give us a collection (cloud) of 3D points on the surface. The main difference between these two sensor output forms is that a range image explicitly provides the connectivity of the points while this information is (at least initially) unknown in a point cloud. The connectivity information is essential for topology reconstruction of the sensed surfaces.

Of limited use are touch-based range sensors which allow a pointwise manual probing of objects. The output is a point cloud. In this case the data acquisition is extremely painstaking and thus only suitable for simple objects.

Image-based ranging techniques make use of a wide spectrum of principles which can be categorized by means of several criterion. The most popular categorization is that into *passive* and *active* methods. Passive ranging totally relies on ambient lighting conditions and does not impose any artificial energy sources upon the environment. These methodologies usually have strong correlates with human visual functionality. Examples are stereo vision and the class of shape-from-X techniques (shading, focus, texture, contour, motion, etc.). The book [22] is a good source for this category of approaches. Despite of the recent advances in this area, in particular stereo vision, passive sensors have not been widely used for 3D modeling due to accuracy reason and some other technical limitations. Stereo vision, for instance, requires sufficient texture details on the objects and have often trouble with statues, industrial parts, building, etc. Most shape-from-X approaches are currently still not mature enough to be applied for complex real-world tasks.

By contrast, active methods impose energy sources (light, near light, ultrasonic, etc.). The controlled nature of these artificial energy sources alleviates the main problems encountered in passive approaches. As a result, there exists today, besides various active sensors built in research labs, a vivid commercial market for active sensors, ranging from relatively low-cost to high-end products. These sensors are also those mostly used in 3D modeling tasks. In the rest of this section we briefly outline three most important active ranging principles: active stereo vision, time-of-flight, and triangulation. In the next section concrete active sensors based on these principles are discussed.

17.2.1 Active Stereo Vision

The main difficulties with (passive) stereo vision are lacking texture details and the correspondence analysis. The first problem can be alleviated by artificially projecting some pattern onto the scene. The pattern should be designed to deliver sufficient and well distinguishable features for the correspondence analysis. Figure 17.1 (from [24]) shows a cat statue imaged this way. Due to the homogeneous surfaces passive stereo approaches encounter many ambiguities while the active stereo is able to resolve them by projecting a color pattern. Note that given sufficient texture details the correspondence analysis remains to be solved.

Actually, the correspondence analysis can be made trivial by local pattern projection. In the extreme case only one light beam is projected to highlight a single point on the object surface. The highlighted point provides a visible feature for the stereo cameras and no correspondence analysis is necessary. A two-dimensional sweeping of the light beam then samples the entire scene. This simple approach can be easily extended to projecting a light plane for capturing

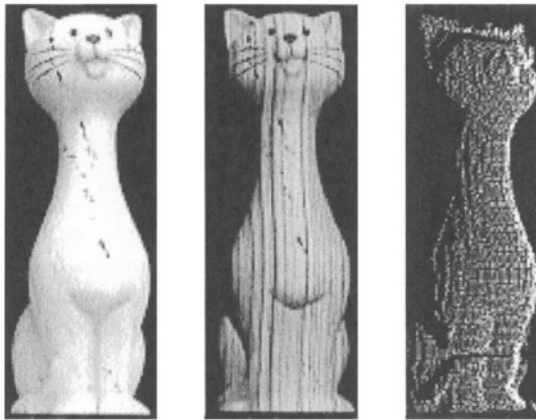


Figure 17.1. Active stereo: a cat statue (left), projection of a color pattern (middle), and the resulting point cloud.

much more 3D point a time. In this case only a one-dimensional sweeping of the light plane is needed.

17.2.2 Time-of-flight

Time-of-flight (TOF) is based on the same principle as radar. Any TOF sensor contains an emitter which sends out light signals and a receiver which receives the signal reflected from the targeted object surface. The distance is measured by determining the time t from sending to receiving the light signals. A scanning mechanism is needed to capture the entire scene.

Dependent of how to measure the time t there exist two basic types of TOF sensors. With *pulse modulation* techniques, pulsed light signals are sent out and t is measured directly. Then, the distance from source to target is computed by $ct/2$, where c is the signal transmission speed. In *continuous wave modulation*, by contrast, the light signal is amplitude-modulated. It is now the phase difference between the transmitted signal and the received signal which is measured and related to the distance. An alternative here is to use frequency-modulation instead of amplitude-modulation.

TOF sensors need sophisticated instrumentation and as a result, they tend to be expensive and belong to the high-end segment of commercial products.

17.2.3 Active Triangulation

Triangulation is certainly the oldest method for measuring range to remote targets and is also today the most common one. The typical setup of an active triangulation sensor consists of a camera and a light projector. The various

subclasses are categorized by means of the sort of light projection: light beam, light plane, and structured light.

Light Beam. Here a single light beam is projected onto the scene and the highlighted object point is observed in the camera image. Obviously, the highlighted object point must be on the light beam itself. In addition it must also be on the line defined by its observed image position and the projection center of the camera. The intersection of these two lines gives the 3D position. Note that it is the task of calibration to determine the camera and light beam geometry which are the fundamental of the 3D position computation. To capture the entire scene a scanning mechanism is needed to generate a large number of light beams.

Light Plane. Like in active stereo vision, the light beam approach can be easily extended to light plane projection for higher data acquisition speed. In this case a light plane is projected onto the scene to produce a highlighted stripe. Then, the 3D position of each point on the highlighted stripe is determined by intersecting the light plane with the line defined by the projection center of the camera and the image position of that point. To capture the object one may sweep the light line. Alternatively, the object may be placed on a turntable for translational or rotational movements.

Structured Light. The main drawback of light plane projection is the lacking efficiency of data acquisition. Structured light approaches intend to reduce the number of projections, in the best case to a single projection. The most popular technique of this class is that of binary-encoded pattern. Rather than project n light planes and process n images it is possible to generate n light planes by projecting only $n \log_2 n$ 2D patterns. In all structured light methods the most crucial issue is the design of the projection pattern(s) such that for each image pixel there is a way to know the identity (i.e. geometry) of the light source highlighting the corresponding 3D point on the object surface and the triangulation can be done.

Numerous variants of structured light methods have been proposed in the literature, see [2] for a review. They are all fast and relatively low-cost devices. Except the binary-encoded light technique, however, most of them are still in the stage of research lab solutions and not commercially available yet.

17.3. 3D Sensors

Although the ranging principles are generally quite simple, several design issues (optics, electronics) have to be carefully considered in realizing a concrete

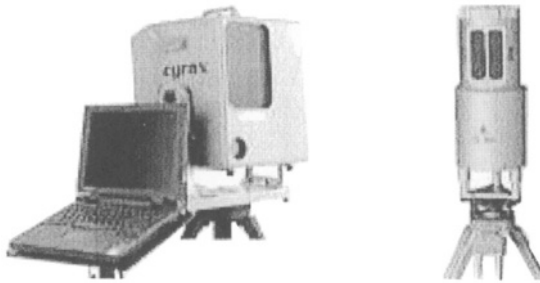


Figure 17.2. Cyra 2500 (left) and LMS-Z420i (right).

range sensor. As an example, Pretov *et al.* [29] reports the technical details in designing a triangular sensor based on laser plane projection.

Today there exist a vast number of commercial and non-commercial 3D sensors. It is not the intention of the authors to give a market survey. Instead, we briefly discuss some popular sensors and point interested authors to sources for more detailed information. Note that the inclusion of commercial sensors should not be interpreted in any way as an endorsement of a vendor's product.

17.3.1 TOF Sensors

Cyra Cyra 2500 scanner (www.cyra.com, see Fig. 17.2) is suited for capturing large "as-built" structures and turning them into CAD representations. They have been used in applications from retrofitting power plants to digitizing movie sets. The approximate system pricing is \$150K and renders it, as for most TOF sensors, to the high-end segment of range sensors. Riegl (www.riegl.com) manufactures a series of TOF sensors, see Fig. 17.2 for the model LMS-Z420i

17.3.2 Active Triangulation Sensors

Several single-light-plane triangulation sensors are available on the market. Typically, they project a single laser plane onto the scene and have no mirrors or other equipments for sweeping the light plane. A scan therefore can only capture a single stripe on the object surface. Because of the simple design, these sensors are compact and called *hand-held* range sensors. In this case a human operator moves the sensor back and forth across surface in an action similar to paint spraying. A hand-held sensor is usually attached to a special robot-arm-like localizer to measure its position and orientation at each time of movement. The ModelMaker (www.3dscanners.com, see Fig. 17.3) and FastSCAN Corba (www.polhemus.com/fastscan.htm) are two representatives of hand-held range sensors. Faro (www.faro.com) produces a series of high-accuracy localizers, see Fig. 17.3 for one model out of this series. Note that hand-held range

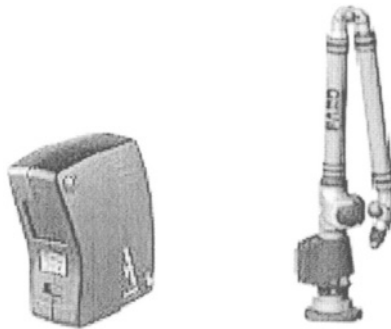


Figure 17.3. ModelMaker sensor (left) and high-precision localizer from Faro (right).

sensors provide a point cloud (unstructured range data) instead of structured range images.

The most popular commercial range sensors are based on active triangulation using laser light plane with some form of light plane sweeping. Minolta developed the Vivid series (www.minolta3d.com, see Vivid 910 in Fig. 17.4) of such sensors, each being a stand-alone unit with all functions built in. They use a galvanometric motor to scan the laser plane across the scene. The most well-known range sensors of this class are those from Cyberware (www.cyberware.com). The company manufactures a series of sensors which capture objects ranging from apple-size models to full-size human bodies, see Fig. 17.4.

The company ABW in Germany (www.abw-3d.de/home_e.html) delivers the LCD-series of projectors for the binary-encoded structured light approach. In combination with a camera and related software one yields a relatively inexpensive range structured light range sensor. Since the two components projector and camera are not built into a unit, the customer has the freedom to configure their relative positions.

Probably the most sophisticated triangulation sensors were developed at the National Research Council of Canada [17] (www.vit.iit.nrc.ca/VIT.html). Built upon both light-beam and light-plane triangulation principles, careful instrumentation resulted in a series of prototypes, each targeting a different class of applications or operational conditions (e.g. high tolerance to ambient illumination, especially sunlight).

17.3.3 Low-cost Range Sensors

The wide range of commercial range sensors have a strong drawback of high costs, typically starting from several thousands of US\$. This leaves them inaccessible to small businesses and domestic users. In addition, the high

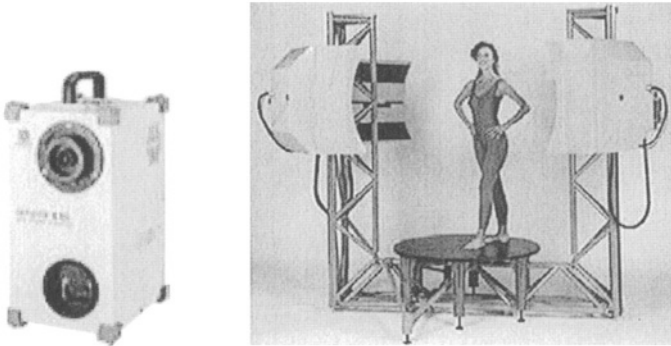


Figure 17.4. Light-plane triangulation range sensors: Minolta Vivid 910 (left) and Cyberware's whole body 3D scanner bundle including four WB4 digitizing heads and WB4 motion system (right).

accuracy claimed for many expensive sensors is not necessary for a variety of applications. Recently, several researchers have studied low-cost range sensors to meet the needs of this segment of users.

In [8] the authors describe their implementation of active stereo vision using a single light beam. The light source is a laser pointer held in the hand of a human operator, making the sensor a flexible and portable one.

Takatsuka *et al.* [38] proposes an interactive version of active triangulation by light beam. The light beam is realized by a laser pointer held in the hand of a human operator. Thus, no costly light beam projection and sweeping mechanism are needed. Three LCDs are attached to the laser pointer and observed in the camera image. The image positions of these LCDs uniquely determine the location of the light beam in space which is necessary for triangulation.

Two low-cost single-light-plane triangulation techniques are described in [9, 16]. No expensive lighting and sweeping mechanism are required. Instead, a desk-lamp or the sun serves as the light source. A human operator sweeps a stick between the light source and the scene, casting a series of shadows into the scene. The 3D shape is extracted from the spatial and temporal location of the observed shadows.

All these techniques have in common that a human operator is involved to interactively move the light source. This manner usually results in an output of unstructured range data. While this makes the topology recovery more difficult than in the case of range images, it enables a variable sampling of surfaces. Highly structured surfaces can be easily controlled to receive a higher sampling rate than simple surfaces. In this case the human knowledge helps establish a reasonable locally variable sampling rate that otherwise must be achieved by postprocessing uniformly sampled surfaces.

17.3.4 Discussion

At www.geomagic.com/support/resources/3scannersA.pdf a list dated October 2002 is available that includes the accuracy, volume, speed, etc. for quite a large number of the most popular range sensors. Table 17.1 contains links to 3D imaging hardware and software manufacturers. Detailed description of commercial range sensors can be found in [5, 7].

Many range sensors are able to provide color images of the sensed scene which are in registration with the range images. The availability of these data allows us to perform texture mapping for realistic rendering of the generated models.

The sensor choice for a particular application depends on a number of factors: object size, surface properties, price, accuracy, speed, portability, operational conditions, etc. With respect to object size, for instance, large objects/environments are best scanned by TOF sensors. At the other end of the spectrum, touch-based sensors may be well suited for small and simple objects. Also, some sensors pose constraints on the surface properties. Most of the sensors require the surfaces be opaque and only touch-base sensors work safely with transparent objects. Finally, the operational conditions may play a central role for out-door modeling tasks.

Table 17.1. Useful information sources.

www.simple3d.com
perso.club-internet.fr/dpo/numerisation3d/
www.docjava.com/scanners/3dscanne.htm
List of 3D range sensor and range data processing software manufacturers
homepages.picknowl.com.au/myers/surface/resources.htm
List of 3D range sensor and range data processing software manufacturers
Lots of related academic stuffs (research groups, conferences, code)
www.3dlinks.com/hardware_scanners.cfm
A great source of information on 3D computer graphics and contains a section on range sensors.
www.geomagic.com/support/resources/
Most vendors of 3D range data processing software provide links to sensor manufacturers. This is one of the numerous software vendors.
www.sculptor.org/3D/
Website of 3D technologies and services for sculptors and contains a sensor section

17.4. Data Fusion

Usually, a number of 3D scannings are acquired at different viewpoints in order to capture the complete geometry. Each view is represented in an in-

¹ All links have been checked at the time of writing (August 2003).

dependent sensor coordinate system. It is the task of data fusion to integrate the multiple views to a single representation of the sensed object/environment. This task is done in two steps:

- Registration: transform the views into a common coordinate frame.
- Integration: merge the multiple registered views to form a single representation.

Due to today's hardware support the output of the data fusion process is typically a triangular mesh.

17.4.1 Registration

Assume that the data acquisition step produces a sequence $V_1, V_2, \dots, V_m$ of views at different viewpoints, each in its own local coordinate frame. The goal of registration is to determine a set of transformations $T_1, T_2, \dots, T_m$ that transform the views to a common coordinate system and minimizes an error function. Here T_i , $i = 1, \dots, m$, is a rigid transformation and consists of a rotation and a translation. The fundamental of registration is the fact that the views possess overlapping parts. The registration aligns the views such that these overlapping parts coincide. The error function measures the degree of the coincidence.

Registration has received substantial attention in the computer vision community. Generally, a first *coarse* registration is performed to determine the inter-view transformation with a few degrees of rotation and a similarly small translational tolerance. Then, a *fine* registration follows in which the transformation parameters are fine-tuned.

Coarse Registration. Many sensors are equipped with a turntable upon which the object is placed for sensing. In this case the coarse registration is directly provided by the system. Note that a fine registration is still needed to compensate for mechanical variability in the turntable mechanism.

The coarse registration is mostly done manually. For instance, a human operator may mark some corresponding points on the views. Although principally simple, this approach can become a tedious job when many views are to be fused and/or when the surfaces are smooth so that no clear features can be easily matched even by the operator.

Several automatic coarse registration methods have been proposed, see [42] and the literature review therein. Typically, these methods detect features and conduct a matching of the features in the different views. An obvious advantage of automatic coarse registration is the elimination of manual intervention. This speedup also allows to build more accurate models because it becomes more feasible to integrate more views. This way the risk of missing data becomes

smaller and by having more measures of the same location a more accurate model can be obtained.

Fine Registration. The iterative closest point (ICP) algorithm developed by Besl and McKay [6] serves as the basis for a large number of fine registration algorithms of two views, see Fig. 17.5. In its original form ICP requires that *every* point in one view have a corresponding point in the other view. To work with overlapping range data it must be extended to handle outliers [40, 44]. For this purpose some heuristics are introduced to decide whether a pair of closest points (p_1, p_2) should be excluded from the new transformation estimation. Several variants of ICP have been suggested in the literature. Rusinkiewicz and Levoy [35] have evaluated these algorithms focusing on computation efficiency in search of real-time performance. The recent study [13] investigates the robustness issue of fine registration.

The transformation parameter space explored by ICP and its variants may contain many local optima. For a convergence to the global optimum two views P_1 and P_2 must be reasonably aligned at the beginning. This is reason for coarse registration. Also, various nonlinear optimization algorithms such similar annealing and evolutionary programming [32] may help climb out of local optima.

In the case of multiple views one may register them incrementally or simultaneously. The incremental approach is efficient and required less memory, but the registration error can accumulate and redundant data are not fully used to improve registration. By contrast, the simultaneous methods are more expensive in space and time. But at least theoretically, they yield more accurate registration results.

Input: Point set P_1 and P_2 (already coarsely registered)

Output: Fine registration of P_1 and P_2

```

while (registration error > threshold) {
    for (every point  $p_1 \in P_1$ ) find its closest point  $p_2 \in P_2$ ;
    Determine the transformation  $T$  from  $P_1$  to  $P_2$ 
        using all correspondences  $(p_1, p_2)$ ;
    Transform  $P_1$  by  $T$ ;
}

```

Figure 17.5. ICP algorithm.

17.4.2 View Integration

After the views have been registered and transformed into a common coordinate frame, it remains to integrate them to a single representation. The approaches for this purpose depend on the nature of 3D data: structured range images or unstructured point cloud.

Structured Range Images. For this kind of data the two major approaches are *mesh-based* and *volume-based*, exemplified by the classical work [40,12] (Both papers and their corresponding software implementation can be downloaded from www-graphics.stanford.edu/data/3Dscanrep/). In [40] a triangular mesh is constructed for each range image based on the pixel connectivity information. Then, two meshes are stitched together by removing redundant triangles in overlapping parts to form the topology of the final mesh. The geometry of the final mesh is further improved by bringing the removed triangles back into a consensus geometry process. This way the redundant data in the overlapping parts are fully used for increased reconstruction accuracy.

Volume-based view integration [12] constructs an intermediate implicit surface which is an iso-surface of a spatial field function $f(x, y, z) = \text{constant}$. If the field function is defined as the distance to the nearest point on the object surface, for instance, then the implicit surface is represented by all points where the field function is zero, i.e. $f(x, y, z) = 0$. This intermediate representation allows a straightforward combination of views. Finally, the explicit surface is reconstructed by the Marching Cubes algorithm for implicit surface polygonization.

Unstructured Range Data. The difficult task here is not the integration but the topology/geometry reconstruction of the integrated point cloud. The work by Hoppe *et al.* [19] constructs, like in volume-based integration of range images, an implicit surface representation. The Marching Cube algorithm is applied to extract the iso-surface $f(x, y, z) = 0$ of the implicit surface function. Another recent work is given in [27].

In the computer graphics community there are several approaches to constructing a representation of mesh or some other form for an (integrated) point cloud. We briefly describe three of them: α -shapes, power crust, and radial basis functions. 3D α -shapes were introduced by Edelsbrunner and Mücke [14]. Given a set P of points one can build its (3D) Delaunay triangulation

²In [40] the mesh is actually constructed before registration and used for improving the accuracy of closest point search in ICP.

³The Delaunay triangulation is a tetrahedrization of the convex hull of P , i.e. a partition into tetrahedra, in such a way that the interior circumscribing sphere of each tetrahedron τ does not contain any other point of P than the vertices of τ . Under non degeneracy assumptions the Delaunay triangulation is unique.

$\mathcal{T}$, which is regarded as a simplicial complex. Each simplex $\sigma \in \mathcal{T}$ (vertex, edge, triangle, and tetrahedron) is assigned a size. Let S_σ be the smallest ball whose boundary contains all vertices of σ . Then the size of σ is defined to be the square of the radius of S_σ . One can intuitively think of an α -shape as the subcomplex obtained in the following way: Imagine that a ball-shaped eraser with radius $\sqrt{\alpha}$ is moved in the space, assuming all possible positions such that no point of P lies in the eraser. The eraser removes all simplices it can pass through, but not those whose size is smaller than α . The remaining simplices (together with all their faces) form the α -shape for that value of the parameter α . Two extreme cases are the 0-shape which reduces to P , and the ∞ -shape, which coincides with the convex hull of P . Note that there exist only a finite number of different α -shapes, and it is the user who decides for a particular α value. Fig. 17.6 (from [24]) shows five α -shapes of a teapot. In some sense the power crust approach [1] is opposite to α -shapes. Instead of deletion, insertion

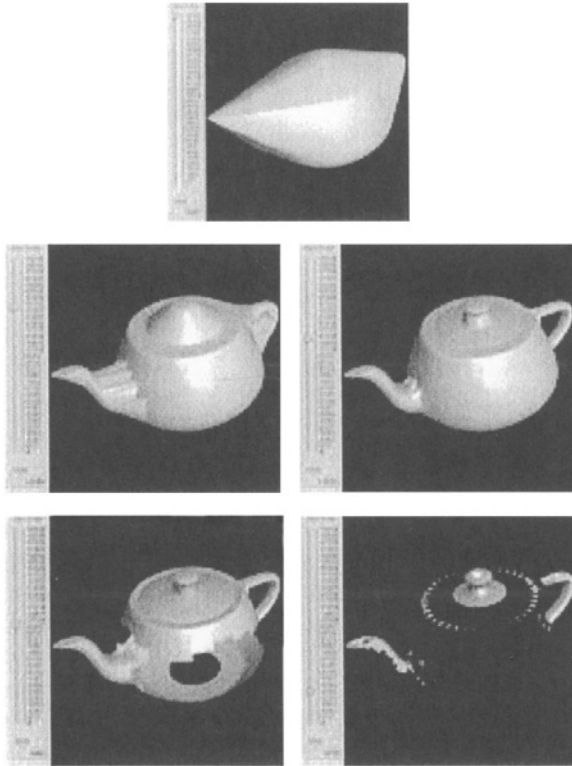


Figure 17.6. α -shapes of a teapot with decreasing α -values. The convex hull is on the top.

operation is performed there to an initial structure defined by the medial axis of P .

Representation with Radial Basis Functions (RBFs) [11] is quite different from both α -shapes and power crust. This approach represents a point set P by a single RBF which is defined as:

$$s(x) = p(x) + \sum_{i=1}^n \lambda_i \phi(|x - x_i|)$$

where p is a low-order polynomial, ϕ is a real valued function on $[0, \infty)$ of simple form, and $P = \{x_1, x_2, \dots, x_n\}$. A RBF can be fitted to P . In this case the object surface is defined implicitly as the zero set of the RBF and can be visualized directly using an implicit ray-tracer. A mesh representation of the modeled surface can be achieved by the Marching Cubes algorithm for implicit surface polygonization.

17.5. Model Generation

A variety of model representations are known from the literature, including parametric forms, algebraic implicit functions, superquadrics, generalized cylinders, and polygonal meshes [10]. Polygonal meshes have been a favorite representation in computer graphics. Most range data fusion methods deliver their final results in this format. If required, other representations can be generated from a mesh of the sensed objects.

17.6. Other Related Topics

So far we have concentrated on the main issues of the 3D modeling process. There are several other relevant topics that are worth mentioning.

17.6.1 View Planning

Scanning a complex object may be tedious work due to the large number of views required to recover the complete geometry. While this process is typically controlled by the human operator, there is a need to devise scanning strategies that minimize the number of views. This will obviously reduce the overhead for both data acquisition and data fusion.

If information regarding the geometry of the object was known, then the problem of view planning would be greatly simplified. In the general case, however, this geometry is exactly what we want to capture and therefore the next (best) view to be scanned must be decided solely on the basis of the range data acquired at previous viewpoints. A survey on the closely related problem

of sensor planning can be found in [39]. Recent work on view planning are in [30, 31].

17.6.2 Geometry Processing

Due to their simplicity and direct hardware support (triangular) meshes are a popular representation of sensed objects/environments. However, they are not compact representations. It is not unusual to have millions and even billions of triangles. Since the rendering speed directly depends on the mesh size, *mesh optimization* strategies have been studied to reduce the number of triangles, while still maintaining a good approximation to the surfaces. The fundamental of mesh optimization is that the neighboring vertices in a good-quality mesh should be geodesically the same distance apart [41]. For highly curved areas, two vertices may be close geometrically, but very far apart geodesically. In these areas, the local mesh representation should be fine. By contrast, in (nearly) planar regions the mesh density can be coarse. An important issue in mesh optimization is to avoid simplifications that yield topological changes, e.g. creation of holes and surfaces smoothed to lines. A large number of mesh optimization methods are known in the literature, see [10] for a detailed discussion.

Several other mesh operations are of interest, including mesh denoising, editing, and compression. The SIGGRAPH Courses Notes on Digital Geometry Processing [37] are a good source for these operations. The book [15] provides fundamental theoretical considerations of meshes.

A further mesh operation is that of watermarking. With the rapid increase in 3D imaging sensors and processing tools, companies and copyright owners of 3D data who sell or present their creations in the virtual space will start to face copyright-related problems. Watermarking techniques may provide a means for copyright protection. While a huge number of watermarking approaches have been published for media types like still images, videos, and audio streams, there are still very few works on 3D watermarking. A recent work [3] addresses the fundamentals of geometry-based watermarking and presents an algorithm that modifies normal distributions to invisibly store information in the model's geometry.

A particularly interesting geometry operation in frequency space is recently introduced in [28]. The authors extend standard Fourier techniques to the domain of unstructured sets of points. The basic idea is to preprocess a point cloud into a model representation that describes the object surface with a set of regularly resampled height fields. These surface patches form "windows"

⁴In computer vision the criteria for a good object representation may be totally different. For instance, Johnson and Hebert [21] require that the model be represented in polygonal mesh format with the mesh density as regular as possible.

in which the discrete Fourier transform is performed to obtain a set of local frequency spectra. The concept of frequency on unstructured data gives us access to the vast space of sophisticated spectral methods resulting from decades of research in signal processing.

17.7. Applications

Applications of 3D imaging have appeared very quickly. Aside from the traditional areas of reverse engineering and model digitizing for movie studios the new generation of sensors prove useful for digital content creation. The ability to efficiently digitize objects, edit them, and put them on the Web opens new dimensions in Web design. Due to recent advances in both hardware and software it does not seem to be unrealistic that 3D imaging will eventually spread into consumer markets.

The most spectacular story in 3D imaging is probably the large-scale Digital Michelangelo Project [23], in which seven of Michelangelo's sculptures were modeled, as well as other artifacts including the parts of the *Forma Urbis Romae*. Actually, heritage conservation has gained much interest in the last years and several projects have been initiated. A group from IBM used a combination of structured light and photometric stereo to model Michelangelo's *Florentine Pietà* [4]. Researchers at University of Tokyo built shape models of the 15-meter bronze Great Buddha in Kamakura [25]. At the Museum of the Terra Cotta Warriors and Horses, Xi'an, China, a team of archaeologists, computer scientists, and museum staff artists applied 3D imaging techniques to recover excavated relics [45]. At the National Research Council of Canada several projects have been conducted in the area of heritage conservation and documentation using their own range sensors [17].

Despite of its relatively short history, 3D imaging has already experienced numerous success stories. Screening the academic literature and the websites of range sensor vendors reveals a wide range of application fields benefiting from this technology. Besides heritage conservation other areas include virtual reality, medicine, forensic recording, and 3D movies. The steady advances in both sensor technologies and range data processing methodologies will continue to widen the application spectrum in future. It is believed that dense range data will change our view of 3D computer graphics [26].

17.8. Conclusions

In this chapter we have given a brief introduction to 3D imaging techniques. More detailed description of ranging principles can be found in [20, 36]. The two survey papers [10, 18] and the special issue [33] contain a broad cover-

age of range data fusion techniques. Further details of this research area can be found in all conference proceedings and journals in computer graphics and computer vision community. In particular, the recent special issues [34, 43], the series of International Conference on 3-D Digital Imaging and Modeling (www.3dimconference.org/) and Internal Conference on Optical 3-D Measurement Techniques, and the recent International Symposium on 3D Data Processing, Visualization, and Transmission (www.dei.unipd.it/conferences/3DPVT/) are good sources of up-to-date information.

Although we are still far away from the dream 3D imaging equipment, the current techniques allow us to tackle challenging tasks that were not possible before. As a matter of fact, we have seen many applications in various areas and the spectrum is continuing to grow. The 3D nature of our world leaves a field of unpredictable dimension for future exploration of 3D imaging technologies.

References

- [1] N. Amenta, S. Choi, and R. Kolluri. "The power crust". *Proc. of 6th ACM Symposium on Solid Modeling*, pp. 249–260, 2001.
- [2] J. Batlle, E. Mouaddib, and J. Salvi. "Recent progress in coded structured light as a technique to solve the correspondence problem: a survey". *Pattern Recognition*, vol. 31, no. 7, pp. 963–982, 1998.
- [3] O. Benedens. "Geometry-Based Watermarking of 3D Models". *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 46–55, 1999.
- [4] F. Bernardini *et al.* "Building a Digital Model of Michelangelo's *Florentine Pietà*". *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 59–67, 2002.
- [5] P. Besl. "Active, optical range imaging sensors". *Machine Vision and Applications*, vol. 1, no. 2, pp. 127–152, 1988.
- [6] P. Besl and N. D. McKay. "A method for registration of 3-D shapes". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
- [7] F. Blais. "A review of 20 years of ranges sensor development". *Videometrics VII, Proc. of SPIE-IS&T Electronic Imaging, SPIE*, Vol. 5013, pp. 62–76, 2003.
- [8] N. A. Borghese *et al.* "Autoscan: a flexible and portable 3D scanner". *IEEE Computer Graphics and Applications*, vol. 18, no. 3, pp. 38–41, 1998.
- [9] J.-Y. Bouguet and P. Perona. "3D photography using shadows in dual-space geometry". *Int. Journal of Computer Vision*, vol. 35, no. 2, pp. 129–149, 1999.
- [10] R. J. Campbell and P. J. Flynn. "A survey of free-form object representation and recognition techniques". *Computer Vision and Image Understanding*, vol. 81, no. 2, pp. 166–210, 2001.
- [11] J. C. Carr *et al.* "Reconstruction and representation of 3D objects with radial basis functions". *Proc. of SIGGRAPH'2002*, pp. 67–76, 2002.
- [12] B. Curless and M. Levoy. "A volumetric method for building complex models from range images". *Proc. of SIGGRAPH'96*, pp. 303–312, 1996.

- [13] G. Dalley and P. Flynn. "Pair-wise range image registration: a study in outlier classification". *Computer Vision and Image Understanding*, vol. 81, no. 1-3, pp. 104–115, 2002.
- [14] H. Edelsbrunner and E. P. Mücke. "Three-dimensional alpha shapes". *ACM Trans. on Graphics*, vol. 13, no. 1, pp. 43–72, 1994.
- [15] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.
- [16] R. B. Fisher *et al.* "A low-cost range finder using a visually located, structured light source". *Proc. of 2nd Int. Conf. on 3-D Digital Imaging and Modeling*, pp. 24–33, 1999.
- [17] G. Godin *et al.* "Active Optical 3D Imaging for Heritage Applications". *IEEE Computer Graphics and Applications*, vol. 22, no. 5, pp. 24–36, 2002.
- [18] A. A. Goshtasby. "3-D model construction from multiview range images: survey with new results". *Pattern Recognition*, vol. 31, no. 11, pp. 1705–1714, 1998.
- [19] H. Hoppe *et al.* "Surface reconstruction from unorganized points". *Proc. of SIGGRAPH'92*, pp. 71–78, 1992.
- [20] R. Jarvis. "Range sensing for computer vision". In *Three-Dimensional Object Recognition Systems*, A. K. Jain and P. J. Flynn (Eds.), pp. 17–56, Elsevier, 1993.
- [21] A. E. Johnson and M. Hebert. "Control of polygonal mesh resolution for 3-D computer vision". *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 261–285, 1998.
- [22] R. Klette, K. Schlüns, and A. Koschan. *Computer Vision: Three-Dimensional Data from Images*. Springer-Verlag, 1998.
- [23] M. Levoy *et al.* "The digital michelangelo project: 3D scanning of large statues". *Proc. of SIGGRAPH'2000*, pp. 131–144, 2000.
- [24] M. V. Löffel. *A Java Framework for Point Sampling and Surface Reconstruction*. Master's Thesis, Univ. of Bern, 2002.
- [25] D. Miyazaki *et al.* "The great buddha project: modeling culture heritage through observation". *Proc. of 6th Int. Conf. Virtual Systems and Multi-Media (VSMM2000)*, pp. 138–145, 2000.
- [26] L. Nyland *et al.* "The impact of dense range data on computer graphics". *Proc. of IEEE Workshop on Multi-View Modeling & Analysis of Visual Scenes*, pp. 3–10, 1999.
- [27] C. Obonsek and N. Guid. "A fast surface-based procedure for object reconstruction from 3D scattered points". *Computer Vision and Image Understanding*, vol. 69, no. 2, pp. 185–195, 1998.
- [28] M. Pauly and M. Gross. "Spectral processing of point-sampled geometry". *Proc. of SIGGRAPH'2001*, pp. 379–386, 2001.
- [29] M. Petrov *et al.* "Optical 3D digitizers: bringing life to the virtual world". *IEEE Computer Graphics and Applications*, vol. 18, no. 3, pp. 28–37, 1998.
- [30] R. Pico. "A solution to the next best view problem for automated surface acquisition". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.
- [31] M. K. Reed and P. K. Allen. "Constraint-based sensor planning for scene modeling". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1460–1467, 2000.
- [32] C. Robertson and R. B. Fisher. "Parallel evolutionary registration of range data". *Computer Vision and Image Understanding*, vol. 81, no. 1-3, pp. 39–50, 2002.

- [33] M. A. Rodrigues *et al.* (Eds.). *Special Issue on Registration and Fusion of Range Images. Computer Vision and Image Understanding*, vol. 81, no. 1-3, pp. 1-130, 2002.
- [34] G. Roth (Ed.). *Special Issue on 3D Digital Imaging and Modeling. Image and Vision Computing*, vol. 17, no. 2, pp. 97-166, 1999.
- [35] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm". *Proc. of 3rd Int. Conf. on 3-D Digital Imaging and Modeling*, pp. 145-152, 2001.
- [36] R. Schwarte, G. Häusler, and R. W. Malz. "Three-dimensional imaging techniques". In B. Jähne (Ed.), *Computer Vision and Applications*, Concise Edition, pp. 177-208, 2000.
- [37] *SIGGRAPH 2001, Course Notes 50, Digital Geometry Processing*, 2001.
- [38] M. Takatsuka *et al.* "Low-cost interactive active range finder". *Machine Vision and Applications*, vol. 14, no. 3, pp. 139-144, 2003.
- [39] K. Tarabanis, P.K. Allen, and R. Y. Tsai. "A survey of sensor planning in computer vision". *IEEE Trans. on Robotics and Automation*, vol. 11, no. 1, pp. 86-104, 1995.
- [40] G. Turk and M. Levoy. "Zippered polygon meshes from range images". *Proc. of SIGGRAPH'94*, pp. 311-318, 1994.
- [41] W. Welch and A. Witkin. "Free-form shape design using triangulated surfaces". *Proc. of SIGGRAPH'94*, pp. 247-256, 1994.
- [42] J. V. Wyngaerd and L. Van Gool. "Automatic crude path registration: toward automatic 3D model building". *Computer Vision and Image Understanding*, vol. 81, no. 1-3, pp. 8-26, 2002.
- [43] H. Zha *et al.* (Eds.). *Special Issue on 3-D Image Analysis and Modeling. IEEE Trans. on Systems, Man and Cybernetics - Part B*, vol. 33, no. 4, pp. 550-721, 2003.
- [44] Z. Zhang. "Iterative point matching for registration of free-form curves and surfaces"., *Int. Journal of Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.
- [45] J. Y. Zheng and Z. L. Zhang. "Virtual recovery of excavated relics". *IEEE Computer Graphics and Applications*, vol. 19, no. 3, pp. 6-11, 1999.

This page intentionally left blank

Chapter 18

RANGE IMAGE SEGMENTATION USING LOCAL APPROXIMATION OF SCAN LINES WITH APPLICATION TO CAD MODEL ACQUISITION*

Inas Khalifa ¹, Medhat Moussa ² and Mohamed Kamel ¹

1. *Department of Design Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*

2. *School of Engineering, University of Guelph, Guelph, Ontario N1G 2W1, Canada*
{inas,mkamel}@pami.uwaterloo.ca, mmoussa@uoguelph.ca

Abstract Automatic acquisition of CAD models from existing objects requires accurate extraction of geometric and topological information from the input data. This chapter presents a range image segmentation method based on local approximation of scan lines. The method employs edge models that are capable of detecting noise pixels as well as position and orientation discontinuities of varying strengths. Region-based techniques are then used to achieve a complete segmentation. Finally, a geometric representation of the scene, in the form of a surface CAD model, is produced. Experimental results on a large number of real range images acquired by different range sensors demonstrate the efficiency and robustness of the method.

Keywords: Range images, segmentation, scan lines, edge models, CAD model acquisition

18.1. Introduction

The process of *reverse engineering* aims at deriving CAD models of existing objects for which no such model is available [26]. This process is essential in many industrial applications such as re-manufacturing of parts for which no documentation is available, re-design through analysis and modification of old

*This chapter is reprinted with permission of Springer-Verlag from the Journal of Machine Vision and Applications, I. Khalifa, M. Moussa and M. Kamel, "Range image segmentation using local approximation of scan lines with application to CAD model acquisition", Volume 13, Numbers 5-6, Pages 263–274, 2003, Copyright © 2003 Springer-Verlag.

products to construct new and improved versions, and modeling of aesthetic designs that usually start with wood or clay prototypes [15]. It can also be useful to some computer graphics applications, as well as object recognition [7].

Many successful systems have been reported in the literature that generate polygonal mesh models from range data [20, 4, 19]. Although such models are useful for some applications such as terrain visualization and modeling of 3D medical data [24], they do not provide a high-level geometric description of a general scene, since they only use the polygon as a modeling primitive. They are also not concise, especially in the presence of complex curved surfaces. Hence, a more efficient and general modeling approach is to generate CAD models that decompose a scene into its natural surface components using higher level geometric primitives, e.g., natural quadrics.

Generally, a model acquisition system is composed of four basic modules, namely, data acquisition, segmentation, surface representation, and model building. Range images are among the most widely used types of input data, since they directly approximate the 3D shape of the scanned objects, and they are acquired automatically using fast and affordable range sensors. The segmentation module attempts to group elements of the input data such that each group corresponds to a certain meaningful part of the underlying object. The surface representation module produces a geometric description of each group. Finally, the model building module uses these geometric descriptions along with information about the object's topology to create a CAD model of the object. The four modules usually overlap and interact, rather than operate in sequence.

Although several encouraging results have been achieved in the design of the different components of a CAD model reconstruction system, a fully automatic solution is yet to be developed. One of the most serious limitations of the available model acquisition systems is the need for extensive operator interaction in data segmentation, e.g. [15, 25], which is a very tedious and time consuming task. Thus, an automated segmentation module is an *essential* part of the solution to the model acquisition problem. User interaction may also be required to specify the geometric representation of the different surfaces [28], and identify trimming curves.

This chapter describes a novel range image segmentation method based on local approximation of scan lines. The method employs edge models capable of detecting noise pixels, as well as capturing position and orientation discontinuities at different scales. Several region-based techniques are then used to obtain a complete segmentation and geometric surface representation of the input image. This information is used to acquire a surface CAD model of the scene automatically. A large number of real range images acquired by different range sensors is used to test the performance of this method. The rest of this chapter

is organized as follows. Section 2 introduces the new segmentation method and describes how complete segmentation is achieved. Section 3 describes the sequence of steps employed to build a surface CAD model of the scene, and discusses how the method can be used to approach the problem of acquiring complete CAD models from range data. Experimental results are presented in section 4. Summary and conclusion are given in section 5.

18.2. Range Image Segmentation

The goal of a range image segmentation algorithm is to partition the input range image into a set of surface regions that naturally decompose the scanned object, such that each region is represented by an appropriate geometric primitive. Edges in a range image can be classified as jump, fold, or smooth edges, which signify position, orientation, or curvature discontinuities, respectively. Segmentation techniques can be broadly categorized as either *edge-based* and *region-based* depending on whether they emphasize the detection of surface discontinuities or smooth surface regions, respectively [22].

While edge-based techniques tend to locate regions boundaries precisely, and are generally fast [27], they usually suffer from edge fragmentation and the need for efficient postprocessing, e.g. gap filling. Region-based techniques, on the other hand, always produce closed regions. However, they typically suffer from the possibility of over or under-segmentation, distortion of region boundaries, and/or the sensitivity to the choice of the initial seed regions, e.g., [8]. Also, they are typically iterative and are generally much slower than edge-based techniques. In fact, since the problems of detecting surface discontinuities and homogeneous surface regions are complementary, combining both techniques can potentially produce better results [6, 12].

Segmentation techniques that utilize scan lines exhibit high-speed performance and allow parallel processing. Region-based techniques that use scan lines have been reported in [11, 16]. However, they are limited to polyhedral objects. Jiang and Bunke presented an edge detection method based on scan lines approximation [13] using Duda's classical splitting algorithm [3]. Although this method is capable of segmenting curved objects, it suffers from two main disadvantages: (a) poor edge localization, since the splitting algorithm is not precise for noisy images and it provides no intuitive geometric relationship between the true edge location and the location where the lines are split, and (b) over-segmentation due to using a quadratic function, since surfaces of order 4 at least are required to accurately represent common quadrics such as spheres and cylinders [1].

To overcome these disadvantages, a novel edge-based segmentation method that is based on *local* approximation of scan lines is presented in this section. The local approximation technique has the potential to achieve better edge

localization and facilitates the detection of noise pixels. Jump edges detection is accomplished using the precise edge model described in [13]. A model for fold edge detection that is capable of capturing weak folds without producing false edges at high curvature regions is introduced. A complete segmentation is achieved by closing any contour gaps, and correcting instances of over and under-segmentation.

18.2.1 The Local Approximation Technique

The edge-based segmentation method is based on local approximation of scan lines using algebraic polynomials. Consider a dense range image regularly sampled in both the x and y directions. Each row or column in the image represents a vertical section of the scanned object surface. The pixels in each such section constitute a digital piece-wise smooth curve, or a *scan line*. Local approximation of scan lines is accomplished as follows. At each image pixel $\mathbf{x}_i$ in a scan line, the best approximating quadric of the range values in a neighborhood of N pixels is calculated by considering the N possible sets of pixels containing $\mathbf{x}_i$, i.e.,

$$\{\mathbf{x}_j : j = i + k - (N - 1), \dots, i + k\} \quad (18.1)$$

where $k = 0, 1, \dots, (N - 1)$, j is the index of $\mathbf{x}$ in the scan line, and N is odd. The best approximating quadric, $f_i(\mathbf{x}_i)$, is represented as:

$$f_i(\mathbf{x}_i) = a_i \mathbf{x}_i^2 + b_i \mathbf{x}_i + c_i \quad (18.2)$$

The value of N should be set small enough to insure fast processing, and allow correct representation of small regions but not too small that noise pixels cannot be identified. In the implementation, N is set through training as follows. Initially, N is set to the smallest acceptable odd value, $N = 5$, which means that the smallest allowable region should be at least 5 pixels wide. Then, a test is performed on a small number of images from a certain range sensor. The test is accomplished by executing the local approximation algorithm and identifying noise pixels. The resulting noise pixels are then examined visually. If the result is not satisfactory, i.e., too many pixels were labeled as noise, N is increased by 2, and the test is repeated.

This technique can potentially overcome the disadvantages of [13] as follows. It allows better approximation at and near jump and fold edges, and thus edges can be localized precisely. Also, the use of quadratic approximation functions will not result in over-segmentation, since only small local regions are considered. If higher order functions, i.e., comparable to N , or piece-wise functions such as splines are used, they would follow the data points very closely. This would be reasonable if the measurements are reliable, or noise-free, which is not the case in most of the available range sensors.

18.2.2 Detection of Noise Pixels

Range images are typically corrupted by different types of noise. Spike noise results in isolated image pixels with range values remarkably different from their surrounding. There are also some instances where a sensor is unable to make a range measurement. Moreover, the range measurements are sometimes valid, but there is insufficient information to discern separation of surfaces, e.g. surfaces almost parallel to the scanning direction may give rise to one or two pixel wide regions. Finally, there is quantization and measurement noise which adds a small tractable error randomly at each image pixel. This type of noise can be estimated by finding the best-fitting plane to a scanned surface known to be planar. The standard deviation, σ , of the error of fit indicates the severity of the noise. In the following, the first three types will be referred to collectively as *noise pixels*.

Noise pixels are identified using the technique of local approximation of scan lines as follows. Typically, one or more of the N sets, in Eq. 18.1, will produce a maximum fitting error comparable to δ , where δ is set to a value slightly larger than σ . However, for noise pixels, the fitting error of *all* sets will exceed δ . A pixel is labeled as a noise pixel if the minimum fitting error in both the x and y directions exceeds δ .

18.2.3 Detection of Jump and Fold Edges

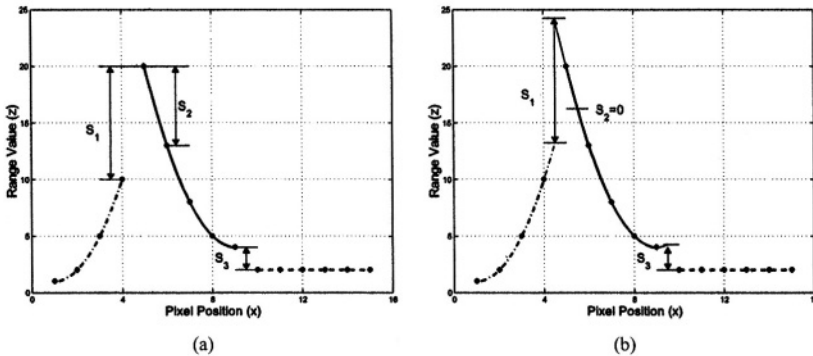


Figure 18.1. Illustration of the jump edge detection model. (a) Conventional measure of jump edge strength. (b) Jump edge strength measure used in [13].

Jump edges signify a discontinuity in position. Traditionally, the difference in range values between two neighboring pixels was used as a measure of jump edge strength and then compared against a threshold to locate jump edges. Equivalently, other techniques used the range value gradient as a measure of

jump strength instead. Figure 18.1 (a) shows a scan line composed of three curve segments separated by jump edges and illustrates an example of the former measure. It is obvious that in the presence of surfaces with a high slope, it is difficult to detect jump edges at different scales, i.e. with varying strengths, using this measure. For example, if the threshold was set lower than S_2 to detect the jump edge separating the two segments on the right, false jump edges will result in the first segment. Alternatively, if the threshold was set higher than S_1 to avoid those false edges, the jump edge on the right will be missed.

Instead, the precise definition of the jump edge strength [13] illustrated in Figure 18.1 (b), is used as follows. Let $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ be two adjacent pixels, approximated locally by the two functions $f_i(\mathbf{x})$ and $f_{i+1}(\mathbf{x})$. The jump strength is defined as:

$$S_{\text{jump}} = |f_i(\bar{\mathbf{x}}) - f_{i+1}(\bar{\mathbf{x}})|, \quad \bar{\mathbf{x}} = \frac{\mathbf{x}_i + \mathbf{x}_{i+1}}{2} \quad (18.3)$$

If S_{jump} is larger than a threshold T_{jump} , both $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are labeled as jump edge pixels, i.e., each of them lies on the boundary of its region. It is clear that this edge model alleviates the problems encountered with the conventional measures. The threshold T_{jump} is determined through training, where its initial value is set slightly larger than the maximum noise level, and it can be fixed for a certain sensor.

The strength of fold edges, which signify a discontinuity in surface orientation, is conventionally measured by the difference in surface normals between two adjacent pixels, or the maximum curvature at an image pixel. Figure 18.2 (a) shows a scan line composed of three curve segments separated by two fold edges. Figure 18.2 (b) depicts the angle of the surface normal vector at each pixel measured from the x axis. It also illustrates an example of the former fold strength measure.

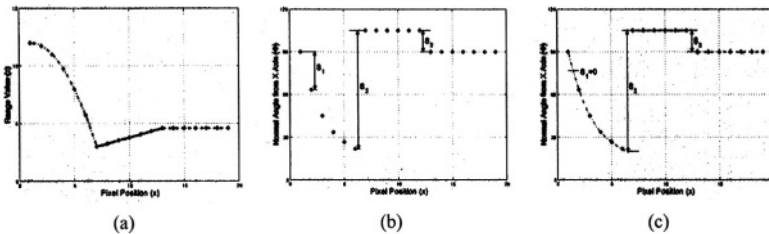


Figure 18.2. Illustration of the fold edge detection model. (a) A scan line segment. (b) Conventional measure of fold edge strength. (c) New measure of fold edge strength.

As can be seen from the figure this measure is not capable of capturing folds at different scales. For example, in order to capture a fold edge at the junction of

two planes with a small difference in surface normals, e.g. S_2 , the threshold will be set too low that pixels on a highly curved surface will be labeled incorrectly as edges. Alternatively, if the threshold was set higher than S_1 to avoid those false edges, the fold edge on the right will be missed. Using the surface curvature as a measure of fold strength encounters the same problems, in addition to being prone to the approximation errors which are inevitable in curvature calculation.

To overcome this problem, a model for fold edge detection that attempts to capture the *trend* of the surface orientation, illustrated in Figure 18.2 (c), is introduced. Consider the scan line shown in Figure 18.2 (a). Using the best local approximating function of the range values $f_i(\mathbf{x})$ at apixel $\mathbf{x}_i$, its normal vector can be calculated as follows:

$$\begin{aligned} \text{slope} &= f'_i(\mathbf{x}_i) = 2a_i\mathbf{x}_i + b_i \\ \phi_i &= \tan^{-1} \frac{-1}{\text{slope}} \\ \underline{\mathbf{n}}_i &= [\cos(\phi_i) \quad \sin(\phi_i)]^T \end{aligned} \quad (18.4)$$

where ϕ_i is the angle between the surface normal, $\underline{\mathbf{n}}_i$ and the x axis at a pixel $\mathbf{x}_i$. Note that the angle changes are relatively *smooth* on each surface, however, at the pixels where two surfaces meet, the angle change is large, i.e., a *jump discontinuity* in the *angle values* occurs. Thus, in order to detect fold edges, the function $g_i(\mathbf{x})$, that best approximates the angles locally, is calculated at each pixel $\mathbf{x}_i$ in a neighborhood of N pixels as before. The fold strength is then calculated as:

$$S_{\text{fold}} = |g_i(\bar{\mathbf{x}}) - g_{i+1}(\bar{\mathbf{x}})| \quad (18.5)$$

The fold strengths are then compared against a threshold T_{fold} to detect fold edges. It is clear that using this definition, T_{fold} can be set low enough to capture weak fold edges without producing false edges on curved surfaces. Moreover, in order to use the above model, the normals at, and in the vicinity, of fold edges should be calculated with negligible error which is facilitated by the local approximation method.

In the current implementation, jump edge detection is performed in the x and y directions, i.e., image rows and columns, whereas fold edge detection is performed in the x and y directions, as well as the two diagonal directions.

18.2.4 Complete Segmentation and Surface Description

The output of the edge detection process is a binary edge map, which typically contains gaps in region boundaries. In order to fill those gaps, an adaptive pixel grouping process [12] is used. This process starts by identifying the different image regions using a connected component labeling algorithm. Then, a region test is performed for each region. The region test is a surface fitting operation, and a region passes the test if the RMS fit error is lower than a certain threshold T_{RMS} and its size is larger than T_{size} . If a region passes the region test, it is

registered and added to the final region map. Otherwise, the edges in that region are dilated once in hopes of closing an existing gap. The above steps are repeated until all regions pass the test. After that, each of the edge pixels, from edge detection and dilation, is assigned to one of the adjacent (4-neighbor) regions where the distance between the pixel and the region is minimum, among all such regions, and that the distance is less than T_{RMS} . The value of T_{RMS} is set using training, where its initial value is lower than δ . It is then decreased in each training iteration until an acceptable segmentation of the training images is achieved.

Instances of over-segmentation may occur due to false edges produced by the edge detection method, that have formed a closed contour due to dilation. To treat this problem, adjacent regions are examined using the region test. If a pair of regions passes the region test, the two regions are merged. Instances of under-segmentation may also occur due to the presence of smooth edges or the presence of closed regions that fail the region test due to the limited surface types allowed. This problem is treated by adding new regions to accommodate such cases as follows. A connected component labeling process is started at an unlabeled pixel and continued as long as the new region passes the region test. The new region is then registered and the above steps are repeated until all unlabeled pixels are accounted for. Although this procedure may not always be able to recover the missed edges, it produces a surface representation that exhibits tolerable deviation, T_{RMS} , at all pixels.

The set of surface types used in the region test, or *segmentation primitives*, can include virtually any family of surfaces, such as planes, biquartic patches, tensor-product NURBS, as well as conventional quadrics. Intuitively, the segmentation primitives should be chosen such that they can accurately represent a wide variety of objects. The limiting factor is that these primitives should be approximated within a reasonable cost, i.e. computational complexity, which is determined by the application requirements. For example, including tensor-product NURBS patches may allow the representation of very complex sculptured surfaces. However, fitting NURBS patches to regions with irregular, i.e. non-rectangular, boundaries requires complex computations [2].

The segmentation primitives used in this implementation are planes, spheres, biquartic patches. These primitives are approximated in a non-iterative manner and at very low complexity, which speeds up the segmentation process. Biquartic patches are used because they are capable of accurate representation of conventional quadrics. Algorithms for fitting conventional quadrics including cones, tori, and cylinders have also been implemented [14]. However, since these techniques involve solving nonlinear least-squares problems and thus increase the processing times remarkably, it has been chosen to include them in the segmentation primitives only when such specific representations are required by the application.

18.3. CAD Model Building

The segmentation and surface representation techniques described above produce a region map as well as a geometric description of each surface region in the segmented image. This information is used to construct a surface CAD model of the scene as follows. Since surface models support only parametric surfaces, the surface description must be converted into a parametric form. A NURBS (Non-Uniform Rational B-Splines) representation [18] is adopted here, since it provides a flexible representation of curves and surfaces, and is very widely used in modeling and visualization systems. Planar surface regions are converted into bilinear Bézier patches, while spherical regions are converted into biquadratic NURBS patches with 9×5 control nets. Biquartic surface regions are parameterized over their bounding boxes to obtain a parametric polynomial representation, which is then converted into biquartic Bézier patches.

Figure 18.3 illustrates the sequence of steps involved in the model building process. After finding the NURBS representation of a region, surface trimming is accomplished as follows. First, the boundary contours of the region are identified using the region map. Then, the different curve segments that compose each contour are identified. Finally, each contour is mapped back onto the pa-

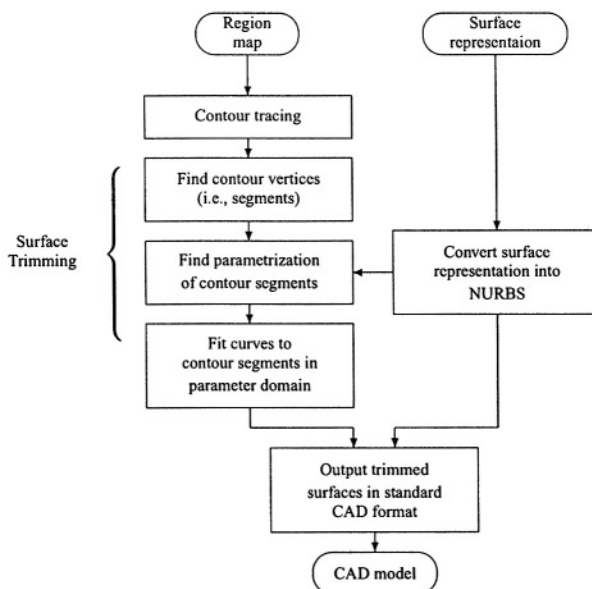


Figure 18.3. A flow diagram of the surface model building process.

parameter domain, and the mapping is approximated by the appropriate NURBS curve(s). These curves are used to trim the NURBS patch, i.e., define the surface part bounded by the region contours. Finally, trimmed surfaces are output in the standard IGES and VRML formats. The above sequence is repeated for each surface region in the segmented image to obtain a surface CAD model of the whole scene.

18.3.1 Application to CAD Model Acquisition

The framework described above, which consists of the segmentation and model building processes, produces a geometric model of a scene from a single range image. This framework can be useful in a number of applications. Since it produces a concise CAD representation from the voluminous input range image, in a relatively short time, the models can be quickly imported in CAD system for viewing, surface analysis, and manipulation. It can also be used for some computer vision tasks such as collision avoidance.

Since a single range image is used as input, the framework produces partial models of the objects in a scene. In order to construct a complete model, multiple range views are required. Principally, there are two approaches to integrating multiple range views: either to process each view to produce a partial model and then merge the acquired models at the feature level, or to merge the data at the pixel, or point set, level, before segmenting and further processing.

The first approach has the advantage of having many existing segmentation algorithms that can be used for feature extraction from a single image. However, automating the process of merging the different surface patches is a difficult task. Sequeira *et al* [23] presented an example of this approach. The second approach, on the other hand, avoids the problem of merging the surface patches. However, the feature detection methods required are more complex. Examples include the systems presented in [28, 15]. They, however, require the user to segment the data manually and remove outliers. An automated algorithm has been proposed by Fisher *et al* [5], which uses a tessellation algorithm [10] to obtain adjacency information which facilitates the extraction of surface patches. Adjacent patches are then intersected to adjust the edges.

A few algorithms that fit between the two approaches have been proposed. Park and Lee [17] developed an algorithm to approximate NURBS patches from either cloud data or range images. Unfortunately, this algorithm suffers from over-segmentation of surface patches because it does not take surface features into account. Also, Roth and Boulanger [21] have developed a model acquisition system that uses both cloud data and registered range images. They extract free-form surfaces automatically from the different range views, and require the user interaction to locate quadric surfaces from the cloud data.

Table 18.1. Segmentation parameters.

Data	N	δ	T_{jump}	T_{fold}	T_{RMS}
MSU	5	0.02mm	0.5mm	20°	0.013mm
NRC	5	0.5d*	1.5d	15°	0.2d
UE	5	0.6mm	1.5d	15°	0.45mm

* Average sampling density.

The output of both the segmentation and model building methods presented above, can be useful to both approaches. For the first approach, the system can be used to provide a CAD model for different range views, which can then be manipulated and merged, either manually or automatically. For the second approach, the output of the segmentation process provides a set of fold edge pixels, which can be used after merging the different views to help guide the automated surface patch extraction process. Jump edges will not be used, since they only occur in partial views of an object. The labeled fold points can also enhance polygonal mesh generation systems, which can avoid interpolating the data at these points, and thus eliminate *rounded edges*.

Moreover, taking into account the state of the reverse engineering systems currently available, which require extensive user manipulation, our framework offers a lot of improvement. Since partial models are generated automatically, the user is only required to work with information at the surface-level—as opposed to the current pixel-level manipulation—which is much more appropriate, and less time consuming for humans.

18.4. Experimental Results

The model acquisition system presented in this chapter has been tested on 157 images from Michigan State University (MSU)¹, 100 images from the Canadian National Research Council (NRC)², and 33 images from the University of Edinburgh (UE)³ range image databases. Sample results are shown in Figures 18.4 through 18.6. The columns show the original images, segmented images, and rendered CAD models, respectively.

Tables 18.1 and 18.2 summarize the segmentation and average performance parameters, respectively. The segmentation and model building methods were implemented in C and run on a PC with an Intel PIII 500MHz processor and 64M RAM, running Linux. The results in Table 18.2 demonstrate the system efficiency in terms of:

¹<http://sampl.eng.ohio-state.edu/~sampl/data/3DDB/RID/index.html>

²http://www.vit.iit.nrc.ca/3D/Pages_HTML/3D_Images.html

³<ftp://ftp.dai.ed.ac.uk/pub/vision/range-images/>

Table 18.2. Performance evaluation parameters.

Data	Image size	No. of regions	Actual RMS	P_1	Iterations [†]	Time*
MSU	207×176	6.68	0.032mm	2.11%	2.41	27sec.
NRC	256×256	29.57	0.17d	0.6%	3.33	136sec.
UE	189×145	27.94	0.26mm	3.88%	2.39	88sec.

† Adaptive pixel grouping iterations.

* The segmentation time, defined as the elapsed (real) time between invocation of the program and its termination as calculated by the UNIX `time` utility.

- Capability of processing range images, acquired by different sensors, which exhibit different combinations of complexity, structure, and surface types, equally well as shown in the sample images.
- Accurate surface representation since the actual average RMS error between the acquired models and the original range data points is small compared to both δ and T'_{RMS} .
- The percentage of pixels where the fit error exceeds δ , P_1 , is relatively small.
- The number of iterations required in the adaptive pixel grouping process is very small, compared to iterative region-based segmentation techniques, e.g. [6].
- The average overall processing times are relatively short, i.e. a CAD model of the scene is acquired automatically in about two minutes.

The results also demonstrate the robustness of the segmentation method in a multitude of ways:

- The local approximation method is capable of identifying noise pixels at different scales, e.g. Figure 18.4 (c).
- The local approximation technique allows identification of small regions and holes, e.g., Figures 18.5 (b) and 18.6 (b).
- Jump and fold edges of different strengths are successfully identified using the local approximation technique, e.g. Figure 18.6 (c).
- The segmentation method is capable of correcting instances of under-segmentation that occur due to the limited surface types used in the region test. Thus, the system can acquire models from scenes that contain complex free-form objects as shown in Figures 18.5 (a) and 18.5 (c).
- Although there is no special procedure to identify smooth edges, the method was able to recover them in some cases, e.g. the junction of planar and cylindrical surfaces in Figure 18.6 (a).

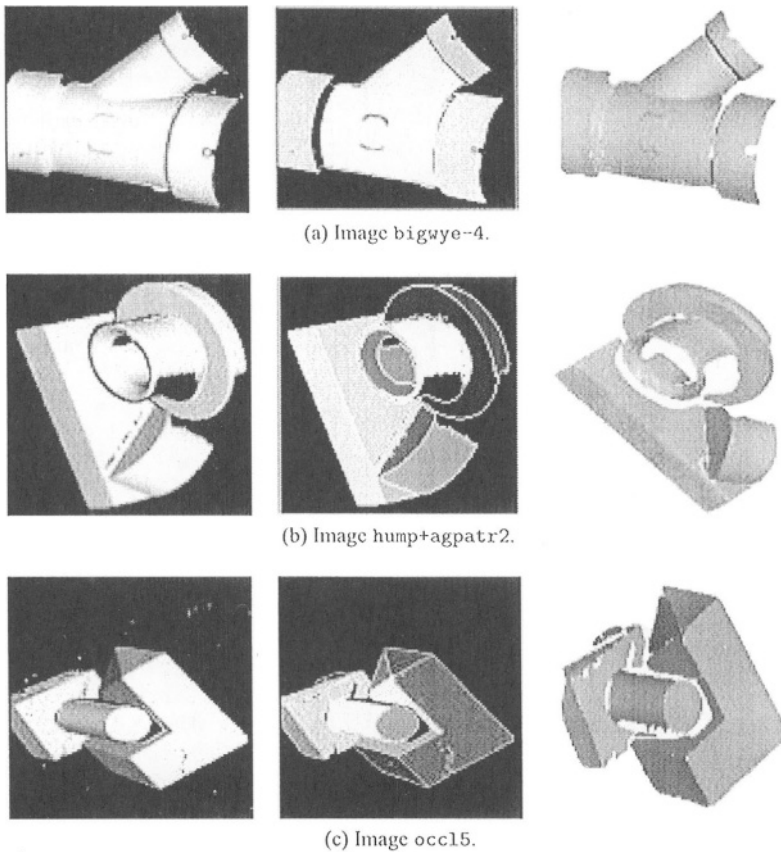


Figure 18.4. Segmentation and model building results of MSU images.

18.4.1 Segmentation Comparison Project

The segmentation comparison project [9] compares machine segmented images (MS), produced by a segmentation algorithm, against ground truth images (GT), which is produced by human operators. It classifies regions in an MS image into one of the following five categories:

- 1 An instance of **correct detection**: a region in the MS image is found in the GT image.
- 2 An instance of **over-segmentation**: a single region in the GT image is divided into more than one region in the MS image.

- 3 An instance of **under-segmentation**: more than one region in the GT image are grouped into a single region in the MS image.
- 4 An instance of a **missed** classification: a region in the GT image that does not participate in any instance of correct detection, over-segmentation, or under-segmentation.
- 5 An instance of a **noise** classification: a region in the MS image that does not participate in any instance of correct detection, over-segmentation, or under-segmentation.

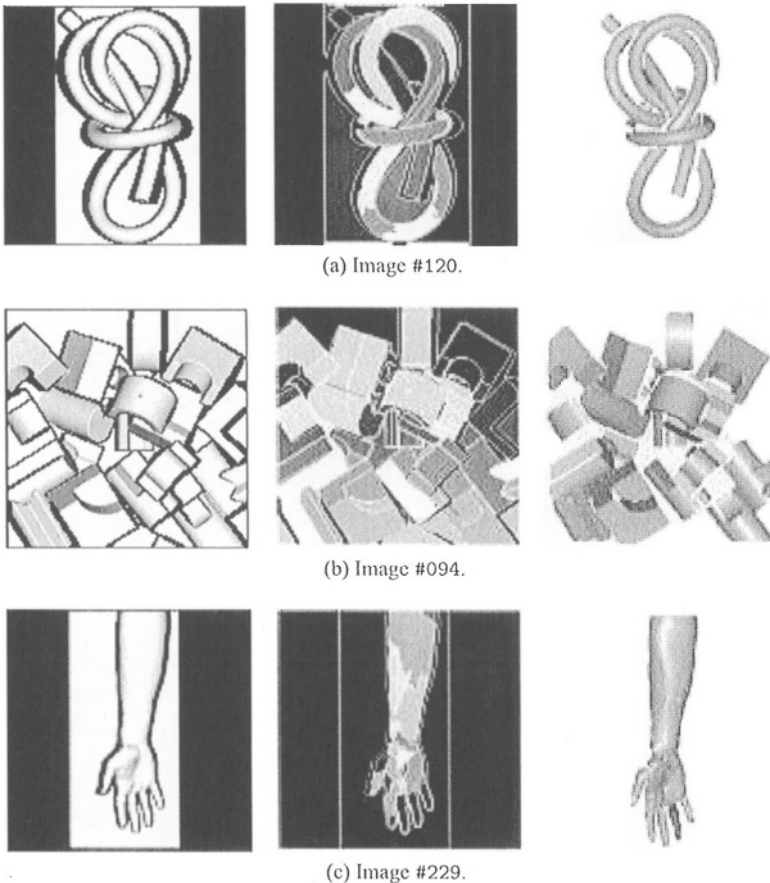


Figure 18.5. Segmentation and model building results of NRC images.

A metric describing the accuracy of the recovered geometry is calculated as follows. Given the angle difference between two adjacent regions in the GT, A_n , if both regions are classified as correct detection, their angle difference is calculated from the MS image as A_m . The average values of $|A_n - A_m|$ and the standard deviation are then calculated. Finally, the compare threshold T is used to reflect the strictness of the above criteria, where the larger value of T implies more strict classification. For example, if $T = 100\%$, then a region in the MS image must completely overlap with a region in the GT image to classify it as a correct detection instance. It is proved in [9] that classifications are not unique for $T < 100\%$, and that for $50\% < T < 100\%$ any region can contribute to at most three classifications, one each of correct detection, over-segmentation, and under-segmentation.

The method presented in this chapter was originally developed for data sets where the scan planes are parallel to the principal planes, xz and yz . It has been modified to work with general scan planes and used to segment the segmentation comparison data set captured by the ABW structure light scanner. To determine the segmentation parameters to be used, the 10 training images were segmented

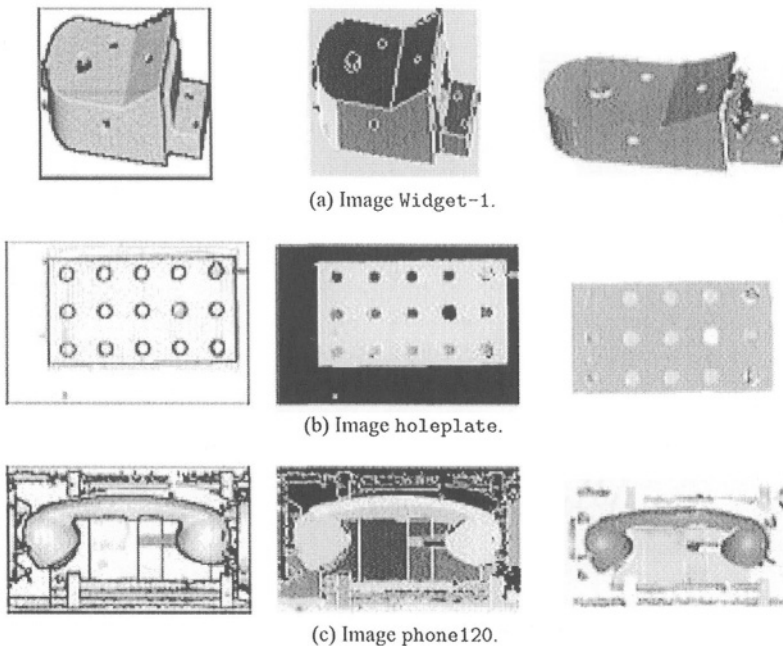


Figure 18.6. Segmentation and model building results of UE images.

Table 18.3. Segmentation comparison results at $T = 80\%$.

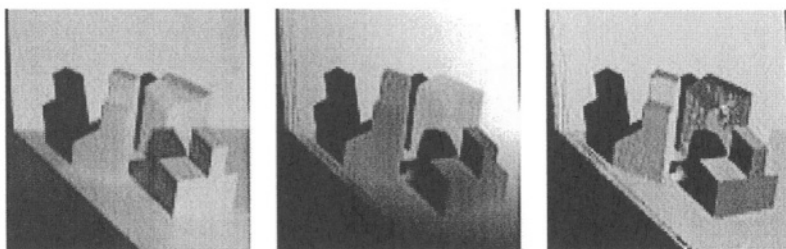
Algorithm	GT regions	correct detection	angle diff. (std. dev.)	over- segmented	under- segmented	missed	noise
USF	15.2	12.7	1.6(0.8)	0.2	0.1	2.1	1.2
WSU	15.2	9.7	1.6(0.7)	0.5	0.2	4.5	2.2
UB	15.2	12.8	1.3(0.8)	0.5	0.1	1.7	2.1
UE	15.2	13.4	1.6(0.9)	0.4	0.2	1.1	0.8
EG	15.2	13.5	not avail.	0.2	0.0	1.5	0.8
SLA	15.2	10.7	2.0(1.1)	1.0	0.6	2.1	4.8

using the following parameter sets: $\delta = \{2.5, 3.0\}$, $N = \{9, 11\}$, $T_{jump} = \{10\}$, $T_{fold} = \{25, 30\}$, and $T_{rms} = \{1.25, 1.5\}$. All combinations were used. The set which produced the highest number of correct detections, which is $\{2.5, 9, 10, 25, 1.5\}$, was used to segment the test images.

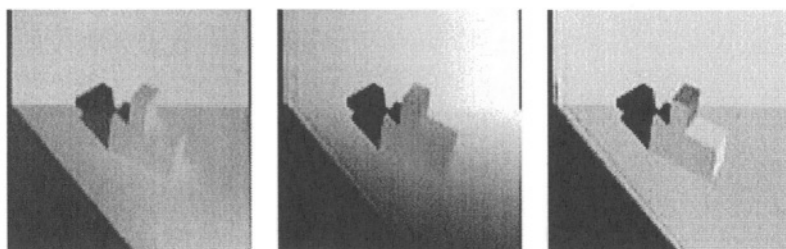
Figure 18.7 show samples of the segmentation results using the ABW images. The columns show the intensity, range, and segmented images, respectively. Figures 18.8 (a) through 18.8 (e) show the average segmentation results obtained using the 30 test images for the four algorithms compared in [9], namely UB, UE, USF, and WSU, as well as the method presented in this chapter (SLA, for Scan line Local Approximation). Jiang and Bunke [13] also evaluated their method, referred to as EG, using the compare tool, but their segmented images were not available to the authors to be able to re-generate the graphs shown in Figure 18.8. However, their results were very similar to the UE and UB results. Table 18.3 summarizes the results for the seven criteria mentioned above at a compare threshold value of 80%, for all six algorithms. Given the data in Figure 18.8 and Table 18.3, it is clear that there is no *across the board winner* among the algorithms compared.

The average processing times for the algorithms were 78 minutes (USF) on a Sun SparcStation 20, 6.3 minutes (UE) on a Sun SparcStation 5, 4.4 minutes (WSU) on a HP 9000/730, 7 seconds (UB) on a Sun SparcStation 20, and 15 seconds (EG) on a Sun SparcStation 5. The average processing time for SLA, as calculated using the UNIX time utility, was 2.55 minutes of user time, 0.26 seconds of system time, and 2.58 minutes of real time, using a PC with an Intel PIII 500MHz processor and 64M RAM, running Linux.

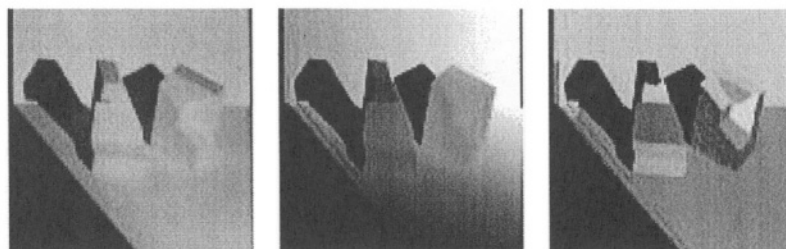
The results shown above show that, although not better than [13], the performance of the local approximation method is still comparable to the other five algorithms, with respect to correct detections, angle, and standard deviation. It had a higher number of over-segmentation and noise instances. This is mainly due to the fact that the parameter that guides the post processing phase is the RMS surface fitting error, T_{rms} , was fixed to a certain value, 1.5. However, the ground truth of the training and testing images exhibited fitting error with the characteristics shown in Table 18.4. So, given a fixed T_{rms} , the method has



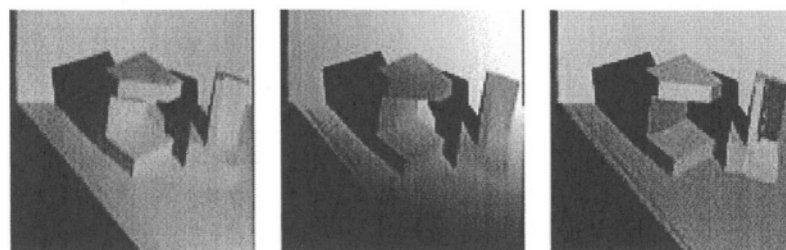
(a) Image abw.test.8.



(b) Image abw.test.11.



(c) Image abw.test.15.



(d) Image abw.test.18.

Figure 18.7. Segmentation results of the ABW data set.

18.5. Conclusion and Future Work

This chapter has presented a range image segmentation method that is based on local approximation of scan lines. The method employed edge detection models capable of identifying jump and fold edges of varying strengths. This was followed by region-based techniques to close gaps in region boundaries and correct instances of over and under-segmentation. The output of the segmentation method was used to acquire a surface CAD model of the scene in the form of trimmed NURBS patches. Application to the different complete model acquisition approaches has also been discussed.

A large number of real range images from different databases has been used to test the performance of the segmentation and model building methods, and the reported results demonstrated their efficiency and robustness. The performance of the segmentation method has also been evaluated using a segmentation comparison framework. It has been shown that the performance of the local approximation method is comparable to other segmentation algorithms with respect to most of the comparison criteria. An explanation why the performance was worse according to the remaining criteria was also presented.

Future investigation can address the following issues. First, improving the region-based techniques used to obtain a complete segmentation. Second, adding more segmentation primitives, such as NURBS patches, which will improve the surface representation capability. It can also be improved by including the conventional quadrics, which were optional in this implementation. However, although these surface types have a better representation capability, their fitting algorithms are computationally expensive. Thus, the approach of using them in the process of correcting over- and under-segmentation only, is expected to lower the computational cost.

Table 18.4. Characteristics of the surface fitting error in the ABW images.

Images	Average	Minimum	Maximum	Std. Dev.	Perc. > 1.5 [†]
Training	0.78	0.0	5.09	0.52	4.48%
Testing	0.78	0.0	7.09	0.54	5.91%

[†] Percentage of regions with fit error > 1.5.

Acknowledgments

The authors acknowledge the Canadian Natural Sciences and Engineering Research Council (NSERC) support of this work through a graduate scholarship to the first author and operating grants to the second and third authors.

References

- [1] P. J. Besl. *Surfaces in Range Image Understanding*. Springer-Verlag, 1988.
- [2] P. Dierckx. *Curve and Surface Fitting with Splines*. Clarendon Press, 1993.
- [3] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1972.
- [4] R. E. Fayek. "Feature-based 3D surface modeling by improved constrained triangular meshes". In *Proc. of the IEEE International Conference on Image Processing, ICIP '97*, vol. 2, pp. 740-743, Santa Barbara, CA, 1997.
- [5] R. B. Fisher, A. W. Fitzgibbon and D. Eggert. "Extracting surface patches from complete range descriptions". In *Proc. of the First International Conference on Recent Advances in 3-D Digital Imaging and Modeling, 3DIM '97*, pp. 148-154, Ottawa, ON, 1997.
- [6] A. W. Fitzgibbon, D. W. Eggert and R. B. Fisher. "High-level CAD model acquisition from range images". *Computer-Aided Design*, vol. 29, pp. 321-330, 1997.
- [7] P. J. Flynn and A. K. Jain. "CAD-Based computer vision: From CAD models to relational graphs". In *Computer Vision: Advances and Applications*, pp. 228-246, R. Kasturi and R. C. Jain Eds., IEEE Computer Society Press, 1991.
- [8] R. Hoffman and A. K. Jain. "Segmentation and classification of range images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 608-620, 1987.
- [9] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon and R. B. Fisher. "An experimental comparison of range image segmentation algorithms". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 673-689, 1996.
- [10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. "Surface reconstruction from unorganized points". *Computer Graphics*, vol. 26, pp. 71-78, 1992.
- [11] X. Jiang and H. Bunke. "Fast segmentation of range images into planar regions by scan line grouping". *Machine Vision and Applications*, vol. 7, pp. 115-122, 1994.
- [12] X. Jiang and H. Bunke. "Range image segmentation: Adaptive grouping of edges into regions". In *Proc. of the Third Asian Conference on Computer Vision, ECCV '98*, pp. 726-733, R. Chin and T.-C. Pong Eds., Springer-Verlag, 1998.
- [13] X. Jiang and H. Bunke. "Edge Detection in Range Images Based on Scan Line Approximation". *Computer Vision and Image Understanding*, vol. 73, pp. 189-199, Academic Press, February, 1999.
- [14] G. Lukács, R. Martin and D. Marshall. "Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation". In *Proc. of the Fifth European Conference on Computer Vision, ECCV '98*, pp. 671-686, H. Burkhardt and B. Neumann Eds., Springer-Verlag, 1998.
- [15] M. J. Milroy, D. J. Weir, C. Bradley and G. W. Vickers. "Reverse engineering employing a 3D laser scanner: a case study". *The International Journal of Advanced Manufacturing Technology*, vol. 12, no. 111-121, 1996.
- [16] E. Natonek. "Fast Range Image Segmentation for Servicing Robots". In *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 406-411, May, 1998.
- [17] I. K. Park and S. U. Lee. "Geometric modeling from scattered 3-D range data". In *Proc. of the IEEE International Conference on Image Processing, ICIP '97*, vol. 2, pp. 712-715, Santa Barbara, CA, 1997.
- [18] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, 1995.

- [19] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro and W. Stuetzle. "Robust meshes from multiple range maps". In *Proc. of the First International Conference on Recent Advances in 3-D Digital Imaging and Modeling, 3DIM '97*, pp. 205-211, Ottawa, ON, 1997.
- [20] G. Roth and E. Wibowoo. "An efficient volumetric method for building closed triangular meshes from 3-D image and point data". In *Proc. of Graphics Interface*, pp. 173-180, Kelowna, BC, 1997.
- [21] G. Roth and P. Boulanger. "CAD model building from multiple range images". In *Proc. of Vision Interface*, pp. 431-438, Vancouver, BC, 1998.
- [22] M. Suk and S. M. Bhandarkar. *Three-Dimensional Object Recognition from Range Images*. Springer-Verlag, 1992.
- [23] V. Sequeira, J. G. M. Goncalves and M. I. Ribeiro. "High-level surface descriptions from composite range images". In *Proc. of the IEEE International Symposium on Computer Vision, ISCV '95*, pp. 163-168, Coral Gables, FL, 1995.
- [24] C. K. Tang and G. Medioni. "Extremal feature extraction from 3-D vector and noisy scalar fields". In *Proc. of the Ninth IEEE Conference on Visualization*, pp. 95-102, Research Triangle Park, NC, October, 1998.
- [25] W. B. Thompson, J. C. Owen, J. de St. Germain, S. R. Stark and T. C. Henderson. "Feature-based reverse engineering of mechanical parts". *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 57-66, 1999.
- [26] T. Várady and R. R. Martin and J. Cox. "Reverse engineering of geometric models – an introduction". *Computer-Aided Design*, vol. 29, pp. 255-268, 1997.
- [27] M. A. Wani and B. G. Batchelor. "Edge-region-based segmentation of range images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 314-319, 1994.
- [28] D. J. Weir, M. J. Milroy, C. Bradley and G. W. Vickers. "Reverse engineering physical models employing wrap-around B-spline surfaces and quadrics". In *Proc. of the Institute of Mechanical Engineers*, vol. 210, pp. 147-157, 1996.

This page intentionally left blank

Chapter 19

SHAPE-BASED IMAGE RETRIEVAL APPLIED TO TRADEMARK IMAGES*

Ossama El Badawy and Mohamed Kamel

Department of Systems Design Engineering, University of Waterloo, Waterloo,

Ontario N2L 3G1, Canada

{obadawy,mkamel}@pami.uwaterloo.ca

Abstract

In this chapter, we propose a new shape-based, query-by-example, image database retrieval method that is able to match a query image to one of the images in the database, based on a whole or partial match. The proposed method has two key components: the architecture of the retrieval and the features used. Both play a role in the overall retrieval efficacy. The proposed architecture is based on the analysis of connected components and holes in the query and database images. The features we propose to use are geometric in nature, and are invariant to translation, rotation, and scale. Each of the suggested three features is not new per se, but combining them to produce a compact and efficient feature vector is. We use hand-sketched, rotated, and scaled, query images to test the proposed method using a database of 500 logo images. We compare the performance of the suggested features with the performance of the *moments invariants* (a set of commonly-used shape features). The suggested features match the moments invariants in rotated and scaled queries and consistently surpass them in hand-sketched queries. Moreover, results clearly show that the proposed architecture significantly increase the performance of the two feature sets.

Keywords:

Shape Analysis, shape representation, shape-based image retrieval, image databases, trademark image retrieval

19.1. Introduction

Today, a lot of images are being generated at an ever increasing rate by diverse sources such as earth orbiting satellites, telescopes, reconnaissance and

*This chapter is a reprint of the paper with the same title appearing at the International Journal of Image and Graphics, Vol. 2, No. 3, 2002, pp.375-393.

surveillance planes, fingerprinting and mug-shot-capturing devices, biomedical imaging, payment processing systems, and scientific experiments.

There is a pressing need to manage all this information that keeps pouring into our life each day. By *managing* images we more specifically want to be able to efficiently store, display, group, classify, query, and retrieve, those images. An image database management system is generally required if we are to get the most out of a large image collection. One of the major key services that should be offered by an image database system is its ability to provide *content-based image retrieval*, or CBIR. The goal of content-based image retrieval is to retrieve database images that contain certain visual properties, as opposed to retrieving images based on other properties such as creation date, file size, or author/photographer name. For example, we might be interested in retrieving images where blue is the most abundant color (color-based image retrieval). Another example is when we ask the system to, say, get us all images that contain squares or rectangles (shape-based image retrieval).

Application areas in which CBIR plays a principal role are numerous and diverse. Among them are art galleries and museum management, trademark and copyright database management, geographic information systems, law enforcement and criminal investigation, weather forecasting, retailing, and picture archiving systems.

The focus of this chapter is on shape-based image retrieval using a feature extraction approach. The main goal is to be able to retrieve database images that are most similar to a hand-sketched query image. Figure 19.1 shows a block diagram of the image retrieval model used in the feature extraction approach. The input images are pre-processed to extract the features which are then stored along with the images in the database. When a query image is presented, it is as well pre-processed to extract its features which are then matched with the feature vectors present in the database. A ranked set of images with high matching scores are presented at the output.

We want to be able to retrieve database images similar, in whole or in part, to a user-supplied query image. In particular, we want to be able to retrieve database images based on (see Figure 19.1):

- a *whole* match with scaled, hand-sketched queries (first row),
- a *partial* match with scaled, rotated, hand-sketched queries (second row),
- a match where the number of components in the query and the retrieved database image is different but the overall shape is similar (third row).

We propose a retrieval architecture based on the analysis of connected components and holes in the image. Any feature set can be used with this architecture, but we suggest the use of a set of three known object properties as image features suitable for shape-based retrieval. The suggested feature set is invariant

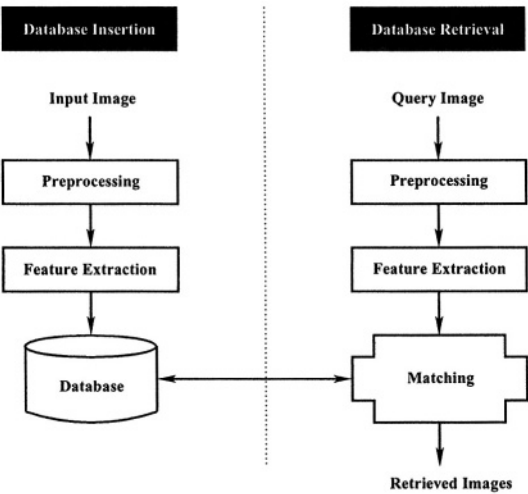


Figure 19.1. Feature extraction approach to image retrieval.

Sample Query	Sample Retrieval
	
	
	

Figure 19.2. Examples of queries that we want to be able to correctly answer. Left column: Query image. Right column: Retrieved image.

to translation, rotation, and scale. The performance of the proposed features is compared to that of the moments invariants feature set in two ways: once in conjunction with the proposed architecture and once on their own. All images used are binary images.

The rest of the chapter is organized as follows. Section 19.2 provides a literature review of recent shape-based image retrieval research. Then, in Section

19.3, the necessary theoretical background is presented and the shape feature set is explained in detail. Section 19.4 details the proposed architecture. Performance analysis results are presented in Section 19.5. Finally, the conclusion and a discussion of future work are provided in Section 19.6.

19.2. Overview of Current Methods

Nowadays, numerous shape analysis and representation techniques exist. These include chain codes, polygonal approximations, signatures, edge histograms, Fourier descriptors, moments and image morphology [1]. As shape is one of the main characteristics of images, it has been used a lot in image database systems as a way of automatically retrieving images by content. In almost all implemented systems, other image cues (i.e., texture and color), in addition to shape, are also used for image retrieval. In this work, we focus more on shape-based retrieval.

In Safar [2], a shape representation method based on minimum bounding circles and touch-point vertex-angle sequences is presented and compared with other common representation like Fourier descriptors and Delaunay triangulation. In another paper [3], the authors conduct a comparative study on various shape-based representation techniques under various uncertainty scenarios, like the presence of noise and when the exact corner points are unknown. In Lu [4], a region-based approach to shape representation and similarity measure is presented. In Kwok [5], edge histograms as shape features (and other color features) are used for retrieval from a database of 110 flower images. In Xu [6], a shape representation algorithm based on the morphological skeleton transform is presented. The algorithm represents shapes as union of special maximal rectangle contained in the shape.

In Adoram [7], an algorithm that uses snakes and invariant moments is presented. In Kim [8], a modified Zernike moment descriptor that takes into account the importance of the outer form of the shape to human perception is proposed. In Muller [9], a deformation-tolerant stochastic model suitable for sketch-based retrieval is presented.

IBM's QBIC system primarily uses statistical features to represent the shape of an object [10]. These include a combination of area, circularity, eccentricity, major axis orientation, and algebraic invariant moments. The ART-MUSEUM system was developed for the purpose of archiving a collection of artistic paintings [11]. In this system, the database consists of the contour images, constructed by extracting all the prominent edge points in a painting. Hand-sketched queries are matched with database images by dividing the images into 8×8 blocks and computing a global correlation indicator. In STAR [12], both contour Fourier descriptors and moments invariants are used for shape representation and similarity measurement.

Several research papers tackled the trademark matching and retrieval issue using different approaches; invariant moments (Mohamad [13]), chain coding (Cortelazzo [14]), edge histograms and deformable template matching (Vailaya [15]). In the Artisan system [16], all images are segmented into components and geometric similarity features are extracted from boundaries of these components.

19.3. Shape Analysis

The word "shape" is very commonly used in everyday language, usually referring to the appearance of an object. More formally, a shape is: *all the geometrical information that remains when location, scale, and rotational effects are filtered out from an object* [17].

If one is interested in shape-based image retrieval, then obviously, color and texture will not play a useful role in distinguishing between various shapes. Consequently, binary images suffice for extracting good shape features. This Section presents a discussion of shape analysis in binary images. It particularly discusses each of the three shape properties suggested for use as a compact shape-description feature vector.

There are many properties that can be used in describing shapes. These include measurements of area and perimeter, length of maximum dimension, moments relative to the centroid, number and area of holes, area and dimensions of the convex hull and enclosing rectangle, number of sharp corners, number of intersections with a check circle and angles between intersections. All these measures have the property of characterizing a shape but not of describing it uniquely. Here we propose the usage of a set of object properties as shape features suitable for shape-based image retrieval. The ideas behind the use of those properties as shape features are fairly simple to grasp. We start by discussing each of the features and then we provide a discussion of the heuristics behind them.

19.3.1 The Solidity

The *solidity* of an object can be defined as the proportion of the pixels within the convex hull of the object that are also in the object [18]. It can be computed as the ratio between the object area and the area of the corresponding convex hull. Figure 19.3 shows two examples of object solidities. In the figure, for each of (a) and (b), the original image is shown on the left and the corresponding filled convex hull is shown on the right. The solidity value, computed as the area of the original image divided by the area of the filled convex hull image, is shown under the convex hull image. Figure 19.3 (a) shows a quite solid object;



Figure 19.3. Examples of object solidities. The solidity can be computed as the area of the object divided by the area of its convex hull. (a) Solid object. (b) Less-solid object.

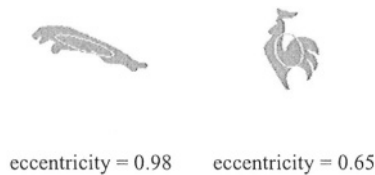


Figure 19.4. Examples of object eccentricities. The object on the left is more elongated, and hence has a larger eccentricity, than the object on the right.

the solidity is 0.86. On the other hand, a "not-quite-solid" object is shown in Figure 19.3 (b). This is reflected by a solidity of only 0.45.

19.3.2 The Eccentricity

The *eccentricity* of an object can be defined as the eccentricity of the ellipse representing the unit-standard-deviation contour of its points. If we view an object image as a set of points in 2-dimensional Cartesian space, then the parameters of the unit-standard-deviation ellipse are easily computed from the covariance matrix of the points [19]. The eccentricity of an ellipse is the ratio of the distance between the foci of the ellipse and its major axis length. The eccentricity is always between zero and one. (zero and one are degenerate cases; an ellipse whose eccentricity is zero is actually a circle, while an ellipse whose eccentricity is one is a line segment).

Figure 19.4 shows two examples of object eccentricities. We note that the object on the left is more elongated than the object on the right. This is reflected by a larger eccentricity for the object on the left. The unit-standard-deviation contours are shown superimposed on the images.

19.3.3 The Extent

The *minimum bounding rectangle* (MBR) of an object is the smallest rectangle that totally encloses the object. The *extent* of an object can be defined as the



Figure 19.5. Examples of object extents. The extent can be computed as the area of the object divided by the area of its minimum bounding rectangle (MBR). (a) Object with high extent value. (b) Object with smaller extent value.

proportion of the pixels within the minimum bounding rectangle of the object that are also in the object [18]. It can be computed as the object area divided by the area of the MBR. Figure 19.5 shows two examples of object extents (the objects here are again logo images). For both part (a) and part (b), the original objects are shown on the left, and the objects enclosed in their MBRs, along with the extent values, are shown on the right. Some basic shapes (a

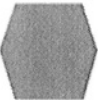
Image	S	C	X	Image	S	C	X
	0.99	0	0.79		1	0.29	1
	1	0.96	1		0.99	0.83	0.78
	0.99	0.45	0.73		0.99	0.40	0.85
	0.51	0.91	0.38		0.47	0.44	0.35

Figure 19.6. Some basic shapes and their corresponding solidity (S), eccentricity (C), and extent (X) values.

circle, a square, a rectangle, an ellipse, a pentagon, a hexagon, a star, and a crescent) are shown in Figure 19.6 along with their corresponding solidity, eccentricity, and extent values. We note that, as expected, the solidity of the first six shapes is either one or very close to one. Indeed, any *filled convex*

polygon will have a solidity value of one (the fact that the solidity value for the circle, ellipse, pentagon, and hexagon is 0.99 is because those objects are digitally represented by a finite set of pixels that approximates the geometry of the real object). The eccentricity values are also well in accordance with the shapes of the basic objects; the most elongated objects (the rectangle, the crescent, and the ellipse) have the highest eccentricity values, whereas the least elongated (the circle) has a zero eccentricity. As for the extent, its values are as well in accordance with the general shapes of the basic objects (a value of one for the square and the rectangle, a higher extent value for the hexagon than the pentagon, ellipse, or circle).

We note that these three features were able, on their own, to easily discriminate between the distinct shapes in Figure 19.6. It is worth noting though that the pentagon and the hexagon in the figure exhibit feature values that are pretty close. This is due to the fact that a hexagon and a pentagon can be considered quite similar shapes (although not identical of course).

19.4. Shape Retrieval Architecture

In this section, we present an architecture for shape-based image retrieval that can be used in conjunction with any feature set. However, we propose the use of a feature vector comprised of the solidity, eccentricity, and extent object properties. We will denote this feature set as the SCX feature set.

The main goal is to retrieve images from the database that closely match a hand-sketched shape image. The matching doesn't have to be based on the whole image. That is, the user-drawn sketch can match a part of a database image. We use the term "partial matching" to denote this process.

A strongly desired property of any shape-based image retrieval system is for the retrieval to be invariant to scale and rotation. generally, the features used are responsible for achieving this invariance. The SCX feature set, like many others, is invariant to scale and rotation.

The proposed image retrieval architecture encompasses many stages starting as early as the image preprocessing stage, going into the feature extraction stage, and ending with the image query stage. We will discuss each stage in detail in the following sections.

19.4.1 Preprocessing

Each image in the database is first pre-processed prior to the feature extraction process. The goal of the preprocessing phase is to appropriately prepare the image for the feature extraction. The three main tasks performed in the preprocessing are small-object removal, small-hole filling, and boundary smoothing. The first two operations are vital since we don't generally want any small spo-

radic group of pixels to be treated as an image component, nor do we want any too-small hole to be treated as an real hole. The last operation, object smoothing, is useful only in cases where the object boundaries are horizontal or vertical. It basically removes any parasitic spurs from the object binary image.

19.4.2 Feature Extraction

In describing the feature extraction stage of the architecture, we will use the SCX feature set as our backbone feature vector. However, it should be kept in mind that any other feature set can be used.

In the proposed architecture, an image is transformed not to a single feature vector as is usually the case, but to several feature vectors all referring to the image of interest.

Figure 19.7 shows the steps required to transform a binary image depicting a shape to the feature space. An image is transformed to several feature vectors (or database records), The first record V_i pertains to image i as a whole. Then, L records, v_{ij} $j = 1 \cdots L$, follow the first one (L is the number of components in the image). Each of those L records represents to a component of the image.

An example would help clarify this stage. Consider the three images shown in Figure 19.8. The images on the left and in the middle contain two components and one hole each, whereas the one on the right contains three components and two holes. Table 19.1 shows how they would be represented in the database.

The left-most column in the table contains the image number. In database terminology, This number acts as a "foreign key" referring to the "primary key" in another table containing unique image numbers and file names. The second column is a whole-image flag; it has a value of one for database records that describe whole images, and a value of zero for database records describing image components. There are two uses for this column (also a database field). The first is to distinguish partial matches from whole-image matches, and the second is to turn on or off the partial-match system feature. The third column is the number-of-components (N) feature. It always has a value of one for components. The fourth column is the number-of-holes (H) feature. The last six columns are the solidity (S), eccentricity (C), and extent (X) of components and holes.

For whole images, S , C , and X values are the solidity, eccentricity, and extent of the image as a whole, but with any holes filled, and S' , C' , and X' are the solidity, eccentricity, and extent of the filled holes in the image treated as one object. For image components, on the other hand, S , C , and X values are the solidity, eccentricity, and extent of the filled component, and, S' , C' , and X' are the solidity, eccentricity, and extent of the filled holes in that component treated as one object (refer to the algorithm in Figure 19.7).

Feature extraction from a binary image database

Procedure PopulateDatabase

```

foreach image  $Y_i$  in the database,  $i = 1, 2, \dots, N$ 
    {Segment  $Y_i$  into its  $L_i$  components using connected components labeling }
    foreach component  $Z_j$  in  $Y_i$ ,  $j = 1, 2, \dots, L_i$ 
         $M_j \leftarrow$  number of holes in component  $Z_j$ 
         $S_j \leftarrow$  solidity of filled component
         $C_j \leftarrow$  eccentricity of filled component
         $X_j \leftarrow$  extent of filled component
         $H_j \leftarrow$  image whose pixels are the set of all pixels in the holes of  $Z_j$ 
        {Fill all holes in  $H_j$  }
         $S'_j \leftarrow$  solidity of the set of all pixels in  $H_j$ 
         $C'_j \leftarrow$  eccentricity of the set of all pixels in  $H_j$ 
         $X'_j \leftarrow$  extent of the set of all pixels in  $H_j$ 
        {form vector  $v_{ij} = (1, M_j, S_j, C_j, X_j, S'_j, C'_j, X'_j)$  }
    endfor
     $F_i \leftarrow Y_i$  with all holes filled
     $S_i \leftarrow$  solidity of the set of all pixels in  $F_i$ 
     $C_i \leftarrow$  eccentricity of the set of all pixels in  $F_i$ 
     $X_i \leftarrow$  extent of the set of all pixels in  $F_i$ 
     $H_i \leftarrow$  image whose pixels are the set of all pixels in the holes of  $Y_i$ 
    {Fill all holes in  $H_i$  }
     $S'_i \leftarrow$  solidity of the set of all pixels in  $H_i$ 
     $C'_i \leftarrow$  eccentricity of the set of all pixels in  $H_i$ 
     $X'_i \leftarrow$  extent of the set of all pixels in  $H_i$ 
     $M_i \leftarrow \sum_{j=1}^{L_i} M_j$ 
    {form vector  $V_i = (L_i, M_i, S_i, C_i, X_i, S'_i, C'_i, X'_i)$  }
    {vector  $V_i$  and vectors  $v_{ij}$ ,  $j = 1, 2, \dots, L_i$ , are added to the database }
endfor
endProcedure

```

Figure 19.7. Steps for populating the feature database.

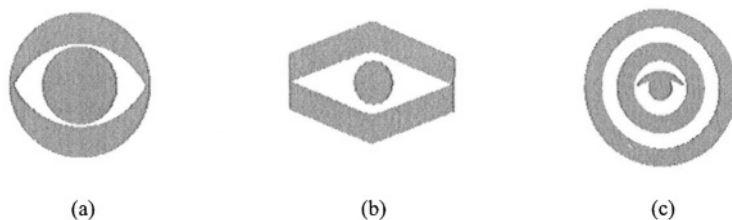


Figure 19.8. Feature extraction example.

19.4.3 The Matching Process

Querying-by-example is the most common way to search image databases by content [20]. In the case of shape-based image retrieval, a hand-sketch of the sought shape is an adequate means for querying, and we indeed use it. In the proposed system, the matching procedure is as follows:

- 1 The query image is first pre-processed (see Section 19.4.1). It is worth noting here that, for the sake of simplifying the system, the hand-sketched query image is assumed to be filled and well sketched (in the sense that it has to be closed if it depicts a closed shape).
- 2 The whole-image feature vector V is extracted from the image. The choice of only extracting V as opposed to also extracting the components feature vectors v_j 's depends on what we are interested in. If we want to retrieve database images that match a *part* of the query image, then there are two approaches; either to extract the v_j 's and use them in a multi-query-image fashion (which will complicate the query), or to simply

Table 19.1. Feature values corresponding to images in Figure 19.8.

Image number	Whole-image flag	N	H	S	C	X	S'	C'	X'
1	1	2	1	0.99	0.18	0.78	0.40	0.86	0.29
1	0	1	1	0.99	0.18	0.78	0.99	0.75	0.71
1	0	1	0	0.99	0.21	0.78	0	0	0
2	1	2	1	0.99	0.78	0.73	0.50	0.96	0.27
2	0	1	1	0.99	0.78	0.73	0.98	0.92	0.54
2	0	1	0	0.98	0.21	0.78	0	0	0
3	1	3	2	0.99	0.22	0.78	0.58	0.22	0.45
3	0	1	1	0.99	0.22	0.78	0.99	0.22	0.78
3	0	1	1	0.99	0.23	0.78	0.98	0.23	0.78
3	0	1	0	0.79	0.59	0.53	0	0	0

sketch only the part that we are interested in from the beginning and use it as the query image.

- 3 (Optional step) Restrict the database to records containing values close to the N and H values of the query image. For example, if for the query image, $N = 3$ and $H = 2$, then we might choose to restrict the database to records where $N \in \{2, 3, 4\}$ and $H \in \{1, 2, 3\}$. This is equivalent to indexing and will help speed up the retrieval when the database gets larger.
- 4 The Euclidean distance was chosen as a distance measure for retrieving the closest images from the database because of its simplicity (other measures can be investigated/used as well). The matching is based on the feature vector $(N, H, S, C, X, S', C', X')$. Depending on whether partial matching is turned on or off (using the whole-image-flag field), some of the retrievals might be based on the query image matching part of the database image.

19.4.4 Enhancing the Matching Process

To add more power to partial matching, the following modifications were introduced to the architecture. Each image in the database is represented by three aliases: the original image, a dilated version, and an eroded version. A new field is added to the database to denote whether the image is the original one or the dilated/eroded version. This flag will be mainly used in order to enable or disable object matching based on dilated/eroded versions of database images. It should be mentioned that a dilated/eroded image is *only* added to the database if the operation changes the number of components or holes in the image. A square 5×5 structuring element is repeatedly used in the morphing until the number of components or holes change.

The above mentioned modifications have two positive effects (discussed in the following two subsections):

- 1 Enhancing whole-image matching.
- 2 Enhancing partial matching.

On the other hand, the disadvantage is a much larger number of records (an upper limit of about three times the original number). This should not degrade the speed of the retrieval since the indexing mechanism alleviates that. A larger number of records might, however, affect the accuracy of the retrieval. Whether the advantages outweigh the disadvantages is the decisive factor in whether to opt for this modification or not (we can always, for each query image, perform two queries, one with the dilation/erosion enabled and one with it disabled).

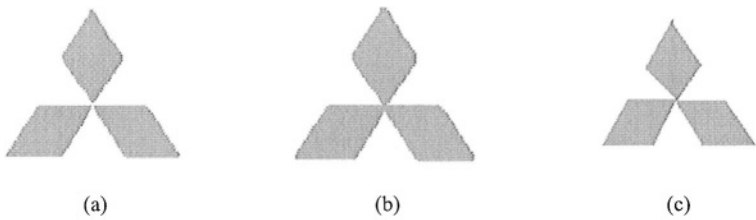


Figure 19.9. Enhancing whole-image matching. (a) original image. (b) Dilated image. (c) Query image.

Table 19.2. Feature values for the image in Figure 19.9.

Image number	Morph. flag	Whole-image flag	N	H	S	C	X	S'	C'	X'
123	0	1	3	0	0.62	0.23	0.32	0	0	0
123	0	0	1	0	0.95	0.83	0.53	0	0	0
123	0	0	1	0	0.91	0.74	0.52	0	0	0
123	0	0	1	0	0.95	0.84	0.55	0	0	0
123	1	1	1	0	0.77	0.23	0.44	0	0	0

19.4.5 Enhancing Whole-image Matching

Consider the image shown in Figure 19.9 (a), its dilated version shown in Figure 19.9 (b), and the query image shown in Figure 19.9 (c). Obviously, according to the discussion in Section 19.4.2, the query image (c) will not match image (a) but will match image (b) (if it existed in the database). This is precisely why including image (b) as an alias for the original image (a) in the database will help retrieve the correct image. We should note that image (b) will not by physically added to the database; only a feature vector (or a number of feature vectors) will represent it in the database. This (these) vector(s) will "point" (using the image number field) to the original image. Table 19.2 shows the feature vectors as they would appear in the database. The second column in the Table 19.2 is a flag that has the value of one for morphologically modified (dilated or eroded) images and a value of zero for original images. We note that since there are no holes in the image in Figure 19.9 (a), S' , C' , and X' all have a value of zero. In addition, since erosion will *not* change the number of components in the image, there are no new records showing a value of N greater than 3.

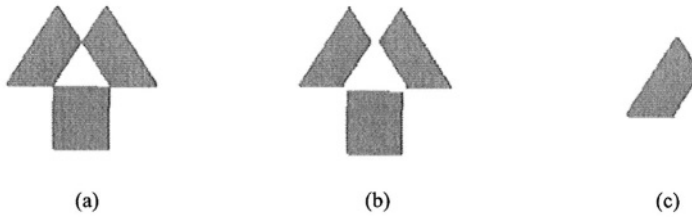


Figure 19.10. Enhancing partial matching. (a) original image. (b) Eroded image. (c) Query image.

19.4.6 Enhancing Partial Matching

Similar to the previous discussion, consider the image shown in Figure 19.10 (a), its eroded version shown in Figure 19.10 (b), and the query image shown in Figure 19.10 (c). Again, according to the discussion in Section 19.4.2, the query image (c) will not partially match image (a) but will partially match image (b) (if it existed in the database). This is precisely why including image (b) as an alias for the original image (a) in the database will help retrieve the correct image.

19.4.7 Discussion

The choice of the solidity (S), eccentricity (C), and extent (X) shape measures (or features) is not a necessary condition. One can freely substitute them with other measures. Alternatively, other measures can be added to them to form an extended feature vector. However, the use of the SCX feature vector has some interesting properties. As discussed in Section 19.3, they are invariant to rotation, position, and scale change. They all have a value range from zero to one; a decent range for features that does not need further normalization in general.

As a matter of fact, the solidity, eccentricity, and extent do not uniquely describe a given object; one can easily come up with two different objects with the same values for the three features. However, the goal is to identify "similar" objects and not to uniquely describe them. That is, two objects with close solidity, eccentricity, and extent values are in general similar.

19.5. Experimental Results

We used a database of 500 logo images to test the performance of the system. The bigger part of the image database used in this work was obtained from the department of Computer Science at Michigan State University. The original

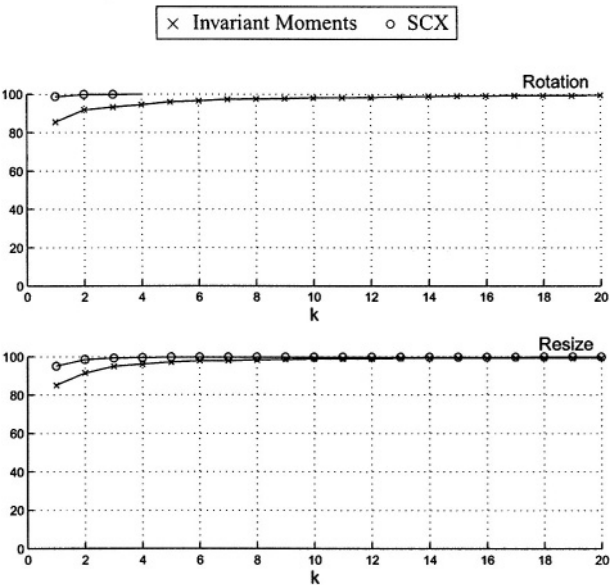


Figure 19.11. Rotation and scale invariance of the SCX feature set compared with the moments invariants. The vertical axis is the percentage of the images in the database that were in the first k (horizontal axis) retrievals.

database consisting of 1100 images was used by A. K. Jain and A. Vailaya in an image retrieval case study on trademark images cite15. According to Jain and Vailaya, the database used was created by scanning a large number of trademarks from several books at 75 dpi. In our research, we used a subset of this 1100-image database, as well as some logos collected over the Internet, for a total of 500 images. The reason of using a subset of those images is that we are primarily concerned with shape-based image retrieval, and therefore, we confined the experiments to image whose main characteristic is shape. Image with texture nature were automatically removed from the analysis based on their number of components and/or holes (texture images have a large number of components and/or holes).

19.5.1 Invariance to Rotation and Scale Change

In order to test the capability of the solidity, eccentricity, and extent (SCX) shape measures to describe objects regardless of their spatial orientation or scale, two tests were performed. In the first test, each of the 500 images in the database was randomly rotated and used as a query image. In the second test, the images were randomly scaled. The angles of rotation uniformly varied between 30° and

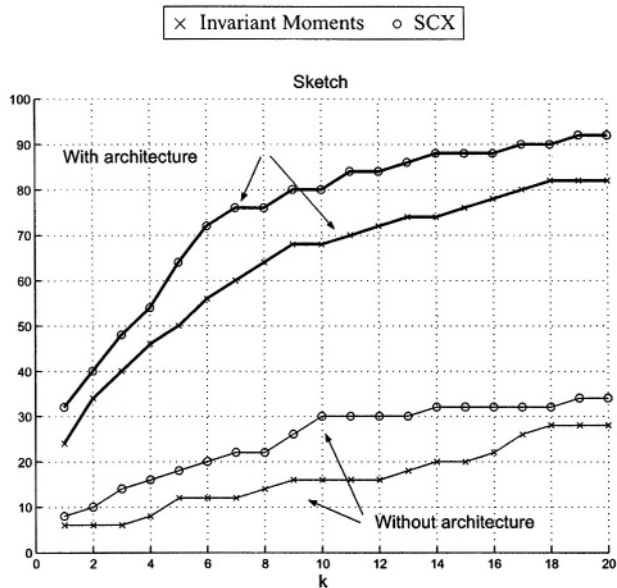


Figure 19.12. Performance for sketch-based retrieval of the SCX feature set compared with the moments invariants. The vertical axis is the percentage of the images in the database that were in the first k (horizontal axis) retrievals.

330°, whereas the scaling factor uniformly varied between $\times 0.25$ and $\times 1.75$. The SCX feature set was compared against the widely used moments invariants feature set. The seven invariant moments as well as the SCX features are extracted from the object image as a whole. That is, the proposed architecture was not used in this test because the purpose was to test the rotation and scale invariance of the feature sets used. The Euclidean distance measure is used in both cases.

Figure 19.11 shows two plots, the lower pertains to the scale invariance test and the upper to the rotation invariance test. In each plot, the vertical axis is the percentage of the images in the database that were in the first k (horizontal axis) retrievals. In the lower plot for example, about 97% of the 500 query images returned the correct database image as the first hit (for the SCX feature set). The plots in Figure 19.11 are based on the average of 20 runs.

From the plots, it is clear that for both rescaled and rotated images, the performance of the SCX feature set and the performance of the moments invariants closely match each other. In addition, both feature sets perform well in terms of being invariant to rotation and scale.

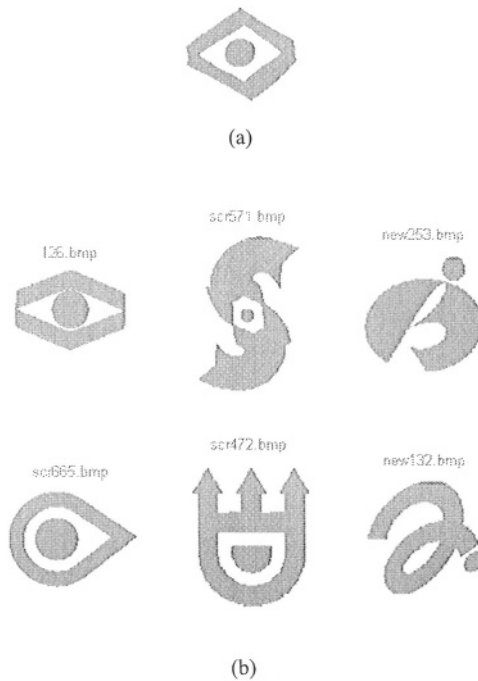


Figure 19.13. Sample query (1). (a) Query image. (b) Best matches starting from top left and proceeding to the right.

19.5.2 Sketch-based Retrieval

To test the performance of the system on hand-sketched images, we used the same test procedure as in the previous section, but now using a test set of 50 hand-sketched images. Five volunteers participated in the test, each sketching his/her own version of the selected 50 images. Moreover, now we compare the performance of the two feature sets both inside and outside the proposed architecture.

Figure 19.12 clearly indicates that the performance of the SCX feature set coupled with the proposed retrieval architecture yield the best results. In conclusion, the SCX feature set outperforms the moments invariants for hand-sketched retrieval and the proposed architecture significantly increases the performance of both feature sets.

The reason for the performance gap between the SCX and moments is that moments tend to be low level features that are sensitive to changes in pixel

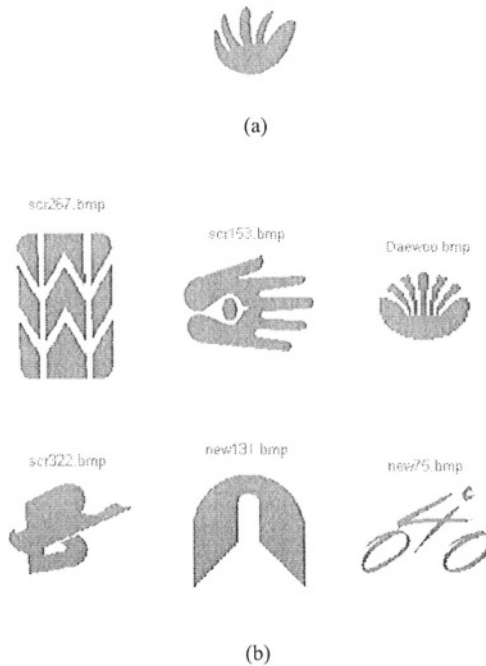


Figure 19.14. Sample query (2). (a) Query image. (b) Best matches starting from top left and proceeding to the right.

layouts. A good example of this layout difference is the one that exists between database images and their hand-sketched versions.

Figures 19.13 and 19.14 show a couple of queries and the best six retrieved images in each case. In each figure, part (a) shows the query image and part (b) shows the retrieved images (top left is the best match, top mid is the second-best match, and so forth). In Figures 19.13, the first candidate returned by the system was the one sought. In Figures 19.14, on the other hand, the sought image was the third candidate returned by the system. However, the first and second candidates returned by the system are a good example of partial matching (where the query image matches a component of the database image). A good example of how an erosion operation helps retrieve the correct candidate is shown in Figure 19.13 where the first candidate, containing one component, was retrieved as an answer to a query image containing two components.

19.6. Conclusion and Future Work

In this paper, a shape-based image retrieval system was presented. Both the architecture of the system and the features used contribute to the efficacy of the system. The feature vector used is based on the solidity, eccentricity, and extent values for components and holes in the image under investigation. The features used are not the only choice. One can add to, or modify, the feature vector.

We showed that the performance of the proposed architecture combined with the features used matches that of the widely-used moments invariants feature set in retrieving rotated and scaled query images, and surpasses it for hand sketched query images.

We have assumed throughout this work that the hand-sketched query images are drawn with closed contours and that they are appropriately filled. A prospective research direction might be to look at ways of relaxing these assumptions by automatically processing the hand-sketched image to put it in a suitable form.

A natural extension of the system would be to store information not only about single image parts, but also about n parts at a time. That is, each part is analyzed on its own and features are extracted from it, then each possible two-parts combination is analyzed and features are extracted from it, and so on.

In this work, we have not attempted to use suitable spatial indexing structures. However, such structures become essential when it comes to large databases.

References

- [1] R. C. Gonzales and R. E. Woods. *Digital image processing*. Addison-Wesley, 1992.
- [2] M. Safar, C. Shahabi and X. Sun. "Image retrieval by shape: a comparative study". In *Proc. of International Conference on Multimedia and Expo*, vol. 1, pp. 141-144, 2000.
- [3] M. Safar, C. Shahabi and C. Tan. "Resiliency and robustness of alternative shape-based image retrieval techniques". In *Proc. of International Database Engineering and Applications Symposium*, pp. 337-345, 2000.
- [4] G. Lu and A. Sajjanhar. "Region-based shape representation and similarity measure suitable for content-based image retrieval". *Multimedia Systems*, vol. 7, pp. 165-174, 1999.
- [5] K.W. Hung and M. Aw-Yong. "A content-based image retrieval system integrating color, shape and spatial analysis". In *Proc. International Conference on Systems, Man, and Cybernetics*, vol. 2, pp. 1484-1488, 2000.
- [6] J. Xu. "Morphological representation of 2-D binary shapes using rectangular components". In *Proc. International Conference on Image Processing*, vol. 2, pp. 862-866, 1999.
- [7] M. Adoram and M. S. Lew. "Image retrieval using shape". In *Proc. International Conference on Multimedia Computing and Systems*, vol. 2, pp. 597-602, 1999.

- [8] H. Kim, J. Kim, D. Sim and D. Oh. "A modified Zernike moment shape descriptor invariant to translation". In *Proc. International Conference on Multimedia and Expo*, vol. 1, pp. 307-310, 2000.
- [9] S. Muller and G. Rigoll. "Improved stochastic modeling of shapes for content-based image retrieval". In *Proc. Workshop on Content-Based Access of Image and Video Libraries*, pp. 23-37, 1999.
- [10] M. Flickner et al. "Query by image and video content: The QBIC system". *IEEE Computer*, pp. 23-32, September, 1995.
- [11] T. Kato, T. Kurita and H. Shimogaki. "Intelligent Visual interaction with image database". *Journal of information processing of Japan*, vol. 12, no. 2, pp. 134-143, 1991.
- [12] C. P. Lam, J. K. Wu and B. Mehtre. "STAR-a system for trademark archival and retrieval". In *Proc. 2nd Asian Conference on Computer Vision*, vol. III, pp. 214-217, 1995.
- [13] D. Mohamad, G. Sulong and S. Ipson. "Trademark matching using invariant moments". In *Proc. 2nd Asian Conference on Computer Vision*, vol. I, pp. 439-444, 1995.
- [14] G. Cortelazzo, G.A. Mian, G. Vezzi and P. Zamperoni. "Trademark shapes description by string matching techniques". *Pattern Recognition*, vol. 27, no. 8, pp. 1005-1018, 1994.
- [15] A. K. Jain and A. Vailaya. "Shape-based retrieval: a case study with trademark image database". *Pattern Recognition*, vol. 31, no. 9, pp. 1369-1390, 1998.
- [16] J. P. Eakins, J. M. Boardman and M. E. Graham. "Similarity retrieval of trademark images". *IEEE Multimedia*, pp. 53-63, April-June, 1998.
- [17] D. G. Kendall. "The diffusion of shape". *Advances in Applied Probability*, vol. 9, pp. 428-430, 1977.
- [18] *Image Processing Toolbox for use with MATLAB*, The Mathworks, 1997.
- [19] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural, and Neural Approaches*. J. Wiley, 1992.
- [20] V. N. Gudivada and V. V. Raghavan. "Content-based image retrieval systems". *IEEE Computer*, pp. 18-22, September, 1995.

Chapter 20

MULTI-RESOLUTION IMAGE REGISTRATION USING MULTI-CLASS HAUSDORFF FRACTION *

Haikel Salem Alhichri and Mohamed Kamel*

*Department of Systems Design Engineering, The University of Waterloo, Waterloo,
Ontario N2L 3G1, Canada
{hhichri,mkamel}@uwaterloo.ca*

Abstract Recently, a new image registration method, based on the Hausdorff fraction and a multi-resolution search of the transformation space, has been developed in the literature. This method has been applied to problems involving translations, translation and scale, and Affine transformations. In this paper, we adapt the above method to the set of similarity transformations. We also introduce a new variant of the Hausdorff fraction similarity measure based on a multi-class approach, which we call the Multi-class Hausdorff Fraction (MCHF). The multi-class approach is more efficient because it matches feature points only if they are from the same class. To validate our approach, we segment edge maps into two classes which are the class of straight lines and the class of curves, and we apply the new multi-class approach to two image registration examples, using synthetic and real images, respectively. Experimental results show that the multi-class approach speeds up the multi-resolution search algorithm.

Keywords: Image registration, Hausdorff fraction, multi-resolution, branch-and-bound

20.1. Introduction

Image registration is the process of determining the transformation which best matches, according to some similarity measure, two images of the same scene taken at different times or from different view points. Recently, there has been interest in image registration methods based on the *Hausdorff distance* and the *multi-resolution search* of the transformation space [2] [3] [1] [4]. With

*This chapter is a reprint of the paper in: H.S. Alhichri and M. Kamel. "Multi-resolution image registration using multi-class Hausdorff fraction". *Pattern Recognition Letters*, vol. 23, pp. 279-286, 2002.

*M. Kamel is a professor with the University of Waterloo and is the corresponding author.

suitable modifications described in [2], the Hausdorff distance is tolerant to noise and outliers. The *Hausdorff fraction* is a variant of the Hausdorff distance that can be efficiently computed. The *multi-resolution* approach uses *branch-and-bound* techniques to prune large sections of the transformation space.

The method developed by Huttenlocher *et al.* [2] works well and is computationally efficient for the translation space, but is significantly time consuming under Affine transformations. Olson *et al.* [5] enhance the above method by incorporating edge point orientations in the Hausdorff distance definition, which reduces the rate of false-alarm and enhances performance. However, their method requires costly preprocessing which should be done off line. For example, such preprocessing includes the computation of many distance transform arrays, one for each set of oriented edge points. Mount *et al* [6], combine the multi-resolution approach with alignment approach. This is because at later stages of the search, the uncertainty regions of edge points become small and hence the set of possible alignments also becomes small. However, it must be noted that this combinations, which produced significant speedups of the algorithm, can still be applied to our algorithm. Yi and Camps [7] [8] use the length (actually $\log(\text{length})$) and the direction of line features to transform scale and rotation differences into translation differences. Their method also makes it possible to search the scale-rotation and translation spaces independently with lower complexity. However, it is not general because it relies on the existence of a good number of straight line segments and a robust and efficient algorithm to extract them.

In this work we propose an image registration method similar to the method in [1], which is based on the following four components:

- Feature points : we use edge points as feature points.
- Similarity measure : we defined a variant of the Hausdorff fraction similarity measure based on a multi-class approach, which we call the *Multi-class Hausdorff Fraction (MCHF)*.
- Search space : we adapt the method to the class of *similarity* transformations consisting of scale, rotation and translation differences.
- Search strategy : we use a multi-resolution search of the transformation space with branch-and-bound techniques.

During the search of the transformation space, the multi-class approach visits a lesser or at least equal number of cells than the single-class approach. To validate our claims in practice, we propose to segment edge maps into two classes which are the class of straight lines and the class of curves. This segmentation is invariant to rotation and moderate scale. During the space search step, edge points in the model that belong to one class are only compared with edge points

in the scene that are of the same class. In other words, the Hausdorff fraction is computed between straight lines and curved lines separately and the final fraction is taken as their weighted sum.

Another interesting idea is to classify edge point into several classes based on their orientation, which would be similar, in a way, to the work in [5]. However, it is important to mention that there is a tradeoff between the number of classes on the one hand and the speed and robustness on the other hand. First, the more classes we have, the higher are the overhead costs of the algorithm. For example the distance transform would have to be computed for each class image. The cost of each distance transform is made even worse by the fact that the edge points are now more sparse. Thus it is possible that the preprocessing costs become too high to the extent that they nullify the savings achieved during the search step. Second, more classes means more classification errors, causing an correct match to be missed in some instances. For these reasons we chose to limit the number of classes to two in this work.

The rest of the paper is organized as follows, in Section 20.2 we introduce briefly the Hausdorff fraction. In Section 20.3, we stuke the transformation space. Next, in Section 20.4, We present an overview of the multi-resolution image registration algorithm and we show how the method is adapted to similarity transformations. We discuss the new multi-class Hausdorff fraction in Section 20.5. Section 20.6 contains two experiments implementing the multi-class Hausdorff fraction using two classes. Finally, Section 20.7 contains conclusions and future work.

20.2. Hausdorff Fraction

The Hausdorff fraction is a similarity measure between point sets which is tolerant to noise and occlusions. Given two point sets $P = \{p_1, \dots, p_N\}$ and $Q = \{q_1, \dots, q_M\}$, the Hausdorff fraction between P and Q is defined as:

$$HF^\tau(P, Q) = \frac{\#\{p \in P \mid \min_{q \in Q} \|p - q\| < \tau\}}{\#(P)} = \frac{n}{N} \quad (20.1)$$

In words it means: find the fraction of points in P , which are close to a point in Q , within a threshold distance τ .

It can be seen, that the Hausdorff fraction is noise-tolerant up to a noise threshold of τ , since any such noise in the location of points does not affect the match. It is also able to handle outliers and extra points, given the reasonable assumption that the Hausdorff fraction of match is still maximum at the correct transformation.

20.3. Transformation Space

The transformation space can be the set of translations t_x and t_y , which are the horizontal and vertical translations respectively. The more general Affine transformation space has 6 dimensions and is defined as follows:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (20.2)$$

Where (x, y) are the original coordinates of the point and (x', y') are the transformed coordinates. In general the parameters a, b, c , and d are continuous, while t_x and t_y are discrete because the images are digitized.

In this work, the *similarity* transformation space is used which has 4 parameters: t_x, t_y, α , and θ which are the translation in x , translation in y , scale and rotation parameters, respectively. The relationship between the scale and angle parameters and the Affine parameters a, b, c , and d is as follows:

$$\begin{aligned} a &= d = \alpha \cos(\theta) \\ b &= -c = -\alpha \sin(\theta) \end{aligned} \quad (20.3)$$

20.4. Multi-resolution Image Registration

The multi-resolution approach for image registration is presented in [3] and [1]. Recall that we are given two point sets P and Q and a similarity space of transformations S . The problem is to find $T \in S$ that maximizes the Hausdorff fraction $HF^\tau(T(P), Q)$. Since P and Q will be fixed for the remainder of the discussion, let us define $HF^\tau(T) = HF^\tau(T(P), Q)$ as the Hausdorff fraction of T . Let T_{opt} denote this optimum transformation and let HF_{opt}^τ be the optimum Hausdorff fraction.

The multi-resolution algorithm constructs a search tree, where each node is identified with the set of transformations contained in some axis-aligned hyper-rectangle called a *cell*. Each cell is bounded by two transformations $T_{lo} = (t_{x,lo}, t_{y,lo}, \alpha_{lo}, \theta_{lo})$ and $T_{up} = (t_{x,hi}, t_{y,hi}, \alpha_{up}, \theta_{up})$. The cell at the root of the tree is given by the user based on a priori knowledge about the problem. It is assumed that T_{opt} belongs to this initial cell. An illustration of the initial cell is shown in Figure 20.1, where t_x and t_y are fixed, and where the Affine parameters a and b are represented by the horizontal and the vertical axis, respectively. The actual cell is the area between the two circles with radius α_{lo} and α_{up} , and by the two rotation angles θ_{lo} and θ_{up} . Figure 20.1 also shows an example of the cell subdivision process, where each cell is subdivided into two sub-cells which represent its children.

The multi-resolution algorithm uses a branch-and-bound technique to prune branches from the search tree. The tree is searched in a best-first order using

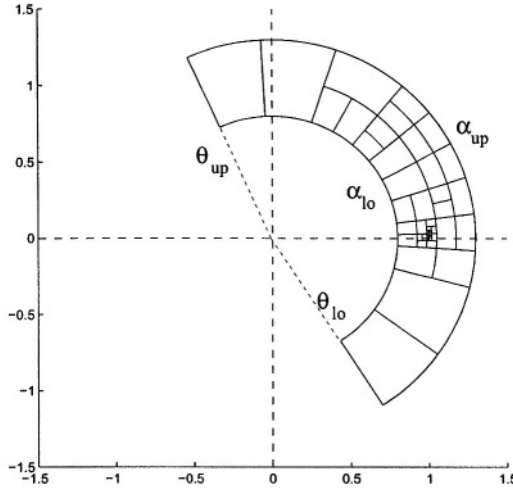


Figure 20.1. Subdivision of the scale-angle transformation space. The x and y axes represent the Affine parameters a and b , respectively.

a priority queue. When a node is visited, the algorithm computes two bounds on the Hausdorff fraction for the cell at that node. The lower bound can be the Hausdorff fraction at any transformation T belonging to the cell, in particular $HF^\tau(T_{mid})$, where T_{mid} is the transformation at the center. It is clear that we have $HF^\tau(T_{mid}) \leq HF_{opt}^\tau$. Note also that this value is used as a key by which the cells are ordered in the priority queue. The upper bound HF_{up}^τ on the optimal Hausdorff fraction HF_{opt}^τ can help us kill an entire cell and hence a branch of the search tree. This occurs if $HF_{up}^\tau < HF_{opt}^\tau$, because it means that the current cell can not contain the optimal transformation. Notice however, that HF_{opt}^τ is unknown, therefore we need to use HF_{best}^τ instead, which is the best Hausdorff fraction found so far.

The upper bound HF_{up}^τ is computed as the number of model points that can be matched by some transformation in the current cell. As in [1], this can be computed with the help of the *uncertainty region* $R_i(p'_i)$ of a given model point p_i , which is defined as the area in the scene covered by $p'_i = T(p_i)$ for all T in the current cell. If a scene point is contained in the region $R_i(p'_i)$, then this means that the model point p_i can be matched. Thus p_i is counted in the upper bound HF_{up}^τ .

However, the area covered by p'_i has an irregular shape as shown in Figure 20.2. First of all, for a fixed scale and rotation values, the area covered by a such a point under the range of translations is rectangular. Now, the total area covered by p'_i under similarity transformation is the result of sliding the corner

of the rectangular region over the range of scales and rotations (which is an arc-like shaped area). This result is illustrated in Figure 20.2 (a). In order to decide

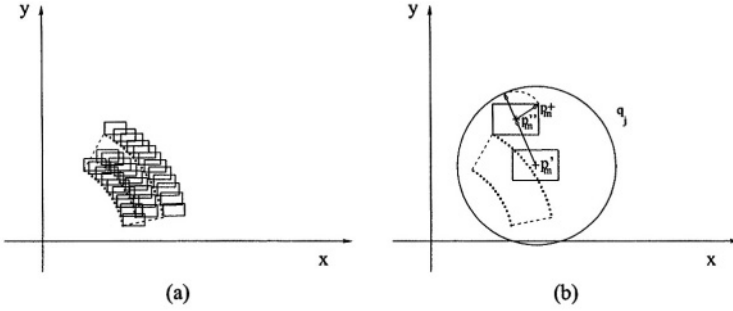


Figure 20.2. (a) Illustration of irregular uncertainty region for a cell of similarity transformations. (b) Approximation of uncertainty region size.

whether or not p_i can be matched, we need to find whether or not some point q_j falls inside the uncertainty region $R_i(p_i')$. Let $p_m' = T_{mid}(p_i)$, where T_{mid} is the center transformation of the cell. Recall that the distance transform gives us the distance $\Delta(p_m')$ from p_m' to its nearest neighbor in the scene q_j . Thus, if $\Delta(p_m')$ is less than the *uncertainty region size*, it means p_i can be matched and vice versa. We define the uncertainty region size as $|p_m'p_m''| + |p_m''p_m^+|$, where the points p_m'' and p_m^+ are defined as shown in Figure 20.2 (b), or respectively:

$$p_m'' = \alpha_1 R(\theta_1)p_i + \frac{1}{2}t$$

$$p_m^+ = \alpha_1 R(\theta_1)p_i + t$$

and where $R(\theta_1)$ is the rotation matrix and t is the translation vector. It is easy to show that this uncertainty region size is greater than or equal to the distance to the farthest point in the uncertainty region.

20.5. Multi-class Hausdorff Fraction

The *multi-class Hausdorff fraction* $MCHF^\tau(P, Q)$ is defined as follows:

$$\begin{aligned} MCHF^\tau(P, Q) &= \sum_{i=1}^L \frac{N_i}{N} HF_i^\tau(P_i, Q_i) \\ &= \sum_{i=1}^L \frac{\#(\{p \in P_i | \min_{q \in Q_i} \|p - q\| < \tau\})}{N} \\ &= \sum_{i=1}^L \frac{n_i}{N} \end{aligned} \quad (20.4)$$

where $N_i = \#(P_i)$, $HF_i^T(P_i, Q_i)$ is the Hausdorff fraction between feature points of the same class i . It is easy to show that the number of cells visited by the multi-class technique is guaranteed to be less than or equal to those visited by the single-class approach. This is because all points which are matched in the single-class approach, may become unmatchable if they are from different classes as shown in Figure 20.3 (a).

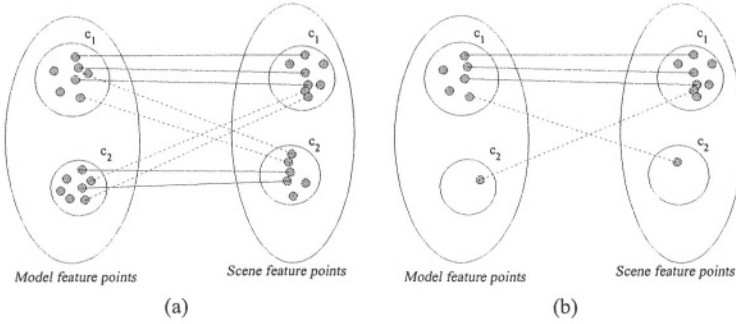


Figure 20.3. (a) A two class example. Solid lines match feature points from the same class. Dashed lines match feature points from different classes, which may not be matchable with a point from the same class. (b) Extreme case of two classes of size $N - 1$ and 1.

Furthermore, more cells are killed when the set of feature points is equally divided between classes (i.e. the classes have similar size) because $MCHF_{up}^T$ is reduced most. To illustrate this, consider the simple case of two classes where one class is of size 1 and the other class is of size $N - 1$. In this case, it is clear that the numerator of $MCHF_{up}^T$ is less than the numerator of HF_{up}^T by at most 2 as illustrated by Figure 20.3 (b), which means any extra pruning will be minimal.

20.6. Experimental Results

In this section, we apply the multi-resolution multi-class algorithm to two image registrations experiments, one involving synthetic images and one involving real images of a circuit board. We propose to segment edge maps into two classes which are the class of straight lines and the class curves. There are algorithms to perform this segmentation such as the *Hough* transform. However, we did not investigate this part in this work, instead we segment the images into straight lines and curves manually.

The first experiment consists of synthetic images shown in Figure 20.4, where straight lines and curves are illustrated with different gray levels. The initial cell given for this experiment is $[(3, 1, 0.5, -1.5) \rightarrow (5, 3, 1.3, 1)]$. The correct transformation is $(4, 2, 1.0, 0)$ with Hausdorff fraction 100%. The target Haus-

dorff fraction is set at 100% and the algorithm does not stop when the target is reached, instead it continues searching all the space.

As was shown in Section 20.5, more cells are killed early on allowing the search to reach the correct transformation quicker. To illustrate this result, we have plotted all the center transformations T_{mid} visited by the algorithm under the single-class approach and the multi-class approach in Figure 20.5 (a) and (b) respectively. It can be seen that the multi-class approach visited much less cells than the single class approach. In the multi-class approach, many cells are killed early on reducing the number of cells visited, and directing the search elsewhere in the space where the Hausdorff fraction is higher.

The same experiment is repeated for another set of images this time from a real example involving registration of a circuit board scene with a model for inspection purposes. Figures 20.6 and 20.7 show the images and the results for this experiment. The initial cell given to the algorithm is $[(238, 238, 1.00, -1.5) \rightarrow (242, 242, 1.50, 0)]$, while the optimal transformation is $(238, 241, 1.21, -0.74)$. [htbp]

We also include the exact results for both experiments in Table 20.1. The table

Table 20.1. Summary of results for both synthetic and real examples.

		Cells visited	CPU time
Experiment 1	single class approach	4009	12.9 s
Experiment 1	multi-class approach	1302	3.9 s
Experiment 2	single class approach	1809	7.1 s
Experiment 2	multi-class approach	1545	6.9 s

shows how the total number of cells visited is less in the multi-class approach for both experiments, which is translated into savings in CPU execution time for both experiments. However, note that the savings in the experiment involving real images are less significant. This is due to the discrepancies between the segmentation results of the model and the scene as well as larger differences in size between the two classes in the case of real images.

20.7. Conclusion

In this paper, we adapt the multi-resolution method, developed previously in the literature for the Affine set of transformations, to the set of similarity transformations. We also introduce a new variant of the Hausdorff fraction similarity measure based on a multi-class approach, which we call the Multi-class Hausdorff Fraction (MCHF). The multi-class approach is more efficient because it visits a lesser or equal number of cells than the single-class approach.

The drawback to such techniques is that some pixels can be misclassified and thus an correct match can be missed in some instances.

To validate our claims in practice, we have proposed to segment edge maps into two classes which are the class of straight lines and the class curves. Experimental results using two image registration examples, one involving synthetic images and one involving real images, have shown that the new multi-class approach visited significantly less cells during the search of the transformation space. It has also shown that, even though the new approach requires two more

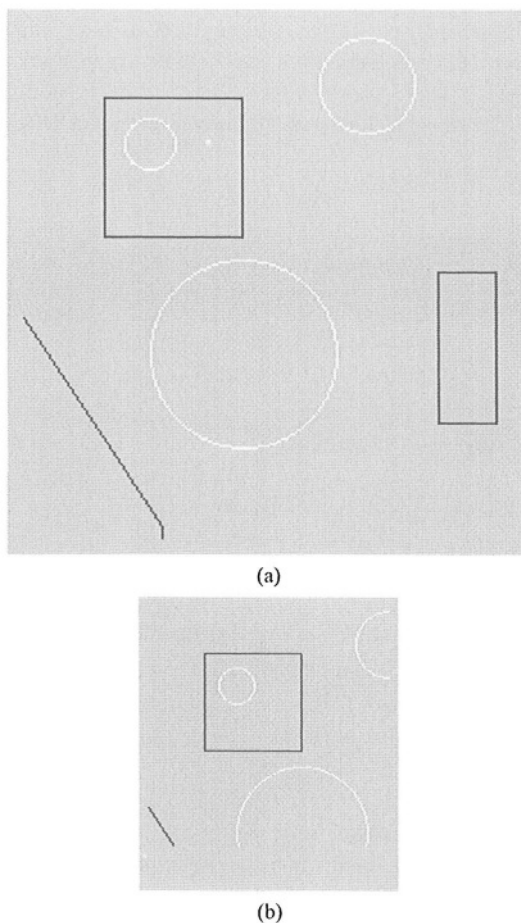


Figure 20.4. Synthetic example. (a) Edge map of the scene image. (b) Edge map of the model image.

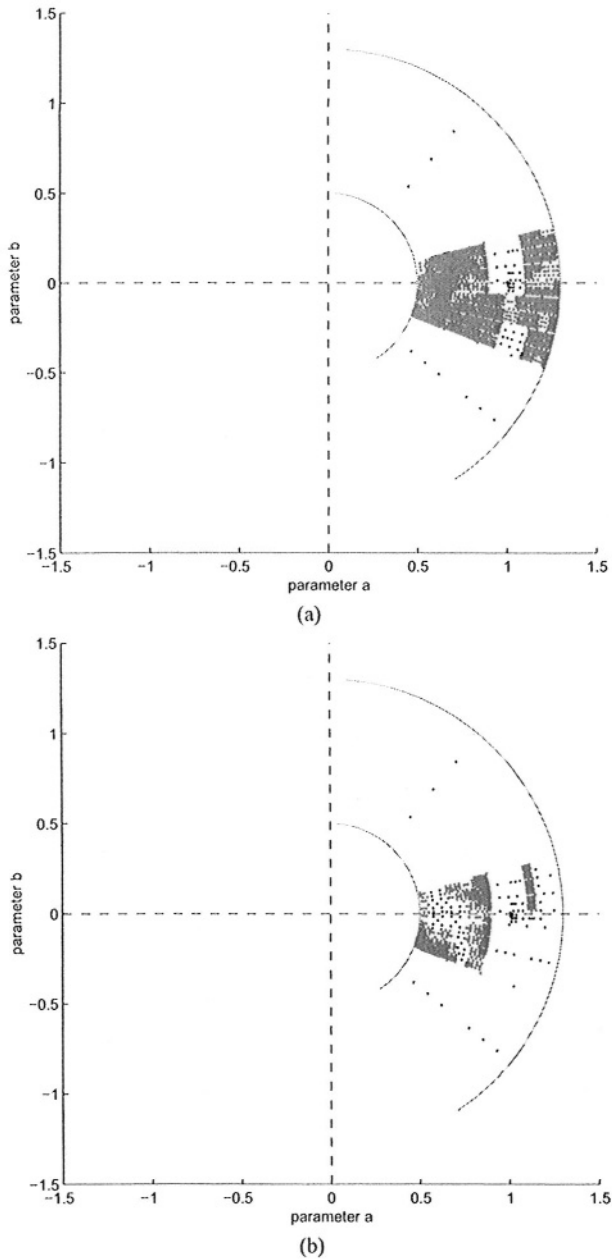


Figure 20.5. Synthetic example. (a) Distribution of transformations tested by the algorithm for the single class approach. (b) Multi-class approach.

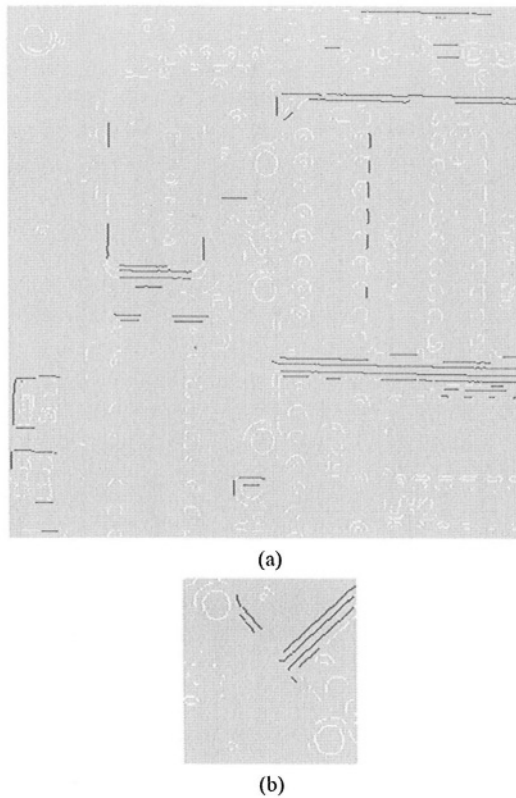


Figure 20.6. Circuit board example. (a) Edge map of the scene image. (b) Edge map of the model image.

distance transforms (one for each image/class combination), it provided savings in execution times.

As our focus was on illustrating the validity of the multi-class approach, we have not investigated how the segmentation process can be accomplished efficiently. Therefore, the problem of finding a suitable segmentation scheme is still open. It would also be interesting to investigate this method with other types of classifications.

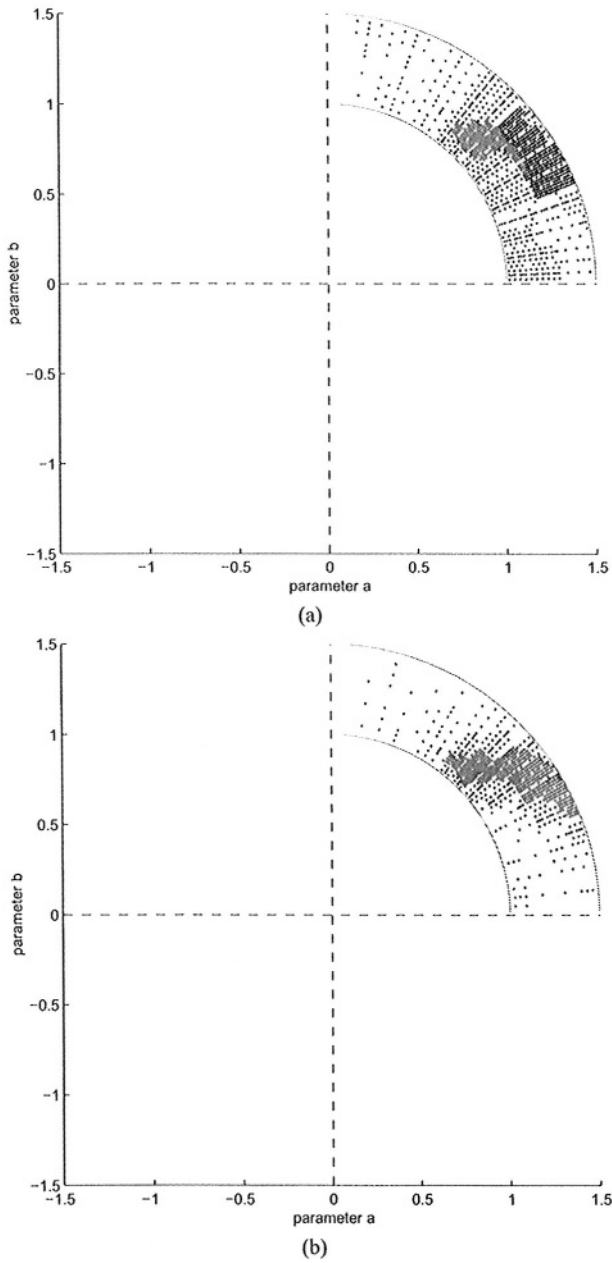


Figure 20.7. Circuit board example.(a) Distribution of transformations tested by the algorithm for the single class approach. (b) Multi-class approach.

References

- [1] W. J. Rucklidge. "Locating objects using the hausdorff distance", In *Proceedings of 5th International Conference on Computer Vision*, pp. 457-464, 1995.
- [2] D. Huttenlocher, G. Klanderman and W. Rucklidge. "Comparing images using the hausdorff distance". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850-863, 1993.
- [3] D. P. Huttenlocher and W. Rucklidge. "A multi-resolution technique for comparing images using the hausdorff distance". In *Proceedings of IEEE Conference on Computing, Vision and Pattern Recognition*, pp. 705-706, 1993.
- [4] W. Rucklidge. *Efficient Computation of the minimum hausdorff distance for visual recognition*. Ph.D. Thesis, Cornell University, January 1995.
- [5] C. F. Olson and D. P. Huttenlocher. "Automatic target recognition by matching oriented edge pixels". *IEEE Transactions on Image Processing*, vol. 6, no. 1, pp. 103-113, 1997.
- [6] D. M. Mount, N. S. Netanyahu and J. Le Moigne. "Efficient algorithms for robust feature matching". *Pattern Recognition*, vol. 32, pp. 17-38, 1999.
- [7] X. Yi and O. L. Camps. "Line feature-based recognition using hausdorff distance". In *Proceedings of the International Symposium on Computer Vision*, pp. 79-84, 1995.
- [8] X. Yi and O. L. Camps. "Line based recognition using a multidimensional hausdorff distance". *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 901-916, 1999.
- [9] W. Rucklidge. *Efficient Visual Recognition Using Hausdorff Distance*. Lecture Notes in Computer Science, Springer-Verlag, 1996.

This page intentionally left blank

INDEX

¥-shape, 343
0-shape, 343
3D, 1, 6, 11, 241, 303, 332

 computer games, 1
 scanners, 11
 seismic data-volumes, 241
 sensor calibration, 332
 virtual environments, 6
 wrinkling measurement, 303

A

Accumulation buffer, 98
Active, 333, 334, 336
 stereo vision, 333
 triangulation, 334
 triangulation sensors, 336
Adjacent connected texture
 regions, 317
Affine transformations, 8
Agent-aided Collaborative
 Environment (CVE), 313
AHR, 5, 31, 33, 34
Amplitude maxima, 252
Angle-preserving, 27, 32
 transformation 32
Anti-aliasing, 2, 3, 189
 filters, 3
Approximation, 268
Arbitrary parametric models, 201
A-shapes, 342
Attribute, 5, 31, 33, 41, 42, 49
 transition, 41, 42, 49
 graph, 5
 hypergraph, 31, 33
Augmented, 32, 47, 65
 reality, 47
 umbra, 65

 virtual reality, 32
Authalic mapping, 14
Automatic, 181, 215
 object modeling, 181
 texture generation, 215
AVR, 32
Axis Aligned Bounded Boxes 82, 83

B

Bandpass filter, 3
Barycentric mapping, 4, 11, 13, 23
Base texture, 319
Behavioural transformation, 46
Bidirectional-Reflectance 169, 178
Binary, 359, 379
 edge map, 359
 images, 379
Boundary smoothing, 382
Bounding, 80, 81
 sphere, 81
 volume, 80
BSP trees, 81
Bump mapping, 12
Bundle adjustment, 200

C

CAD, 8,76
Category theory, 33, 34, 36
CCD, 51, 115, 116, 121
Cg, 67, 107
 shader language, 67
 shading language, 106
Chord-length parameterization, 19
Coarse registration, 340
Coherent layers, 6, 131
Collision detection, 2, 6, 75
Combination model, 272

Compression, 2, 7, 12
 techniques, 7
 Computer, 2, 3, 11, 32, 36
 animation, 32
 vision, 2, 3, 11, 36
 Concentric mosaics, 136, 137, 154
 Conformal mapping, 11, 13
 Continuous wave modulation, 334
 Conversation agents, 220
 Convex combination, 13
 Convolution, 3, 96, 97, 288
 kernel, 97
 CIA-Tool, 312
 Coretraction, 34, 35
 preserving, 35
 Correction detection, 363
 Covariant functor, 35, 34, 35
 CSG models, 78
 Curvature, 12, 25
 Cut, 21
 nodes, 21
 path, 21
 Cylinders, 32

D

DAH, 38, 39
 Data, 37, 242, 332, 339
 acquisition, 332
 fusion, 332, 339
 volume, 242
 Deconvolution, 244
 Deformable, 32, 75
 objects, 75
 superquadrics, 32
 Density map, 278, 279, 281, 276
 generation, 276
 Depth, 2, 7, 182, 302
 conversion, 302
 image, 7, 182

 field, 2
 Dichotomy, 41, 42
 Digital, 2, 116
 image processing, 2
 micromirror device, 116
 Dilation operation, 293, 294
 Directional, 37, 61, 163-164, 167
 light source, 163-164, 167
 link, 37
 umbra, 61
 source formulation, 163
 DirectX9, 106
 Dirichlet energy, 14, 18
 Discrete, 14, 98, 171, 172, 173
 conformal mapping, 14
 Cosine Transform, 173
 Fourier Transform, 98
 surface, 14
 Wavelet Transform, 171, 172
 Displacement mapping, 24
 Dynamic, 2, 14, 37, 60, 67, 81
 hypergraph, 37
 illumination, 2
 parameterization, 14
 scene, 60, 67, 81
 structure, 37

E

EasyMall, 8
 EasyShow, 8
 Eccentricity, 376, 379, 384, 388
 E-commerce, 307
 Edge-based segmentation
 techniques, 351
 Edge-length-preserving, 27
 Elementary, 38, 46
 edges, 46
 graph, 38
 node, 38, 46

Entropy encoding, 26
 E-Partner, 7, 220
 system, 220
 Erosion operation, 293, 294
 Expression editing, 235
 mapping, 235
 Extended Morphological Fractal
 Analysis, 291, 293, 296, 305
 Extent, 378, 381, 386, 390
 Extremal vertex, 21

F

Fabric appearance, 7
 Face region subdivision, 232, 233, 234
 Facial, 7, 33, 220, 221, 224
 animation, 7, 224
 expression, 33
 expression synthesis, 220, 221
 Fast Fourier Transform, 6, 95, 96
 Feature, 42, 43, 289, 303, 374, 381, 396
 extraction, 289, 303, 374, 381
 level, 42, 43
 points, 396
 Fine registration, 341
 Fingerprint, 263, 264, 265, 266-268, 272, 280
 classification, 267
 identification, 263, 264, 265
 recognition, 263, 264, 266, 267, 268, 280
 Floater's, 13, 14
 method, 14
 Parameterization, 13
 Fold edges, 356
 Fourier, 95, 98, 345
 processing, 95
 techniques, 345

Transform, 98
 Fractal, 291, 292, 296, 297
 dimension, 291, 296
 model, 292
 vector, 297
 Fragment programming, 96, 103, 105
 Fragment shader, 67
 Frame buffer, 3
 Fringe, 115-116, 120-121
 pattern, 115-116, 120-121
 projection, 115
 From-point occlusion culling, 58
 From-region occlusion culling, 58
 Full-field optical methods, 115
 Functors, 36
 Fuzz, 284

G

Gaussian function, 285, 288, 289
 Geometric proxies, 146, 147, 148
 Geometric, 6, 12, 18
 processing, 12
 proxies, 6
 stretch metric, 18
 Geometrical, 11, 37
 spaces, 37
 stretch, 11
 Geometry, 230, 345
 Processing, 345
 driven, 230
 GPU 95, 106
 Grade rating, 291, 304
 Graphic, 1, 76, 95
 design, 1
 acceleration hardware, 76
 pipeline, 95
 Graphics-to-image, 4

H

Hardware-assisted rendering, 78
 Harmonic mapping, 11, 13, 17
 Hausdorf-Besicovich, 291
 Hausdorff, 393, 395, 397, 399,
 400
 distance, 393
 fraction, 395, 397, 399, 400
 High frequency, 19, 172, 173
 subband, 172, 173
 detail, 19
 Hole filling, 189
 Horizon, 242, 243, 251-256
 picking, 242, 251, 252, 253
 Hough transform, 401
 Human-computer interactive
 interface, 310
 HYCODE, 90
 Hydrid modeling, 183, 184
 Hyperedges, 37, 38, 40, 44, 46

I

IBFA, 7
 IBR, 7
 Image, 2, 5, 6, 26, 95, 166, 221
 alignment, 232
 capturing, 2
 compression, 26
 correlation, 221
 databases, 373
 level, 2
 processing 1, 95
 registration 6, 395
 resolution, 5
 segmentation, 288
 space, 6
 superposition, 166
 collision detection 75,78,82
 facial animation 220

 rendering, 2, 6, 333
 texture, 12
 Image-space interference test, 75
 Image-to-graphics, 4
 Impulse response, 97
 Integration, 118, 119, 123, 124
 Interactive, 1, 6, 57
 Interference, 6, 76
 detections, 76
 test, 6
 inter-pixel, 7, 171
 correlation 171
 Intra-pixel, 7, 169
 correlation, 169
 Isomorphic, 12, 22
 planar triangulation, 12
 Isomorphism, 34, 35
 preserving, 35
 Iso-surface, 248, 249, 250, 251
 Iterative closest point (ICP),
 116-117, 122-123, 131-132,
 145-148, 155, 341
 algorithm, 341

J

Join, 42
 JPEG, 161-162, 175-177
 Jump edges, 355

K

Kd-tree, 60, 67
 Kinematic, 32, 33, 37, 44
 features 33
 transformations 37, 44

L

Lagrangian, 7
 Language-independent, 229
 Large-scale Digital Michelangelo

Project, 346
 Laser, 284, 298
 probe, 298
 scanner, 284
 Layered Depth Image, 132,
 141-143, 153
 Light, 132, 135, 144-145, 148
 -152, 164, 335
 beam, 335
 field, 132, 135, 144-145, 148-152
 plane, 335
 Lighting system, 299
 Linear least squares, 18
 Lipsynching, 7
 Local approximation, 8, 354
 technique, 354
 Low-cost range sensors, 337
 Lowest Frequency Subband, 172
 Lumigraph, 132, 135, 146, 153

M

Manifold, 20
 Map-based, 247, 248
 techniques, 248
 volume visualization, 247
 Marching cube algorithm,
 250-252
 Matting information, 6
 Mesh, 5, 11, 26, 345
 compression, 5, 26
 optimization, 345
 parameterization, 11
 transform, 23
 Minutiae, 266, 267
 extraction, 267
 matching, 266
 Missed classification, 364
 Model generation, 332, 344
 Morphological, 8, 22, 284, 292

fractal, 8, 284, 292
 watersheds, 22
 Mosaics, 137-138
 Motion, 6, 60
 response, 6
 volumes, 60
 MPEG, 161-162, 175-177
 Multi-body, 32
 Multi-chart transform, 20, 22
 Multi-resolution, 12, 395, 398
 analysis, 12
 Multi-scale Fractal Vector, 291, 296
 Multi-target picking theory, 254

N

Noise, 355, 364
 classification 364
 pixels 355
 Non-directional light source,
 167-168, 168
 Non-linear minimization, 18
 Normal mapping 12
 NURBS (Non-Uniform Rational
 B-Splines), 359
 NVIDIA, 4

O

OBB-tree, 184
 Object, 76, 87, 92
 Interactions, 76
 space coherence, 92
 space culling, 76, 87
 Object-based collision detection,
 78, 81
 Objective, 283, 305
 Assessment, 305
 Evaluation, 283
 Object-space interference test, 75
 Occluder fusion, 59

Occlusion, 5, 58, 59, 60
 culling 5, 58
 map, 59
 queries, 60

Online synthesis phase, 226

Opacity, 256

OpenGL, 24, 106

 Functions, 24

 Shading Language, 106

Operators, 46

 orientation field, 264-269,
 278-281

 generation, 276

Oriented Bounded Box, 82

P

Panorama, 134, 136-138, 154

Parameterized coordinates, 12

 Partial, 8, 45, 374, 380, 384

 match, 8, 374, 380, 384, 386

Photometric stereo, 8, 284, 298

Photo-realistic, 220

Pilling 7, 8, 283, 284, 290, 305

 modeling, 8

Plenoptic, 6, 133-135, 138,

 143-145, 153, 154, 161-167

 function, 133-135, 138, 154, 161-167

 sampling 6, 143-145, 153

Point, 7, 13

 clouds, 7

 sequences, 13

Point-based representation, 189,

 190, 191

Point-Charge, 272-275

Polar fleece, 7, 283, 284, 291, 294

Principal component analysis, 165

Q

Quadratic surface, 7, 199, 205

 reconstruction, 199

 recovery, 205

Quantization, 26

R

Radial Basis Functions (RBFs), 344

RAG, 39

RAMDAC, 4

Random graphs, 33

Range image, 111, 114, 117-118,

 122-126, 332, 353

 segmentation, 353

Ray, 69, 83, 256, 257

 casting, 83, 256, 257

 space, 69

Region-based segmentation

 techniques, 353

Relighting, 165-168

Rendering, 78, 192

S

Sample buffer, 3

Segmentation, 2, 358

Seismic, 7, 241-245, 247, 248,

 250, 252, 254, 257

 data 242, 244, 245-247,

 252, 254, 258

 exploration, 7, 241-243, 247

 survey, 242, 243

 visualization, 248, 250, 257

 wavelet, 252

Separating, 82, 90

 axis theorem, 82

 vector algorithm, 90

 vector technique, 82

Shape, 2, 373, 376, 377, 380

 analysis, 373, 377

 blending, 2

 representation, 373, 376

Shape-based image retrieval, 373,
376, 383, 387, 390
Speech driven, 221
Spherical harmonic 165-166,
169-174, 178
Stencil buffer, 60, 78, 82, 85
Stereo vision, 50
Structured, 284, 342, 335
light, 284, 335
range images, 342
Subjective evaluation, 289
Subsurface, 7, 241
phenomena, 241
resource extraction, 241
structures, 7
Surface, 13, 47
discontinuity, 47
flattening, 13
Sweeping volume, 88

T

Template, 8, 284, 285, 287, 288,
305
matching, 8, 284, 285, 287,
288, 305
training, 287
temporal coherence, 92
Tetrahedrons, 33
Texture, 5, 12, 105, 108, 248, 315,
317, 318
block, 315
buffers, 105
generation, 108
mapping, 5, 12, 248, 317
morphing, 317
smooth, 318
vector, 317
multi-sliced volume, 248
Time-of-flight (TOF), 334, 336

sensors, 336
Topological disk, 11, 12
Triangular, 7, 31, 46
meshes, 31
patches, 7

U

Uncertainty region, 399, 400
Under-segmentation, 364
Uniform parameterization, 13
Unstructured range data, 342

V

Vector quantization, 135, 171
Vertex programming, 105
View, 6, 7, 58, 139, 140, 161-162
cell, 6
dependency, 6
interpolation, 7, 139, 140,
161-162
morphing, 140
cell, 58
View-dependent, 141, 143,
145-146, 153
geometry, 145-146
texture mapping 141, 143, 153
Virtual, 32, 57, 65, 75
environment, 57
navigation, 75
occluder, 65
reality, 32
Virtual Reality (VR), 32, 310
Visibility culling, 58
Visualized registration, 122-124
Volume, 244, 246, 247, 256, 258
data, 246
illustration, 258
rendering, 247, 256
visualization, 7, 244

Voronoi diagram, 22, 78

Voxel, 246, 256

Voxel grid, 81

W

Walkthrough, 7

Wavelet coder, 26

Weighted Least Square algorithm,
274

Whole match, 374, 381, 384

wire-frame, 47

wrinkling, 284, 304, 305

wrinkle, 297

assessment, 297

wrinkling, 7

Y

YUV, 175

Z

Z-buffer, 78, 82

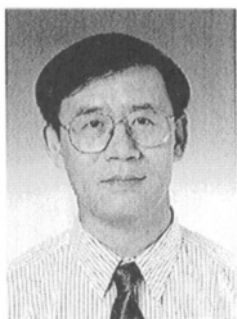
hardware, 78

Zero-pole model, 268, 269, 271,
272, 274, 276

Zone system, 256

Integrated Image and Graphics Technologies

This book provides a collection of twenty chapters containing tutorial article and new material describing, in a unified way, the basic concepts, theories and characteristic features of integrating/formulating different facets of integrated image and graphics technologies, with recent developments and significant applications in various areas. The book, which is unique in its characters, will be useful to graduate students and researchers in computer science, electrical engineering, systems science, and information technology as a reference book and a text book for some parts of the curriculum. The researchers and practitioners in industry and R&D laboratories working in the fields of image processing, computer vision, graphics, CAD, intelligent system, pattern recognition, and Internet will also be benefited.



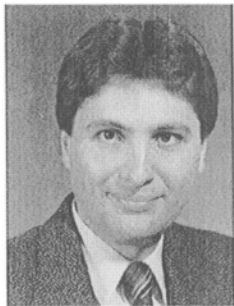
David Zhang graduated in Computer Science from Peking University and received his M.Sc and Ph.D degree in Computer Science and Engineering from Harbin Institute of Technology (HIT) in 1983 and 1985, respectively. From 1986 to 1988 he was a postdoctoral fellow at Tsinghua University and became an associate professor at Academia Sinica, Beijing, China. In 1988, he joined University of Windsor, Ontario, Canada, as a Visiting Research Fellow in Electrical Engineering. He received his second Ph.D degree in Electrical and Computer Engineering at University of Waterloo, Ontario, Canada, in 1994. Currently, he is a professor in

Hong Kong Polytechnic University. He is Founder and Director of Biometrics Research Center (UGC/CRC) supported by Hong Kong Government. He is also an Adjunct Professor in University of Waterloo, Tsinghua University, Shanghai Jiao Tong University and HIT. In addition, he is Founder and Editor-in-Chief, *International Journal of Image and Graphics*, and Associate Editor of many international journals like *IEEE Trans. on Systems, Man and Cybernetics*, *Pattern Recognition*. His research areas include biometrics, pattern recognition/image processing and integrated image and graphics technologies. So far, he has published over 200 papers including seven books around his research areas.



Mohamed Kamel received the B.Sc. (Hons) degree in Electrical Engineering from the University of Alexandria, Egypt, in 1970, his M.Sc. degree in Computation from McMaster University, Hamilton, Canada, in 1974 and his Ph.D. degree in Computer Science from the University of Toronto, Canada, in 1981. Since 1985, he has been with the Department of Systems Design Engineering, University of Waterloo, Canada. He is currently a Canada Research Chair Professor in Cooperative Intelligent Systems and Director of

the Pattern Analysis and Machine Intelligence Laboratory. In 1989/1990 he was a visiting faculty at the University of Toronto and Purdue University. In 1996/1997 he was a visiting professor at ACE NASA center, University of New Mexico and the University of Porto in Portugal. His research interests are in computational intelligence, pattern recognition and distributed and multiagent systems. He has authored and co-authored over 200 papers in journals and conference proceedings, 2 patents and numerous technical and industrial project reports. He is the Editor-in-Chief of the *International Journal of Robotics and Automation*, Associate Editor of the *Intelligent Automation and Soft Computing*, *Simulation*, *Pattern Recognition Letters*, the *International Journal of Image and Graphics*, and *IEEE trans on SMC*. Dr. Kamel is a member of ACM, AAI, CIPS and a senior member of IEEE. He served as a consultant for General Motors, NCR, Diffracto, Bell & Howell, IBM, Northern Telecom and Spar Aerospace. He is a member of the board of directors and co-founder of Virtek Vision Inc. of Waterloo.



George Baci is currently an Associate Professor in the Department of Computing at the Hong Kong Polytechnic University. He holds degrees in Computer Science and Applied Mathematics and a PhD in Systems Design Engineering from University of Waterloo, where he has been a member of the Computer Graphics Lab and the Pattern Analysis and Machine Intelligence group. He joined the Computer Science Department at the Hong Kong University of Science and Technology (HKUST) in 1993 and initiated one of the first graduate and undergraduate Computer Graphics curricula in Hong Kong. He was the founding

director of the Graphics and Music Experimentation (GAME) Laboratory at HKUST where he conducted research in motion synthesis, 3D user interfaces, rendering, geometric modeling and animation. Currently, Dr. Baci has been serving as the founding director of the Graphics and Multimedia Applications (GAMA) Laboratory at the Hong Kong Polytechnic University where he performs research in 3D motion capture, motion analysis and synthesis, 3D geometric modeling and network graphics. He is a member of IEEE and ACM.