Abstract of "Statistical Inverse Ray Tracing for Image-Based 3-d Modeling" by Shubao Liu, Ph.D., Brown University, May 2011.

This thesis addresses the problem of capturing 3-d geometry and appearance from multiple 2-d images, which has wide applications in virtual reality, entertainment, human-computer interface, surveillance, navigation, and high-level vision tasks (e.g., visual tracking, classification, recognition, scene analysis for robotic task accomplishment), etc. This thesis presents an inverse ray tracing approach based on statistical inference. Instead of matching image features/pixels across images as most of the traditional approaches do, the inverse ray tracing approach models the image generation process directly and searches for the best 3-d geometry and surface reflectance model to explain all the observations. This approach can better handle difficult problems in multi-view stereo, including large camera baseline, occlusion, matching ambiguities, irregular concavities and convexities, etc., than traditional methods, without additional information and assumptions, such as initial surface estimates or simple background assumptions. Here the image generation process is modeled through volumetric ray tracing, where the occlusion/visibility is accurately modeled. All the constraints (including ray constraints and prior knowledge about the geometry) are put into the Ray Markov Random Field (Ray MRF) formulation. This MRF model is unusual in the sense that the ray clique, which models the ray-tracing process, consists of thousands of random variables instead of two to dozens as in typical MRFs. This large ray clique presents a major challenge to the inference algorithm, because of the combinatorial explosion of possible configurations. In this work an algorithm with linear computational complexity is developed to solve the estimation problem. More specifically, it is shown that a highly optimized belief propagation algorithm, deep belief propagation (DBP), can tackle the challenging problem effectively by exploring the recursive chain structure of the ray clique energy. Then the DBP algorithm is also extended to solve the inference problem for a broader class of higher-order MRFs. The algorithm developed is capable of handling general and complex scenes of large varieties, general camera configurations (both small and large baselines), and can generate accurate and photo-realistic 3-d models. The advantages of the proposed approach have been verified by extensive experiments on standard and locally collected challenging datasets.

Statistical Inverse Ray Tracing for Image-Based 3-d Modeling

by

Shubao Liu

B. E., The University of Science and Technology of China, 2003

M. Phil., The Chinese University of Hong Kong, 2005

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the School of Engineering at Brown University

Providence, Rhode Island

May 2011

This dissertation by Shubao Liu is accepted in its present form by
the School of Engineering as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

David B. Cooper, Advisor
School of Engineering

Recommended to the Graduate Council

Date _____ _____

Joseph Mundy, Reader
School of Engineering

Date _____ _____

Gabriel Taubin, Reader
School of Engineering

Date _____ _____

Stuart Geman, Reader
Division of Applied Mathematics

Approved by the Graduate Council

Date _____ _____

Peter M. Weber
Dean of the Graduate School

Vita

Shubao Liu was born in Changle, Shandong, China on May 14, 1981. He obtained the B.E. degree from the University of Science and Technology of China (USTC), Hefei, China in 2003, and the M.Phil. degree from The Chinese University of Hong Kong, Hong Kong in 2005. Since August 2005, he has been a graduate student at Brown University, studying computer vision, supervised by Professor David B. Cooper at the Electrical and Computer Engineering Program, School of Engineering. In summer 2008, he was an intern at Google Inc, Mountain View, CA. His main research interest is on developing scalable statistical methods for various computer vision problems, in particular, MRF-based generative modeling and efficient inference, and hierarchical graphical models for visual recognition and scene understanding. During his PhD period of study, he has published several papers in selective peer-reviewed journals and conferences on computer vision.

Acknowledgements

This dissertation would not have been possible without the help and support of many people. In particular, I would like to thank my advisor, Professor David B. Cooper, for introducing me to this exciting research field, and providing me with the guidance and support essential for the achievements in this work. Professor Cooper has always been supportive of my academic exploration and research work, encouraging me to explore alternative directions, and kept my interests in top priority. I would like to thank my thesis committee member Professor Stuart Geman. I have benefited a lot from taking his three mathematical courses. I am deeply impressed by his enthusiasm for research and education. With his profound knowledge in mathematics, Professor Geman gave me many insightful comments on the thesis. I would like to thank Professor Joseph Mundy and Professor Gabriel Taubin for serving on my thesis committee and providing me with very helpful suggestions on both my thesis work and career planning. I have learned a lot from them about video processing and geometry processing. I am grateful to Dr. Kongbin Kang and Dr. Jean-Philippe Tarel for providing me essential guidance during my PhD study, especially the first two years. I would also like to thank my mentors at Google during my internship there: Dr. Raquel Romano, Dr. Dar-Shyang Lee and Dr. Ranjith Unnikrishnan for providing a pleasant and fruitful experience, from which I have learned a lot.

Brown University and LEMS lab, in particular, have provided a relaxed and pleasant environment for my study and research. For this, I would like to thank my labmates at Brown LEMS, who have made the lab an enjoyable place to stay, and provided all kinds of inspirations and information during discussion, especially Yong, Ibrahim, Eduardo, Osman, Nhon, Ming-Ching, Ozge, Kilho, etc. I would also like to thank all my friends at Brown. They made my life here colorful and memorable.

I would not have been able to receive the high-quality education and to freely pursue my interests in scientific research without the continuous support and encouragement from my family. I cannot thank my parents and my sister enough for all these. Finally, the

thesis is dedicated to my wife, Dandan, who has contributed a lot through encouragement, discussion, editorial assistance, and most importantly LOVE.

Contents

List of Tables **x**

List of Figures **xi**

1 Introduction **1**

1.1 Approaches to 3-d Modeling 2

1.2 Image-Based 3-d Modeling 3

1.2.1 Goal of Image-Based 3-d Modeling 4

1.2.2 Basic Concepts of Image-Based 3-d Modeling 5

1.2.3 Difficulties in Image-Based 3-d Modeling 8

1.3 Statistical Inverse Ray Tracing Approach 10

1.3.1 Image-based 3-d Modeling as an Inverse Problem 10

1.3.2 Important Role of Probability and Prior Knowledge 10

1.4 Main Contributions 11

1.5 Thesis Organization 12

2 Literature Review **14**

2.1 The State-of-the-Art Approaches to Image-Based 3-d Modeling 14

2.1.1 Feature-Based Point Cloud Reconstruction 14

2.1.2 Iterative Surface Evolution 17

2.1.3 Volumetric Reconstruction 21

2.2 Camera Calibration Development 25

3 A Statistical Inverse Ray Tracing Approach to Multi-View Stereo **27**

3.1 Review of Ray Tracing 27

3.1.1 The Rendering Equation and the Shading Model 28

3.1.2 Ray Tracing Algorithm 31

3.2 Statistical Inverse Ray Tracing . . . 34
3.2.1 Analysis by Synthesis Framework . . . 34
3.2.2 Photo-Consistency Optimization . . . 35
3.2.3 Related Work . . . 36

4 Ray Markov Random Field **40**
4.1 Review of Markov Random Fields . . . 40
4.2 Ray Markov Random Field . . . 45
4.2.1 Ray Clique . . . 45
4.2.2 Ray Markov Random Field Model . . . 45
4.2.3 Comparison between Ray MRF and Standard MRF . . . 46
4.2.4 Factor Graph Representation of Ray MRF . . . 47
4.2.5 Ray-Voxel Intersection Relationship . . . 49
4.3 Related Work . . . 51

5 Voxel Occupancy and Color Estimation **52**
5.1 Review of Loopy Belief Propagation . . . 53
5.1.1 Dynamic Programming and Message Passing . . . 55
5.1.2 Dynamic Programming and Belief Propagation . . . 58
5.1.3 Canonical Form of Belief Propagation . . . 60
5.2 Voxel Occupancy Estimation, Given Voxel Colors . . . 61
5.2.1 Belief Propagation for Voxel Occupancy Estimation . . . 61
5.2.2 Deep Belief Propagation for Voxel Occupancy Estimation . . . 65
5.3 Voxel Color Estimation, Given Voxel Occupancies . . . 74
5.4 Algorithm Summary . . . 78
5.5 Algorithm Convergence . . . 80
5.6 Related Work . . . 80

6 Modeling Background and Transient Clutter **83**
6.1 "Volume + Bubble" Model . . . 83
6.1.1 Domain of Localization (DOL) . . . 83
6.1.2 "Bubble" Background Model . . . 85
6.1.3 Spherical Background Estimation . . . 86
6.1.4 Related Work . . . 88
6.2 Handling Transient Clutter . . . 89
6.2.1 Transient Clutter Modeling and Estimation . . . 89

6.2.2 Related Work 90

7 Experiments and Evaluation **91**

7.1 Dino Sparse Ring Dataset (dinoSR16) 91

7.2 Temple Sparse Ring Dataset (templeSR16) 94

7.3 Elephants Dataset (elephants40) 94

7.4 Building Dataset (building14) 101

7.5 Horse Dataset (horse29) 106

7.6 Flower Dataset (flower31) 111

7.7 Capitol Dataset (capitol40) 112

8 Deep Belief Propagation For Efficient Higher-Order MRF Inference **118**

8.1 Deep Belief Propagation for Higher-Order Cliques 119

8.1.1 Dynamic Programming for Higher-Order Cliques 121

8.1.2 Deep Factor Graph 122

8.1.3 Deep Belief Propagation 123

8.2 Deep Factorization of the Ray Clique 126

9 Conclusion and Discussion **129**

Bibliography **132**

List of Tables

1.1 Comparison between laser scanning, structured light scanning and image-based modeling 13

2.1 Characteristics of different approaches to multi-view stereo 15

3.1 Notations in inverse ray tracing 29

4.1 MRF notations 40

5.1 Notations in belief propagation 53
5.2 Notations in message passing for Ray MRF 61

7.1 Accuracy and completeness comparison between PMVS2, IRAY and voxel world model for the dinoSR16 dataset. Notice that the voxel world algorithm has only been evaluated on the dinoR (40 images) datasets. Since the dinoSR dataset is a subset of dinoR dataset, it is reasonable to assume that the performance of voxel world model will not be better than the performance reported here. 94
7.2 Accuracy and completeness comparison between PMVS2, IRAY and voxel world model for the templeSR16 dataset. Notice that the voxel world model algorithm has only been evaluated on the templeR (40 images) datasets. Since the templeSR dataset is a subset of templeR dataset, it is reasonable to assume that the performance of voxel world model will not be better than the performance reported here. 95

List of Figures

1.1 Zakhor group's work on city scan with a 2-d laser scanner and a digital camera [42]. From left to right: Equipments mounted on a moving vehicle, scanned city blocks, close look at the reconstructed facade. 2

1.2 Capturing 3-d object with structured light [77]. From left to right: equipments (1 projector, 2 cameras), sparse structured light cast on the David bust, dense structured light, and reconstructed textured model 3

1.3 Illustration of the image based 3-d modeling problem's input and output . . 4

1.4 BRDF diagram 6

1.5 Illustration of triangulation 7

1.6 The variety of scenes that image-based 3-d modeling needs to deal with: indoor (a–d) vs. outdoor (e,f), small (a–e) vs big (e–g), object of simple topology (a,b) vs. object of complex topology (d, e, g), Lambertian objects (a,b,c) vs. object with reflective surface (f,g), images captured with cameras mounted on robot arm (a,b) vs. images captured with a hand-held camera (c–g). 9

2.1 The pipeline of the depth fusion approach 16

2.2 Furukawa's multi-view stereo pipeline 17

2.3 Working example of Furukawa's multi-view stereo pipeline: (a) example input image; (b) detected sparse features; (c) reconstructed sparse points; (d) filtered dense points; (d) reconstructed mesh model 17

2.4 Representative reconstruction result from the state-of-the-art feature expansion approach [44]. Reconstruction of Rome Colosseum from 1167 images collected from Internet (after image selection, 490 of them are used). The model has 5,747,083 points. 18

2.5 Representative results from one of the state-of-the-art mesh-based surface evolution approaches [34] 19

2.6 Representative reconstruction result from one of the state-of-the-art mesh-based surface evolution approaches [58]. Reconstruction of Aiguille du Midi (France) from 53 images with 5M pixels taken from a helicopter. The model has 4M triangles. 20
2.7 Representative reconstruction result from one of the level-set surface evolution approaches [60]. Reconstruction of Van Gogh statue, made of polished metal with strong specular highlights, from 281 images (shown on the left are 4 typical images). Silhouette information is used. 20
2.8 Representative result from one of the state-of-the-art graph-cuts-based volumetric reconstruction methods [136]. On the top is 4 out of 140 input images of a fully textured toy house; reconstruction of the model rendered with the same view angle. In the reconstruction, manual segmented silhouettes are used. 21
2.9 Representative result from the probabilistic space carving method by Broadhurst [18]. (a) Four out of 11 input images. These clean images are manually segmented from a cluttered image with trees and other building in the background. (b, c) Rendering of the reconstruction. 22
2.10 Representative result from the voxel-world-model method by Pollard, Crispell and Mundy [106, 105, 30]. On the left is one out of 255 input aerial images extracted from a 720p video; in the middle is a volumetric rendering of the reconstruction of the Capitol building; on the right is a novel view synthesized from the reconstructed volume. 22
2.11 Representative result of the state-of-the-art self-calibration algorithm [124]. Visualization of the reconstructed point cloud and cameras produced by Bundler (Trevi Fountain dataset). 26
2.12 camera pose visualization and epipolar-line checking between two images in the horse dataset. The images shown are cropped to highlight the picked points. (best viewed in color) 26

3.1 Illustration of ray tracing 28
3.2 Illustration of inverse ray tracing 28
3.3 The interaction amongst the light source, surface geometry and surface material in ray tracing 28
3.4 Illustration of ray traversal path 31

3.5 Illustration of the relationship between pixel, ray, and voxels: (a) a pixel ray passes through 4 voxels of the 2-d volume; (b) the voxels are ordered with the ray traversal order. 33
3.6 Connections amongst rendering, multi-view stereo, ray tracing and inverse ray tracing 35
3.7 Illustration of ray constraint in inverse ray tracing 35
3.8 Relighting objects from image collections [56]. From left to right: samples of the input images downloaded from flickr [1], sparse point cloud reconstruction and camera visualization, reconstructed surface model, re-lighted object. . . 38
3.9 Diagram of inverse rendering [56] 39
3.10 Diagram of full inverse ray tracing 39

4.1 Factor graph representation of probability distribution $P(x_1, x_2, x_3, x_4, x_5) \propto f_1(x_1, x_3) f_2(x_2, x_3, x_5) f_3(x_3) f_4(x_3, x_4) f_5(x_5)$, or equivalently energy $E(x_1, x_2, x_3, x_4, x_5) = E_1(x_1, x_3) + E_2(x_2, x_3, x_5) + E_3(x_3) + E_4(x_3, x_4) + E_5(x_5)$. 44
4.2 Conventional representation and factor graph representation of the pair-wise MRF and the 4th-order MRF 44
4.3 Visualization of the unary and pair-wise cliques in Ray MRF 47
4.4 Ray clique visualization 48
4.5 Ray MRF visualization (with the factor graph representation [71]): black balls represent voxels, cubes represent clique factors and tubes represent clique connections: green for unary clique, blue for pair-wise clique, and red for ray clique. (best viewed in color) 48
4.6 Ray line model 49
4.7 Ray tube model 49
4.8 Ray cone model 49
4.9 Even sampling in ray-cone model 50
4.10 Proportional sampling in ray-cone model 50

5.1 Alternating estimation of voxel occupancies and colors 53
5.2 Message-passing rule for soldier counting in a queue 57
5.3 Counting soldiers in a queue with a message passing rule 57
5.4 Message-passing rule for soldier counting in a tree configuration 57
5.5 Counting soldiers in a tree configuration with a message passing rule 57
5.6 Tree-structured joint probability 60

5.7 Messages for computing $P(x_3)$ on the tree-structured probabilistic graphical model . . . 60
5.8 Messages for computing all the marginal distributions on the tree-structured probabilistic graphical model . . . 60
5.9 Toy ray clique example . . . 64
5.10 Voxel's visibilities in different views, given binary voxel occupancies. White squares denote empty voxels; dark squares denote solid voxels. . . . 76
5.11 Voxel's visibilities in different views, given probabilistic voxel occupancies. Darker squares denote higher probability of the voxel being solid. . . . 76
5.12 Information flow in the inverse ray tracing algorithm . . . 79
5.13 Convergence behavior of the proposed algorithm . . . 81
5.14 Convergence comparison of two visibility estimation methods: product of marginal and joint distribution . . . 82

6.1 Camera frustum . . . 84
6.2 Intersection of camera frustums . . . 84
6.3 Domain of Localization (DOL) . . . 84
6.4 Typical camera configurations. (a) circular camera motion around an object; (b) camera rotating around a center to create panoramic view of the surrounding scene; (c) irregular camera motion. . . . 85
6.5 3-d representation of scene: volume + background panorama . . . 86
6.6 Spherical panorama [93] . . . 86
6.7 Cube representation of the 360° panorama . . . 87
6.8 Ray-volume-background intersection relationship . . . 88
6.9 Ray-background intersection model . . . 88
6.10 A ray passes through a line of voxels, and finally hits the background pixel (x_{rg}). . . . 88
6.11 Ray-mask-volume-background intersection relationship . . . 89
6.12 A ray passes through the transient mask, a line of voxels and finally hits the spherical background . . . 90

7.1 Stanford Spherical Gantry image capture and camera calibration system. `http://graphics.stanford.edu/projects/gantry/` . . . 92
7.2 dinoSR16: thumbnails of all 16 images in the dataset . . . 92

7.3 dinoSR16: 1/4 cut of the occupancy-color volume: colored and non-colored model . . . 93
7.4 dinoSR16: two different views of the textured and non-textured reconstruction models generated by IRAY (the proposed approach) . . . 93
7.5 templeSR16: thumbnails of all 16 images in the dataset . . . 94
7.6 templeSR16: 1/4 cut of the occupancy-color volume: colored and non-colored model . . . 95
7.7 templeSR16: two different views of the textured and non-textured reconstruction models of IRAY (the proposed approach) . . . 95
7.8 elephant40: thumbnails of all 40 images in the dataset . . . 97
7.9 elephant40: two sample images . . . 97
7.10 elephants40: visualization of camera poses and sparse point cloud of elephants40 dataset generated by Bundler software (each camera is represented by two color dots) . . . 98
7.11 elephant40: PMVS2's oriented point cloud outputs . . . 99
7.12 elephants40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon . . . 99
7.13 elephant40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + BallPivoting . . . 100
7.14 elephant40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm) 100
7.15 elephants40: 1/2 cut of the reconstructed occupancy-color volume . . . 100
7.16 building14: thumbnails of all 14 images in the dataset . . . 101
7.17 building14: two sample images . . . 102
7.18 building14: visualization of camera poses and sparse point cloud generated by the Bundler software (each camera is represented by two close color dots) 102
7.19 building14: PMVS2's oriented point cloud outputs . . . 102
7.20 building14: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon . . . 103
7.21 building14: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by from IRAY (the proposed algorithm) . . . 104
7.22 building14: 1/2 cut of the reconstructed occupancy-color volume . . . 104

7.23 building14: two example transient masks generated automatically with IRAY. Top: two sample input images (the same two views as shown in Fig. 7.17); bottom: the corresponding transient masks generated by the proposed algorithm, where whiter pixels mean transient observations. 105
7.24 horse29: thumbnails of all 29 images in the dataset 107
7.25 horse29: two example images 107
7.26 horse29: visualization of the camera poses and sparse point cloud generated by Bundler software (each camera is represented by two close color dots) . . 107
7.27 horse29: PMVS2's oriented point cloud outputs 108
7.28 horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon 108
7.29 horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + BallPivoting 109
7.30 horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm) . 110
7.31 horse29: different 1/2 cuts of the reconstructed occupancy-color volume . . 111
7.32 flower31: thumbnail views of all 31 images in the dataset 112
7.33 flower31: two example images 112
7.34 flower31: visualization of the camera positions and sparse point cloud generated by the Bundler software (each camera is represented by two color dots) 113
7.35 flower31: two different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm) . 113
7.36 captiol40: thumbnail views of the 40 training images in the dataset 113
7.37 capitol40: visualization of the camera locations of the 40 training images (in blue cross) and 9 test images (in red circle). The ground is approximately on the xy plane with z = 0. 114
7.38 capitol40: visual comparison between (a) test image (ground truth), (b) rendered image from the voxel world model, (c) rendered image from IRAY, and (d) rendered image from IRAY with prior terms (pairwise smoothing terms and baloon terms) disabled. 115
7.39 captiol40: mean absolute error of the synthetic images for both VWM, IRAY and IRAY without prior 116

7.40 capitol40: sub-region visual comparison between (a) test image (ground truth), (b) rendered image from the voxel world model, (c) rendered image from IRAY, and (d) rendered image from IRAY with prior terms disabled. . 116
7.41 captiol40: mean absolute error of the synthetic images over the cropped center region for both VWM, IRAY and IRAY without prior 117

8.1 Factor graph representation of the clique $f(x_1, x_2, x_3, x_4)$ 120
8.2 Deep factor graph of a clique node 123
8.3 A branch of the deep factor graph centered around an intermediate variable node z. Here the parent node f_z of z could be a factor/clique node or an intermediate variable node, so it is represented by an overlapped shape of square and triangle; each child node could be an original variable node or an intermediate variable node, so it is represented by an overlapped shape of circle and triangle. 124
8.4 Message flow from an intermediate variable node z to its parent node f_z . . 124
8.5 Message flow from an intermediate variable node z to its child node x_{zi} . . 124
8.6 Message passing on deep factor graph 125
8.7 A ray passes through n voxels 126
8.8 Deep factorization of ray clique, i.e., the deep factor graph of ray clique . . 126
8.9 Message passing on ray clique deep factor graph 127

Chapter 1

Introduction

With the prevalence of digital cameras, a new culture and life style is slowly emerging. Unlike analog cameras (aka, film cameras), taking a picture essentially has no cost with digital cameras. So nowadays people take many tour photos on a vacation to share their experience with family and friends. According to facebook [1], its service averages about 100 million photo uploads per day as of 2010. What to do with this huge and increasing number of images? Browsing photos is a pleasant experience, but quite often it is not enough. For people who have never been to that place, it is often difficult for them to build connections between photos that are taken around the same place but from different angles and zooms, due to the missing 3-d context. What if they could join a virtual trip by navigating through a full 3-d presentation of the scene created from the captured photos? Furthermore, in the future there will likely be virtual touring services providing photo-realistic virtual reality experiences that could take people to many places in the world. These applications are exciting and will likely dramatically change people's travel and living styles. Imagine other application scenarios: users can capture and share photo-realistic 3-d models of their gadgets with their friends; for sellers, they could post their goods or the whole physical stores online in 3-d, instead of 2-d photos; for archaeologists, they could share archaeological findings with their colleagues in 3-d; in industry, better surveillance camera system can be built with 3-d tracking and recognition. In summary, 3-d representation can provide a natural user experience, and is a powerful intrinsic scene representation for high-level vision tasks; and the technique that can turn 2-d images into full 3-d representation will open the door to many potential applications, such as virtual reality, entertainment, human-computer interface, surveillance, navigation, and high-level vision tasks, etc.

[1] `http://blog.facebook.com/blog.php?post=403838582130`

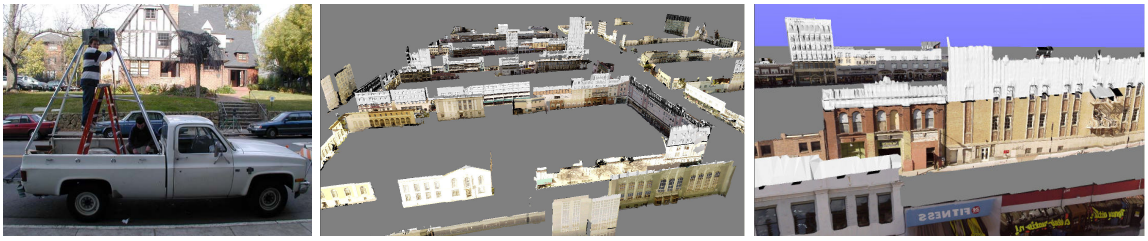


Figure 1.1: Zakhor group's work on city scan with a 2-d laser scanner and a digital camera [42]. From left to right: Equipments mounted on a moving vehicle, scanned city blocks, close look at the reconstructed facade.

Making these hypothetical applications into reality requires high-quality photo-realistic 3-d capture of physical objects and scenes. There are many competing technologies for that purpose, but the most general, affordable and accessible approach is with consumer cameras. Before going to the details on 3-d modeling from images, some of these alternative approaches to capturing 3-d models are briefly reviewed.

1.1 Approaches to 3-d Modeling

For automatic 3-d modeling, besides the image-based approach, there are other competing technologies with their own advantages and limitations. Two general and prominent technologies are laser scanning and structured light scanning.

Laser scanner is a time-of-flight active scanner that uses laser light to probe the object surface. The scanner measures the distance to a surface point by timing the trip of the light pulse. Typical time-of-flight 3-d laser scanners can measure the distance of 10,000–100,000 points every second. Time-of-flight devices are also available in a 2-d configuration as a time-of-flight "camera", which can offer higher operation speed. For example, Zakhor's group at UC Berkeley have built a city-scan system with a 2-d laser scanner and a digital camera linked to high-end PC and additional electronics performing a complex timing synchronization [42], as shown in Fig. 1.1.

Structured light 3-d scanners are devices for measuring objects' 3-d shape using projected light patterns and cameras, as shown in Fig. 1.2. The light pattern can be generated with a laser interference device, a video projector [147], or an infrared light source [40]. The primary difficulty for the structured light system is the rapid attenuation of the light (except for the laser light source), and strong interference from environmental lights. These problems limit its application in scanning outdoor or large scale scenes. In summary, the structured light 3-d scanner offers a low-cost geometry capture device for indoor mid-size

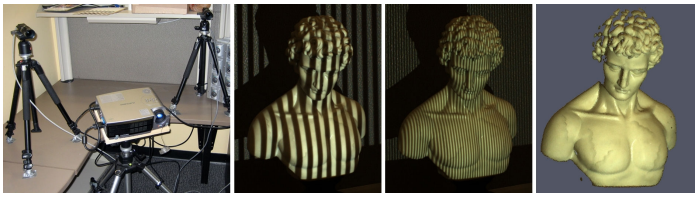


Figure 1.2: Capturing 3-d object with structured light [77]. From left to right: equipments (1 projector, 2 cameras), sparse structured light cast on the David bust, dense structured light, and reconstructed textured model

objects. It has been used to document archaeological artifacts for cultural preservation [10], human faces for motion capture [147], and human body motion tracking for controller-free video gaming [2], etc.

The above two approaches scan objects or scenes actively by sending out laser or structured light to the surface to be scanned and recording the reflected light. These approaches have the advantage of low computational cost and high accuracy. But their condition of applicability (including the ability to handle a variety of scenes, portability, and capture speed) is not comparable to image-based approach. For example, if strong light is used, it could be dangerous to eyes, which makes it forbidden in some applications; if weak light is used, rapid attenuation with distance limits the range of the scanning and downgrades the quality because of the interference of natural light. Image-based method, however, is an inactive approach, where cameras receive the light passively. This passive approach offers the advantage of more general applicability, faster capture speed, lower cost, etc, at the cost of higher computational load. Also nowadays, the equipments of the image-based approach, i.e., cameras, are prevalent. Consumer cameras can be used for large varieties of scenes and are very portable. Correspondingly, for the laser scanning and structured light, different kinds of equipments are needed for objects of different size, and, depending on the geometry of the scene, customization is usually needed. So image-based 3-d modeling is usually the preferred solution for 3-d modeling if satisfactory algorithms can be developed to accomplish its promise under normal working conditions. The characteristics of these three approaches are briefly summarized in Table 1.1.

1.2 Image-Based 3-d Modeling

Image-based 3-d modeling is a technique that generates a 3-d model of the scene from a collection of images taken around the scene, as shown in Fig. 1.3. The visual information in the collection of images is redundant in the sense that one spot in the scene can be observed

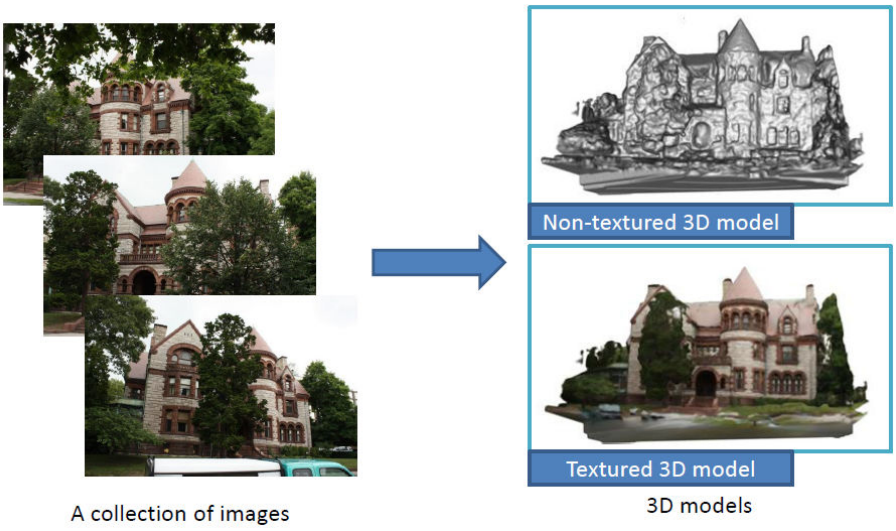


Figure 1.3: Illustration of the image based 3-d modeling problem's input and output

in serveral images. The image-based 3-d modeling is to explore the redundancy and create a compact representation that requires less storage and offers more freedom of navigation and control, useful to many applications.

1.2.1 Goal of Image-Based 3-d Modeling

Whata are the final goals of image-based 3-d modeling? Or what are desired performances for a "perfect" image-based 3-d modeling technique? Following is a (incomplete) list of important criteria for a successful image-based 3-d modeling algorithm.

Accurate Geometry: Accurate geometry is important for most applications.

Photo-Realistic Appearance: Photo-realistic rendering and novel view synthesis require accurate appearance modeling. It is important for many applications such as virtual reality, entertainment, etc.

Minimum User Interaction: User interaction should be minimized to allow fast and large scale deployment.

General Applicability: Natural scenes are of large varieties: indoor, outdoor, different surface materials, different lighting conditions, background clutter, etc. It is desirable that the technology can handle most commonly encountered scenes.

Simple Geometric Model: Image-based 3-d modeling tends to generate a model with a large number of vertices and triangles (if a mesh is used to represent the reconstructed 3-d surface geometry). In some applications, such as real-time large-scale visualization (e.g., city-scale visualization), it is desirable to have a simple 3-d model for fast visualization and easy post-editing.

Good Scalability: The algorithm should not only reconstruct a statue, but also a building, a campus or even a city; It should also be able to reconstruct objects to a very high resolution. These desired properties all require the algorithm to be scalable with respect to the size and the resolution of the scene.

Fast Speed: Now most image-based 3-d modeling algorithms are quite slow. They normally take hours to reconstruct a medium size object. It is desirable to improve the speed to minutes, or further to seconds.

Sparse Cameras: In some applications, for instance surveillance, it is costly to install a large number of cameras to monitor one spot. An algorithm that can work with a small number of cameras enables these kinds of applications.

1.2.2 Basic Concepts of Image-Based 3-d Modeling

In this section, the basic concepts of image-based 3-d modeling are briefly introduced.

BRDF: Bidirectional Reflectance Distribution Function BRDF is used to model surface material's reflective properties. When hitting a surface, the light will get absorbed, reflected and refracted depending on the surface material. Here the interest is on the reflected light, because this is what the image sensors detect. The BRDF is a 4-d function to model the relationship between the incoming light and reflected light, as shown in Fig. 1.4. The function takes an incoming light direction, ω_i, and outgoing direction, ω_o, both defined with respect to the surface normal n, and returns the ratio of reflected radiance existing along ω_o to the irradiance incident on the surface from direction ω_i. Note that each direction ω is itself parameterized by azimuth angle and zenith angle, therefore the BRDF as a whole is 4-dimensional. The BRDF was first defined by Edward Nicodemus [97] around 1965. The formal definition is

$$\rho(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{dE_i(\omega_i)} = \frac{dL_o(\omega_o)}{L_i(\omega_i)cos(\theta)d\omega_i} \tag{1.1}$$

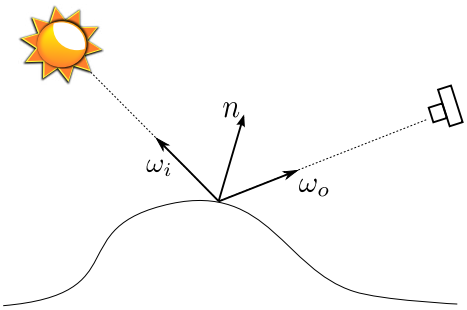


Figure 1.4: BRDF diagram

where $L_i(\omega_i)$ and $L_o(\omega_o)$ are the radiance along direction ω_i and ω_o respectively, E is the irradiance, and θ is the angle between incoming light direction ω_i and the surface normal $\mathbf{n}$.

Lambertian Reflectance If a surface exhibits Lambertian reflectance, light falling on it is scattered such that the apparent brightness of the surface to an observer is the same regardless of the observer's angle of view. More technically, the surface luminance is isotropic.

Lambertian reflectance can be taken as the DC component of the BRDF function, serving as the first order approximation. And in computer graphics, Lambertian reflection is often used as a model for diffuse reflection, and in computer vision, it is taken as the standard material model in most stereo work for its simplicity and nice invariance property.

Photo-Consistency Under the assumption of a surface being Lambertian, a point has the same radiance along all directions. This invariance is called the **photo-consistency** constraint, which is widely used in multi-view stereo.

Triangulation Triangulation is the operation of localizing a 3-d point, from its multiple projections (in different images), as shown in Fig. 1.5 for the case of 2 views.

Feature Matching Image feature detection and matching are important for both camera self-calibration and 3-d reconstruction. The features are designed to be (partially) invariant to affine transformation and illumination change. Popular features include SIFT [89], SURF [9], MSER [92, 98], etc. These features serve as image's sparse representation. Matching between images can be established through the matching of these features. The matched features provide constraints on the camera parameters,

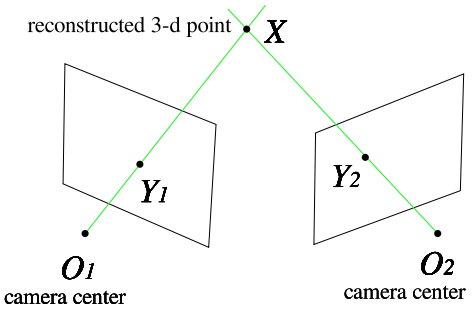


Figure 1.5: Illustration of triangulation

which are explored by self-calibration algorithms to calibrate the cameras, as explained below.

Camera Model In multi-view stereo, camera model specifies the parameters that define the 3-d to 2-d transformation. One commonly used simplified model is the pin-hole camera. It has 6 external parameters controlling the location T (a translation vector) and orientation R (a rotation matrix) of the camera, and 4 internal parameters – the focal length in pixels, the skew angle between x-y axis of the image plane, and the principal point on the image plane (2 parameters). The real cameras with lens cannot be exactly modeled with the pin-hole camera model, because of the lens distortions. The common lens distortions include radial and tangential distortions, which are usually modeled with a polynomial function with 5 parameters. The exact model has been documented clearly in many calibration tools, e.g. [15].

Camera Calibration Camera calibration is the process of determining the camera parameters, including camera focal length, camera axis skew angle, radial and tangential lens distortion, and camera location and orientation. For applications with fixed camera setup, calibration with known geometry, e.g., a checker board, can be used. One commonly used open-source toolbox is provided by Bouguet [15]. For applications with moving cameras, it is much more convenient to calibrate cameras from observed images. This process is called *self-calibration* (or auto-calibration), which can be performed through feature extraction and matching, and an iterative optimization routine, called bundle adjustment. One well-known software for doing that is Bundler [123], which offers an easy-to-use library of routines to do the job.

Geometry Representation In image-based 3-d modeling, generally speaking there are

three widely used geometry representations: point cloud, surface and occupancy volume. A point cloud is only a partial representation of the geometry, where the point connection information is missing. Surfaces can be reconstructed from a point cloud with some surface reconstruction algorithms, but possibly with wrong connections and holes left. The mesh structure is a commonly used discrete representation of a surface, and is widely used for rendering. An occupancy volume representation models space occupancy directly, and has regular neighborhood structure.

1.2.3 Difficulties in Image-Based 3-d Modeling

As discussed before, if pixels can be matched among images, the corresponding 3-d points can be reconstructed with triangulation easily. So if given a good-quality stereo matching algorithm, a full 3-d model can be easily constructed by pair-wise stereo reconstruction. In this sense, multi-view stereo seems easy. However it is not as easy as it seems. There are many difficulties associated with multi-view stereo.

Ambiguity of Matching Multi-view stereo is based on photo-consistency matching across images. But when an image region is (almost) homogeneous, each pixel in the region may match to many pixels in the other image due to the ambiguity of matching. Most algorithms will discard this region and leave a hole in the surface. More complete reconstruction of the ambiguous regions can be estimated based on the visibility constraints and regularity of the surface geometry.

Occlusion In complex environments, there are strong occlusions between objects and within objects (self-occlusion). While the effect of occlusion is not obvious in small-baseline stereo, it becomes important for the accuracy and completeness of the reconstruction in multi-view stereo, where the baseline between two views can be large.

Surface Topology The topology of natural scenes can be very complex, e.g, flowers, trees, compared with planar buildings. For model-based reconstruction methods, the model should be updated correctly to adapt to the surface topology. This operation could be a challenge for mesh-based methods, because topological surgery of mesh is a difficult job. It is also challenging for the meshing algorithm to reconstruct a surface with complex topology from unconnected points.

Non-Lambertian Material There are many surfaces not being Lambertian in common world scenes, e.g., cars, plastics, windows, statues, etc. To reconstruct objects in this category, specular reflectance needs to be modeled or handled in an appropriate way.

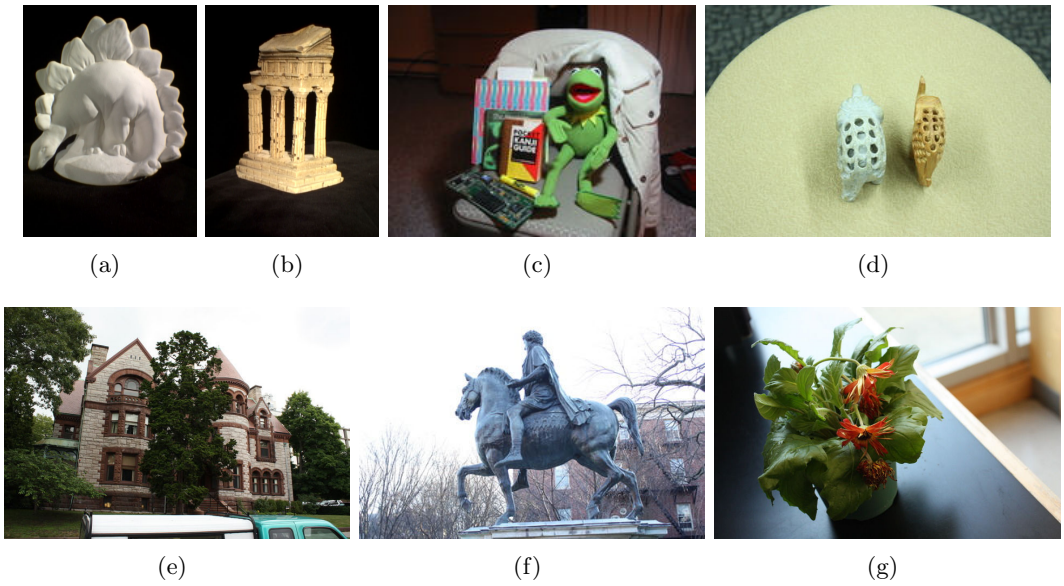


Figure 1.6: The variety of scenes that image-based 3-d modeling needs to deal with: indoor (a–d) vs. outdoor (e,f), small (a–e) vs big (e–g), object of simple topology (a,b) vs. object of complex topology (d, e, g), Lambertian objects (a,b,c) vs. object with reflective surface (f,g), images captured with cameras mounted on robot arm (a,b) vs. images captured with a hand-held camera (c–g).

Background Clutter Many algorithms require an initial good surface estimate, which is usually got from object silhouettes. In the presence of background clutter, it is not easy to extract silhouettes reliably.

Transient Clutter Most multi-view stereo algorithms assume that objects are static during multi-view image capture. (Dynamic scenes can be modeled in the same way with a synchronized multi-camera system, since at each moment the cameras are capturing the same static moment of the scene). However in many cases, while capturing the photos, moving objects come and go, such as pedestrians on a street, or visitors in a museum. They are usually treated as transient clutter. These clutter can completely confuse the reconstruction algorithm or create ghost effects.

The above difficulties have driven people to investigate different approaches to multi-view stereo, which will be reviewed in the next chapter.

1.3 Statistical Inverse Ray Tracing Approach

In the last section, the main difficulties in image-based 3-d modeling problem are exposed. In this thesis, these challenges are tackled from a novel perspective. Different from most of the other approaches which use the pixel matching – triangulation pipeline, the approach adopted here starts from modeling the image generation process, given that cameras are calibrated. Then the goal is to find the best 3-d model that can generate rendered images as similar as these observations. That is, image-based 3-d modeling is treated as an inverse problem. Instead of directly solving the problem, all the constraints are first modeled. Then the problem is to find the best model that satisfies all the constraints. In this process, the probabilistic modeling and prior knowledge about the scene help to model and reduce ambiguities.

1.3.1 Image-based 3-d Modeling as an Inverse Problem

There is a similar problem involving 3-d models and images in computer graphics, but in opposite direction — 3-d rendering. 3-d rendering is the process of generating (realistic) images given a 3-d scene (including geometry, appearance and light), which is exactly the opposite operation of multi-view stereo. 3-d rendering has been highly developed and is commercially successful in entertainment (film, games), visualization, simulation, etc. Since multi-view stereo is the inverse problem of 3-d rendering, the advances in 3-d rendering can be leveraged to further the development in 3-d reconstruction. One exciting development of 3-d rendering is ray tracing, where photo-realistic rendering can be generated by tracing each ray from the observer to object and then further to the light source. In this study, it is shown that the concept of ray tracing also plays essential role in multi-view stereo.

1.3.2 Important Role of Probability and Prior Knowledge

Multi-view image-based 3-d modeling, as an inverse problem, is ill-posed, with many local ambiguities to be resolved at a larger scale (globally or semi-globally). The inverse problems can be cast as a Bayesian inference problem, where prior knowledge can play an important role in resolving ambiguities [12, 132, 55]. The probabilistic setting also helps to resolve conflicting factors that pose difficulties for hard decision approach. Probability provides tools to model ambiguities and incomplete information. Multi-view images often do not have enough observation to cover every spot of the scene, because of occlusion. The probabilistic approach can make an inference about the incomplete regions with an measure of reliability. In summary, probabilistic modeling and inference provide good tools for

modeling and solving ambiguities, conflicting information, incomplete information and regularization. Moreover, there is a rich set of tools in statistical analysis that can be borrowed to solve challenging problems in image-based 3-d modeling. In this thesis, statistical inference and machine learning are leveraged to help solve some hard problems in image-based 3-d modeling, such as occlusion.

1.4 Main Contributions

The main contributions of this thesis can be briefly summarized as following.

1. The multi-view stereo problem is formulated as solving a one-step optimization problem. The optimization problem is defined by a novel Ray MRF model, combining ray constraints and prior model.

2. A statistical inference algorithm of linear compuational complexity for solving the Ray MRF inference problem is developed, based on loopy belief propagation and dynamic programming.

3. The Ray MRF formulation of the problem models the occlusion relationship accurately and the estimation algorithm can recover strongly-occluded regions well as shown in the experiments.

4. The developed algorithm can directly produce an accurate and complete textured surface easily without resorting to other meshing (aka, surface reconstruction) methods.

5. A model that can handle the transient clutter is proposed in the Ray MRF framework, which can handle non-static objects, reflective highlights and other dynamic or viewpoint-dependent phenomena.

6. The deep belief propagation algorithm is extended to handle the inference of a broader class of higher-order MRF models.

The proposed algorithm is shown to be capable of generating high-quality 3-d models for complex objects under challenging conditions. The main limitation of the current work is the high computational and memory cost by using an evenly-divided volume, which could be addressed in the future with sparse volumetric representations. The last chapter provides a more detailed summary of the proposed approach's advatanges and shortcomings. As a side note, parts of the thesis material have been presented in conferences, including [83] and [84].

1.5 Thesis Organization

The rest of the thesis is organized in the following way. First the state of the art approaches to image-based 3-d modeling are reviewed in Chapter 2. The idea of inverse ray tracing is introduced in Chapter 3. Then the corresponding mathematical formulation and graphical representation — Ray Markov Random Field (Ray MRF) — is defined in Chapter 4. To solve the hard inference problem in Ray MRF, an extension of belief propagation — deep belief propagation (DBP) — is developed for the ray clique, and is used for voxel occupancy and color estimation 5. To handle various varieties of scenes, other components including background modeling and transient clutter handling are studied in the same framework in Chapter 6. Quantitative evaluation of the algorithm on standard benchmark datasets and more challenging datasets on large variety of scenes is discussed in Chapter 7. As an extension of the deep belief propagation algorithm for the ray clique, general DBP algorithm for higher-order MRFs is discussed in Chapter 8. The last chapter summarizes this work and discusses some possible extensions.

	laser scanning	structured light scanning	image-based modeling
device	laser scanner + camera	projector + camera	camera
mobility	hard, usually mounted on a car or airplane	not very mobile	very portable
operation condition	indoor and outdoor with different devices	mainly indoor	both indoor and outdoor
device price	$10000+	$500+	$100+
accuracy	high	high	reasonably high[a]
appearance?	need to register appearance with geometry	need to register appearance with geometry	get appearance for free
capture speed	slow	medium	high
computational cost	low	medium	high
drawback summary	expensive device, not very portable and capture speed is slow	mainly restricted to indoor geometry capture	computational cost is high

Table 1.1: Comparison between laser scanning, structured light scanning and image-based modeling

[a]With the improvement of algorithms, the accuracy of this approach has been improved a lot in the last 5 years and is approaching the accuracy of laser scanner [43], and it will continue to improve.

Chapter 2

Literature Review

Before diving into the details of the proposed approach, a brief overview of the existing works on image-based 3-d modeling is sketched. The focus is put on the representative state-of-the-art works. These representative works are categorized according to the adopted 3-d geometry representations. At the end, recent developments on camera calibration are briefly reviewed. The review is not aimed to be comprehensive, but focused on sketching the key developments in this field. Other reviews, e.g., [117][43], have provided more comprehensive accounts.

2.1 The State-of-the-Art Approaches to Image-Based 3-d Modeling

As discussed in Chapter 1, there are three commonly used 3-d geometry representations in image-based 3-d modeling: point cloud, surface and volume. Based on the past decades of research, it is commonly agreed that 3-d representation is a key factor that dominates the mathematical formulation of the problem, and the corresponding algorithms' ability to handle various kinds of difficulties in multi-view stereo reconstruction [117]. Here the advantages and disadvantages of each approach are discussed, with its representative state-of-the-art work. Table 2.1 briefly summaries the characteristics of each representation.

2.1.1 Feature-Based Point Cloud Reconstruction

The feature-based reconstruction consists of several steps: feature extraction, feature matching and triangulation. The output of this procedure is a set of unconnected points, called a point cloud. Merits of this approach include 1) image matching has a large body of existing research to draw upon; 2) in the pipeline based procedure, each step can be studied,

	point cloud based approach		surface evolution approach		volumetric approach		
representative approaches	depth fusion	feature expansion	mesh evolution	level set evolution	space carving	voxel world model	this work
information explored	invariance		photo consistency		photo consistency	ray constraint	
2-d – 3-d relationship	matching – triangulation		projection		back projection		
method	pipeline based		iterative		iterative		
topology	any topology		fixed topology	any topology	any topology		
occlusion handling	point cloud filtering		local visibility		local visibility	global visibility	
surface regularization	no		yes		no		yes
optimization	N/A		gradient descent		N/A	local Bayesian updating	approximate global optimization
main limitation	does not work well for wide baseline		need a good initial surface estimation		greedy solution no regularization	local algorithm no regularization	N/A

Table 2.1: Characteristics of different approaches to multi-view stereo

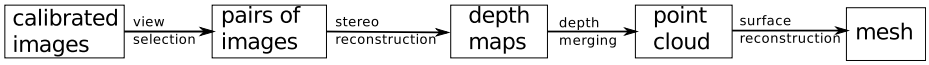


Figure 2.1: The pipeline of the depth fusion approach

implemented, and tested individually; 3) different combinations of these steps result in new algorithms, which have been studied and evaluated thoroughly; 4) and because the point cloud representation is computationally simple to process, it can handle high-resolution images. The disadvantages of this approach include 1) image feature matching limits the applicability of the approach to textured objects only; 2) there is no formal treatment of occlusion; 3) it cannot handle large baseline camera configurations robustly; 4) and the image information cannot be utilized optimally for two reasons: firstly, only a subset of the images are used for each point reconstruction, and secondly, the information is processed sub-optimally, because an early decision is usually made in every step.

Here, two representative approaches in this category are reviewed. One is the depth fusion approach, which is the direct extension of stereopsis. Another is the feature expansion approach, which follows a sparse to fine search/matching scheme.

Depth Fusion

The depth fusion approach handles the multi-view stereo problem as a direct extension of stereo vision. The view selection step selects pairs of images with an appropriate baseline, from which depth maps are generated with the stereo algorithm. Then the depth maps are merged to remove redundant and noisy points. The output of this depth merging procedure is a clean point cloud, from which a surface can be reconstructed with a surface reconstruction algorithm. This approach is simple and has been shown to be quite effective [126, 17, 20, 87, 53, 81]. The pipeline of the depth fusion approach is illustrated in Fig. 2.1.

Feature Expansion

The feature expansion approach, represented by the work of Furukawa and Ponce [47, 48], is another successful feature-based point cloud reconstruction method. Feature expansion approaches start from a set of sparse matched features, then propagate the matching to their neighbor pixels, and until all the pixels are processed. In [47, 48], a variant of point cloud representation, namely the patch representation, is used. A patch is an oriented planar circular region in 3-d, which can be projected to different images in order to check

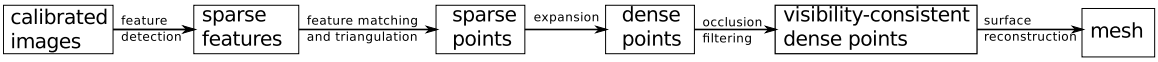


Figure 2.2: Furukawa's multi-view stereo pipeline

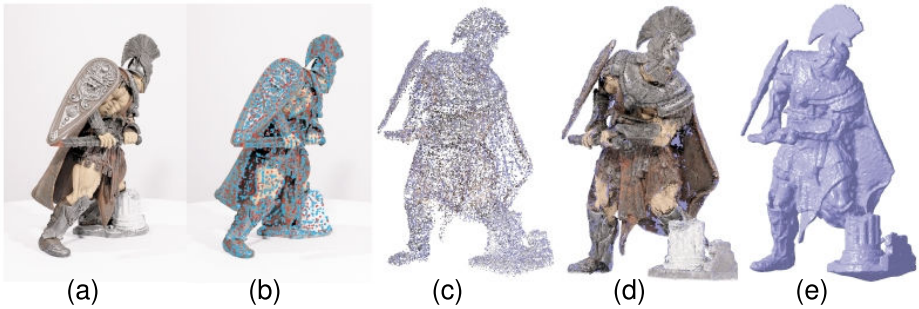


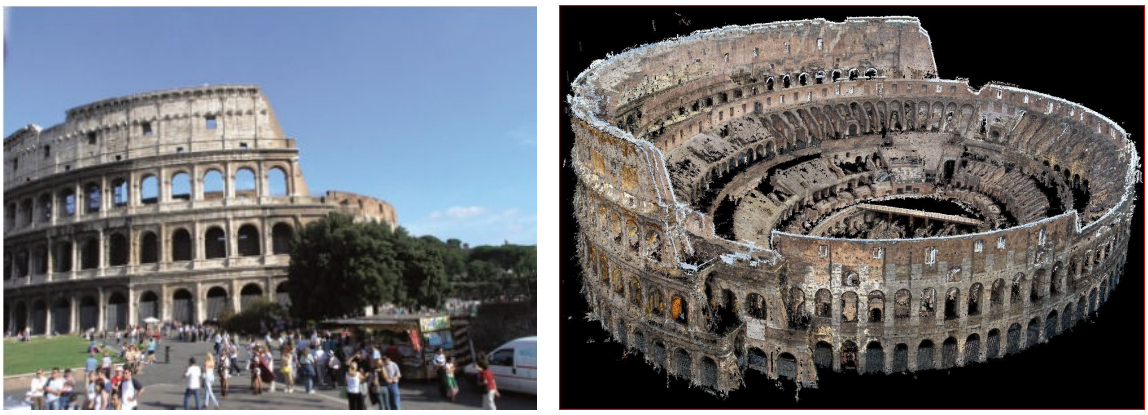
Figure 2.3: Working example of Furukawa's multi-view stereo pipeline: (a) example input image; (b) detected sparse features; (c) reconstructed sparse points; (d) filtered dense points; (d) reconstructed mesh model

their photo-consistency. There have been several works reported on using 3-d patches to represent the 3-d scene [27, 113, 72]. [54] adopted a similar strategy in the community photo collection, where a photo-selection scheme was introduced to select good candidates for photo-consistency checking. In [47] and [48], Furukawa and Ponce adopted the patch representation within the feature expansion reconstruction pipeline and showed the flexibility and accuracy of the representation in handling matching and occlusion. The pipeline is summarized in Fig. 2.2, and its working procedure is demonstrated in Fig. 2.3. In [44], Furukawa et al. extended this approach to reconstruct large scale scenes with a large number images, as demonstrated in Fig. 2.4, by clustering images based on their visibility and processing each cluster independently.

Besides the popular depth fusion approach and the feature expansion approach, there are other variants of the feature-based point cloud approach. For example, instead of performing feature point matching, image edges can be matched between images to create 3-d curves. This representation is an extension of the point cloud representation in the sense that it connects the points along the curve [85, 33, 35].

2.1.2 Iterative Surface Evolution

The iterative surface evolution approach adopts a surface representation of the scene, either explicitly with a mesh or implicitly with a level set function. This approach starts from an initial surface, then iteratively refines it to increase its consistency with observed



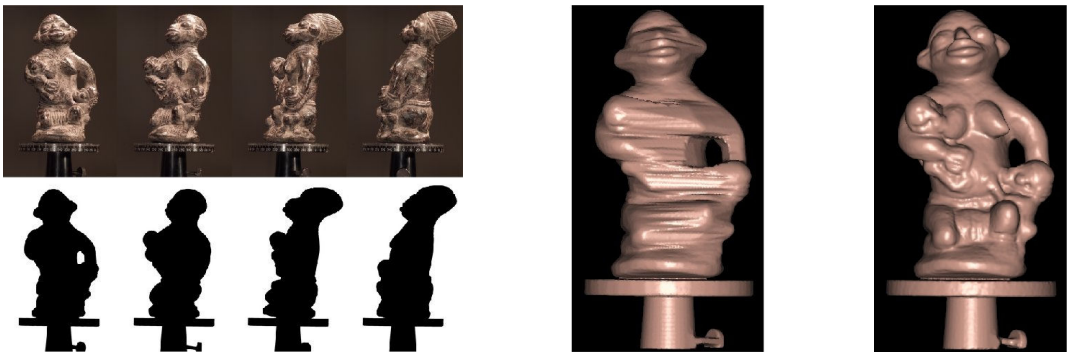
(a) One of 490 input images (b) Rendering of the reconstructed patch model

Figure 2.4: Representative reconstruction result from the state-of-the-art feature expansion approach [44]. Reconstruction of Rome Colosseum from 1167 images collected from Internet (after image selection, 490 of them are used). The model has 5,747,083 points.

images. The convergence of the local refinement highly depends on the quality of the initial surface, which has become the main disadvantage of this approach. An initial surface, which should be close to the ground truth, is required. The commonly used initial surfaces include the visual hull from object silhouettes or a coarse surface generated from a quasi-dense point cloud. To extract silhouettes, the object is usually put on a turn table with uniform surrounding background, making the segmentation easy. While making the problem easier, requiring the silhouettes severely limits the approach to the indoor small object reconstruction. It cannot reconstruct outdoor objects and large objects.

Mesh Evolution

Mesh surface representation is widely used in computer graphics for rendering, animation, CAD model design, etc. So it is a natural idea to adopt the same surface representation in 3-d reconstruction. The mesh evolution approach starts from an initial mesh, and evolves it according to the photo-consistency measure between the projected image and the observed image. Example works in this category include [34, 46, 32]. Fig. 2.5 shows a typical working example of this approach [34]. One key problem of the mesh evolution approach is the hardness to control the mesh topology and density. In [109], Pons and Boissonnat proposed a robust and efficient approach for updating surface meshes undergoing large deformation and topology changes. In [74], Labatut, Pons and Keriven built an initial triangular mesh from quasi-dense points generated by matching keypoints across images. Then the mesh



(a) 4 out of 36 input images and silhouettes (b) Visual hull reconstructed from silhouettes, serving as the initial mesh (c) Refined mesh

Figure 2.5: Representative results from one of the state-of-the-art mesh-based surface evolution approaches [34]

is refined through photo-consistency optimization. In [31], Delaunoy et al. combined the above two methods and made the updating equation more mathematically vigorous. Similar work includes [17, 21, 46]. Vu, Pons et al. [58] extended the methods to make it capable of handling large scene reconstruction, as shown in Fig. 2.6. One good property of this approach is its easiness to add surface regularization on the mesh model. For example in [32], a Laplacian tangential smoothing operator is developed, which can accurately preserve sharp features.

Level Set Evolution

Although there are efforts to remedy the topology difficulties in mesh based representation as discussed above, it is still a hard problem. The level set based implicit surface representation overcomes this problem by embedding the surface into a 3-d volume. In this way, the change of topology corresponds to a continuous change in the level set values [120][100].

In [36], Faugeras and Keriven introduced the photo-consistency variational principle, from which the Euler-Lagrange equation can be deduced from the variational energy functional. Then a set of PDEs, deforming an initial surface towards the desired surface, can be constructed. The PDE can be implemented with the level set representation naturally. In the original paper, only 2-d illustrative examples are provided. After that there have been lots of follow-up work to address the computational and implementation issues [145, 115, 61, 86]. In the level set representation, shape priors can also be naturally expressed and incorporated in the optimization formulation [114].

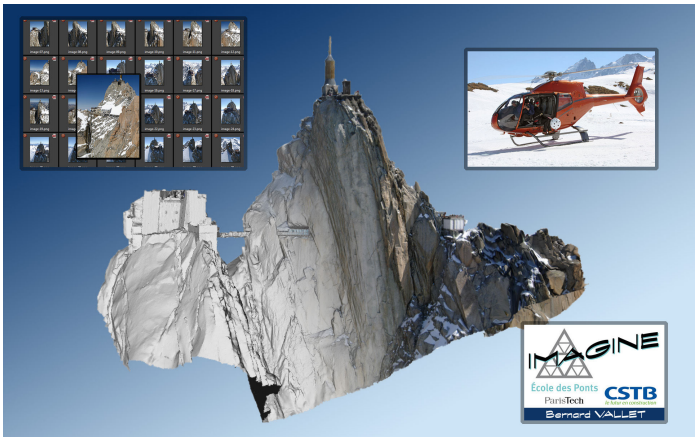
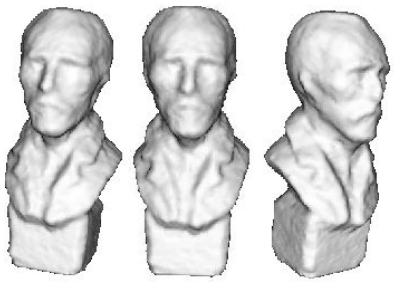


Figure 2.6: Representative reconstruction result from one of the state-of-the-art mesh-based surface evolution approaches [58]. Reconstruction of Aiguille du Midi (France) from 53 images with 5M pixels taken from a helicopter. The model has 4M triangles.

(a) 4 out of 281 input images



(b) three views of the reconstructed 3-d model

Figure 2.7: Representative reconstruction result from one of the level-set surface evolution approaches [60]. Reconstruction of Van Gogh statue, made of polished metal with strong specular highlights, from 281 images (shown on the left are 4 typical images). Silhouette information is used.

Figure 2.8: Representative result from one of the state-of-the-art graph-cuts-based volumetric reconstruction methods [136]. On the top is 4 out of 140 input images of a fully textured toy house; reconstruction of the model rendered with the same view angle. In the reconstruction, manual segmented silhouettes are used.

2.1.3 Volumetric Reconstruction

Volumetric reconstruction works with occupancy volume directly. Based on how to handle the visibility, the work in this approach be categorized into two groups: one relies on an initial surface to compute the visibility, represented by the work on using graph cuts to get the best surface [125, 136], another does not assume any prior visibility knowledge, represented by the work on space carving [118, 119, 73, 19] and voxel world model [106, 30].

In [125], Snow et al. used graph cuts (exact graph cut, since the label is binary) to estimate voxel occupancies, which is probably the first work on using graph cuts in volumetric reconstruction to our knowledge. The energy defined was simple, and the experimental result looks very preliminary. In [137, 136], Vogiatzis et al. used visual hull of the scene to infer occlusions and as a constraint on the topology of the scene. A photo consistency-based surface cost functional is defined and discretized with a weighted graph. The optimal surface under this discretized functional is obtained as the minimum cost solution of the weighted graph. The method provides a viewpoint independent surface regularization, approximate handling of occlusions based on an initial surface and a tractable optimization scheme. [133]

Different from the above works on using approximate visibility estimation, space carving and its variants handles visibility more vigorously. Since they are closely related to this work, detailed reviews are provided here.

Space Carving: In [118, 119, 73], Seitz, Dyer and Kutulakos proposed the seminal work

Figure 2.9: Representative result from the probabilistic space carving method by Broadhurst [18]. (a) Four out of 11 input images. These clean images are manually segmented from a cluttered image with trees and other building in the background. (b, c) Rendering of the reconstruction.

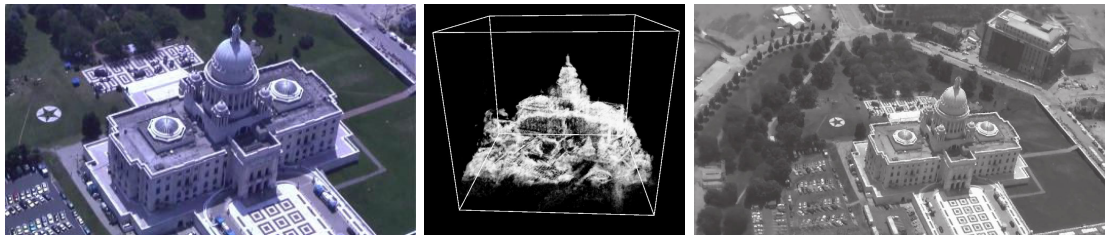


Figure 2.10: Representative result from the voxel-world-model method by Pollard, Crispell and Mundy [106, 105, 30]. On the left is one out of 255 input aerial images extracted from a 720p video; in the middle is a volumetric rendering of the reconstruction of the Capitol building; on the right is a novel view synthesized from the reconstructed volume.

on space carving (aka, voxel coloring), a matching-free volumetric approach, which can cope with large camera baseline. The method takes full account of occlusions by carving out non-photo-consistent voxels from outside to inside with a special order to satisfy the visibility constraint. The photo-consistency test is controlled with a single threshold parameter. "An overly conservative (small) threshold value results in an accurate but incomplete reconstruction. On the other hand, a large threshold yields a more complete reconstruction, but one that includes some erroneous voxels. In practice, thresh should be chosen according to the desired characteristics of the reconstructed model, in terms of accuracy vs. completeness." The dilemma between accuracy and completeness consists of the main drawback of this approach. And in practice, the geometry accuracy of the reconstruction is fuzzy and not as good as the state-of-the-art point cloud or surface evolution approaches.

Probabilistic Space Carving: To remedy the accuracy and completeness dilemma, Broadhurst et al. [19][18] for the first time introduced the probabilistic framework for space carving. The key problem turns to computing the probability of voxel's existance, which is estimated from the data with Bayes' theorem. This approach eliminates the need for a global threshold parameter and eliminates the possibility of carving large holes in the model, as in the space carving algorithm. In this work, the ray concept (the connection between a pixel and the set of voxels that the pixel's viewing ray passes through) is introduced to compute the visibility of a voxel along a particular direction. To compute the probability of voxel's existance, in principle it is necessary to integrate over all the possible visibility configurations. This integral is very slow as it has complexity $O(2^n)$, where n is the number of images. The proposed algorithm is based on some approximation — rather than intergrating over all possible visibility configurations, a local threshold is introduced to avoid the integration operation by taking a greedy solution. Fig. 2.9 shows a typical experimental result of this approach from [18]. It can be seen that it still suffers from the problem of fuzzy geometry.

Voxel World Model: In [106], Pollard and Mundy proposed the pioneering work on full Bayesian updating of voxel's surface probability. In this work, the ray constraint, which connects each pixel with the voxels that the corresponding viewing ray passes through, plays a central role. And the voxel updating along each ray is carried out efficiently. It uses mixtures of Gaussians to model the dynamic 3-d scene in the change detection application. The online nature of the algorithm serves the change detection application well. However, when used as a 3-d reconstruction algorithm,

because it is only guaranteed to satisfy the constraints of the current frame being processed, it does not provide an optimal solution that satisfies all the constraints imposed by all the frames. In [30], Crispell further developed this approach to handle large scene reconstruction and to enforce the global constraints from all the frames, based on the voxel independence assumption. First Crispell extended the algorithm to reconstruct an octree-based sparse volume, which can represent larger scene with less space requirement. Secondly, Crispell developed a batch version of the Bayesian voxel updating scheme to enforce the global constraints.In practice, to make the algorithm converge, the estimator needs to be damped based on an empirical formula, in order to prevent wild-jumping and oscillating. Fig. 2.10 shows a typical reconstruction result from [30]. It can be seen that the synthetic view looks fairly good, while many surface regions (such as building boundaries) are not well localized.

In the above, three commonly used geometric representations in image-based 3-d modeling and their representative state-of-the-art algorithms have been reviewed. As a summary, each approaches have their advantages and weakness. As a combination of the existing methods' vitures, the ideal solution should have the following properties:

- It can generate photo-realistic 3-d textured models;
- It can produce accurate 3-d geometric models;
- It does not require any additional information, such as object silhouettes;
- It can handle object of complex topologies;
- It can work for both indoor and outdoor scenes.

In this work, the volumetric geometry representation is used, for the following reasons:

1. Occupancy volume can represent wide kind of geometry information. It can not only represent surface point location and connection information, but also the volumetric occupancy information. For example, because of the ambiguities, the surface may not be localized well everywhere, but volumetric approach can still provide us information about occupancy, which is important for navigation and virtual reality.
2. Volume representation has regular spatial neighborhood structure and can handle objects of arbitrary topology. It is for this reason that volume has been taken as an intermediate surface representation in many applications. For example, the popular

meshing algorithm, PoissonRecon [65], uses volumetric representation as an intermediate representation to convert point cloud to mesh. As another example, many mesh operations, such as mesh simplification, re-meshing, use the volumetric representation for its easiness of topology handling and regular spatial structure. So it is natural to use the volumetric representation at the beginning to directly model the surface, instead of resorting to surface reconstruction algorithms to convert a point cloud to a surface, where information could be lost in the intermediate steps.

3. The large memory and computation cost of the volumetric representation can be alleviated with the octree-based sparse representation. For example, this representation has been used in [65] and [30], to solve similar memory and computation problems.

2.2 Camera Calibration Development

As a critical step of image-based 3-d modeling, camera calibration has been steadily improving in its accuracy, robustness and scalability. Broadly speaking, there are two methods: marker-based calibration and self-calibration, differing on whether geometry-known markers are used. For the marker-based calibration, in 1987, Tsai [134] proposed the first marker-based camera calibration algorithm; in 2000, Zhang presented an easy-to-use calibration method, based on a planar checker board [148], which has become the de facto standard marker-based calibration method with an easy-to-use open source implementation provided by Bouguet [15]. However, in many applications, it is inconvenient or even impossible to put markers in the scene, especially a large scale scene. Much study has been done on camera self-calibration based on the images themselves [37, 108, 107]. In 2006, Snaveley et al. presented a system that can calibrate thousands of images automatically [124], with their Bundler software released in open source [123]. In this work, the Bundler software is used to auto-calibrate the associated cameras for each image in the experiments. What is the accuracy of the self-calibration procedure? From other people's work based on Bundler [3, 44, 54] and this work, the calibration error seems to be less than 2 pixels for a 12 mega-pixel digital camera (Canon EOS Rebel XSi Digital SLR)[1]. The calibration accuracy can be checked with the epipolar constraints, as shown in Fig. 2.12. This accuracy is good enough to get a high-quality 3-d reconstruction.

[1] There are cases for which the Bundler's camera calibration is off a lot from the ground truth. But usually this can be easily detected by looking at the camera location and the sparse point cloud generated by the Bundler software. And it can be further verified by the epipolar constraint, as shown in Fig. 2.12.

Figure 2.11: Representative result of the state-of-the-art self-calibration algorithm [124]. Visualization of the reconstructed point cloud and cameras produced by Bundler (Trevi Fountain dataset).

(a) camera poses recovered from images

(b) left image and picked points

(c) right image and corresponding epipolar lines

Figure 2.12: camera pose visualization and epipolar-line checking between two images in the horse dataset. The images shown are cropped to highlight the picked points. (best viewed in color)

Chapter 3

A Statistical Inverse Ray Tracing Approach to Multi-View Stereo

Cameras are devices for recording incoming lights. To decode the light signals to recover the physical world's intrinsic properties, such as geometry, material, lighting, etc, the first step is to understand the encoding process, i.e., how the light interacts with the physical world's geometry and material. Ray, a line along the light's path of propagation, plays a central role in optics because most optical phenomena can be explained with recourse to a single ray only and there is little interaction among rays. Each ray starts from a light source, then hits some surface, and is reflected, refracted, or absorbed. The reflected and refracted light will continue its journey, hitting another surface, and so on, until it gets totally absorbed by the environment or the eye. Because most light get absorbed by the environment, the simplest way to simulate light transportation is to trace the ray backwards from the eye to the surface, further to the light source, and not the other way around (see Fig. 3.1). This is the basic concept of ray tracing. Since light ray connects the scene properties to pixel observations, can the scene's intrinsic properties be recovered by inverting the ray tracing process? This chapter explores this idea in details.

3.1 Review of Ray Tracing

In computer graphics, ray tracing is an algorithm for rendering 2-d images from a 3-d scene by tracing the path of light. Ray tracing is capable of simulating a wide variety of optical effects, such as reflection, refraction and scattering, making it capable of producing photo-realistic images. In ray tracing, at each point of intersection between the ray and surface,

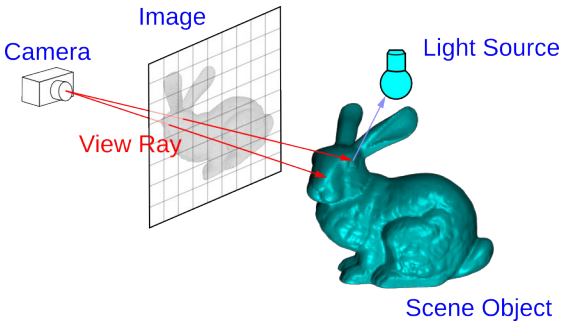


Figure 3.1: Illustration of ray tracing

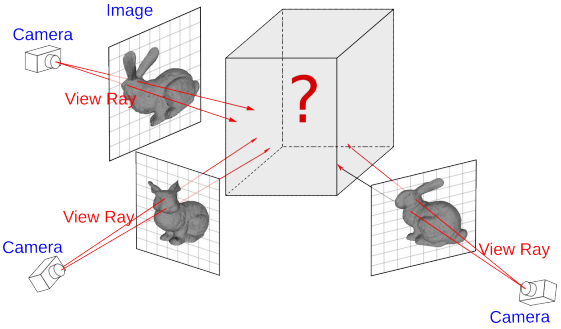


Figure 3.2: Illustration of inverse ray tracing

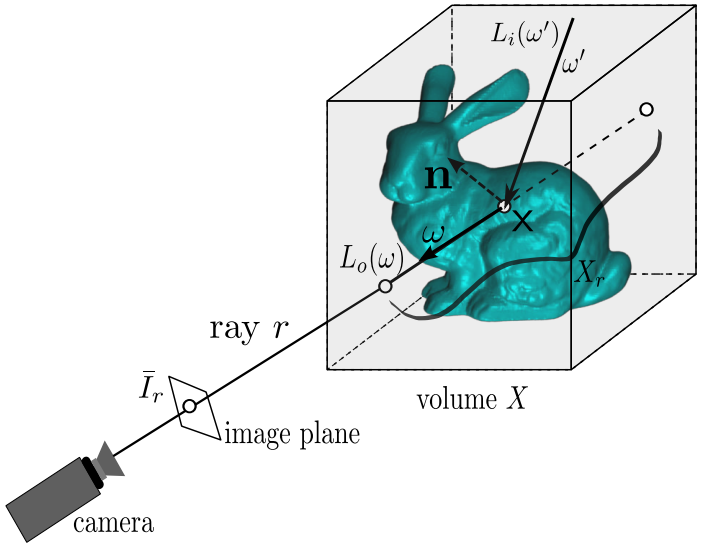


Figure 3.3: The interaction amongst the light source, surface geometry and surface material in ray tracing

the reflected amount of light along the ray needs to be computed based on the shading model.

3.1.1 The Rendering Equation and the Shading Model

Assume a viewing ray r intersects the surface $\mathcal{S}$ at a point x. The rendered pixel intensity value $\bar{I}_r$ of the ray r is proportional to the amount of light the pixel receives, or equivalently, the amount of light reflected from the visible surface point x towards direction ω, where ω is the reverse direction of ray r. The above geometry relationship is illustrated in Fig. 3.3. Formally,

$$\bar{I}_r = \gamma L_o(\omega), \tag{3.1}$$

notation	meaning
r	viewing ray, which starts from the camera center and goes through a pixel
R	the set of viewing rays
X	the set of voxels in a volume
I_r	intensity value of the pixel where the viewing ray r originates from
$\bar{I}_r(X)$	the rendered pixel intensity value of the viewing ray r
$\Phi_r(X)$	the volumetric rendering equation for the viewing ray r, given volume X
X_r	the set of voxels through which the ray r passes through The voxels are ordered in the viewing ray traversal order
x	a voxel in discrete volume
x_r	a voxel in X_r, i.e., $x_r \in X_r$
x^o	occupancy of a voxel x
x^c	color intensity of a voxel x
X_r^o	occupancies of the voxels on ray r
X_r^c	color intensities of the voxels on ray r

Table 3.1: Notations in inverse ray tracing

where $L_o(\omega)$ is the radiance of light reflected towards direction ω, γ is a scale constant. With suitable units, γ is set as 1. $L_o(\omega)$ is a function of environmental lighting and surface material, modeled by the rendering equation [62]:

$$L_o(\omega) = \int_\Omega \rho(\omega, \omega') L_i(\omega')(-\omega' \cdot \mathbf{n}) d\omega' \tag{3.2}$$

where $\rho(\omega, \omega')$ is the Bidirectional Reflectance Distribution Function (BRDF) of the surface material at x, which measures the proportion of light reflected from ω' to ω, $L_i(\omega')$ is the radiance of light from direction ω', $\mathbf{n}$ is the normal of the surface at point x, Ω is the hemisphere of inward directions. Literally, Eq. (3.2) integrates the reflected lights of the incoming light from all direction at a surface point.

Eq. (3.2) exactly models the light propagation at a point. In ray tracing, the incoming light from direction ω' is further traced to another intersection surface point or light source. If the ray hits another surface point, the same operation will continue recursively. This process models all the lights to one surface point x directly from light source or indirectly from the light reflection among surfaces. This is called the global illumination model. It is usually computationally costly. As an approximation, the local illumination model does not account for the inter-reflection between surfaces, instead it assumes that the light from direction ω' comes directly from the light source. To compensate for inter-reflection lighting, the pre-computed spherical harmonics can be used to replace the original lighting.

Another computational barrier is the material BRDF, which is a general four-dimensional

function. Several parametrized approximations of BRDF, called shading models, have been developed in computer graphics to simplify it. Lambert model and Phong model are two commonly used ones.

Lambert Model In the Lambert shading model, the reflection property of the material is assumed to be purely diffusive. That is, the light reflected from the surface is equal in all directions. So it does not matter where the eye is. The diffusive property of surface material is measured by parameter κ_d. The radiance of the reflected light L_d can be computed based on Lambert's law, which is an approximation of the rendering equation (3.2):

$$L_d = L_i(\omega')cos(\theta) = L_i(\omega')(-\omega' \cdot n)), \tag{3.3}$$

where θ is the angle between the incoming light direction and the surface normal

In the Lambert model, $\int_\Omega \kappa_d * (-\omega' \cdot n)d\omega'$ is a constant for a surface point. Interpreted in another way, the observation of the surface point is the same in any viewing direction. So from the multi-view images, it is impossible to separate the surface's material (intrinsic color) and the lighting. In multi-view stereo, this is called the surface observed color under the given lighting condition, or simply the surface color. Then voxel colors can be defined similarly. Lambert model is the simplest shading model. In this study, as the first step to full inverse ray tracing, the Lambert model of the scene is assumed. Extension to more general shading models and global illumination is discussed at the end of this chapter by building connections between this work and other researchers' work on different but related problems in inverse ray tracing.

Phong Model In the Phong shading model, the illumination of a point is composed of three components: ambient illumination, diffuse reflectance, and specular reflectance. These three components are represented with three coefficients: κ_a, κ_d and κ_s respectively. The ambient component accounts for the light scattered by the environment; this component accounts for a simple approximation of global illumination, and frequently assumed to be constant anywhere. The ambient component is independent of light position, object orientation, observer's position or orientation. The diffusive reflectance is the same as the Lambert model. The specular component models the highlight spot on the object. This component is dependent on the viewing angle, surface normal and incoming light angle. How much received reflection depends on the receiver's location and orientation. The specular component is computed as $L_s(\omega, \omega') = \kappa_s L_i(\omega')cos^\alpha(\psi)$, where ψ is deviation of the view angle from mirror direction, α measures the degree of the material's similarity to a mirror. When ψ is

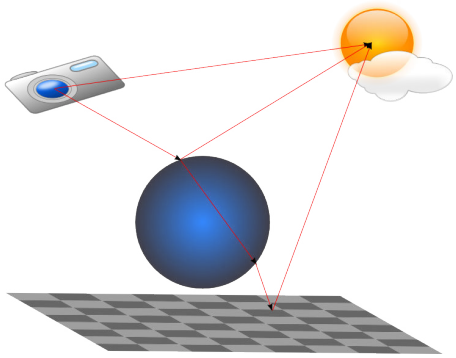


Figure 3.4: Illustration of ray traversal path

small, the specular light component is strong.

$$\begin{aligned} L_o(\omega) &= \int_\Omega L_a + L_d(\omega) + L_s(\omega, \omega')d\omega' \\ &= \int_\Omega \kappa_a L_i(\omega') + \kappa_d L_i(\omega' \cdot \mathsf{n}) + \kappa_s L_i(\omega') \cos^\alpha(\psi) d\omega' \end{aligned} \tag{3.4}$$

3.1.2 Ray Tracing Algorithm

Ray Casting

The first ray tracing algorithm used for rendering was presented by Arthur Appel in 1968 [5], which is usually called ray casting. The idea behind ray casting is to shoot rays from the eye, one per pixel, and find the closest object surface blocking the path of that ray. Given scene geometry, material and lights in the scene, this algorithm can determine the shading of objects.

Ray Tracing

In the ray casting algorithm, a viewing ray is cast from eye to the scene, but is not traced further. Whitted [141] continued the process. When a ray hits a surface, up to three new rays are generated: reflection, refraction, and shadow. A reflected ray continues on in the mirror-reflection direction from a shiny surface. It is then intersected with objects in the scene; the closest object it intersects is what will be seen in the reflection. Refraction rays traveling through transparent material work similarly, with the addition that a refractive ray could be entering or exiting a material. To further avoid tracing all rays in a scene, a shadow ray is used to test if a surface is visible to a light. A ray hits a surface at some point. If the surface at this point faces a light, a ray (to the computer, a line segment) is traced between this intersection point and the light. If any opaque object is found in between the

surface and the light, the surface is in shadow and so the light does not contribute to its shade. This further ray calculation adds more realism to the rendered images [139].

Volumetric Ray Tracing

Volumetric ray tracing is a variant of ray tracing algorithm for rendering 3-d volumetric data, capable of rendering not only opaque solid objects but also transparent objects and half-opaque objects, such as fog, smoke, fire, water, etc. The volumetric ray tracing has also been used a lot in medical image to visualize the 3-d body tissues captured with computer tomography (CT) technology.

In its basic form, the volumetric ray tracing algorithm consists of the following four steps [79, 80]:

Ray casting For each pixel on the image plane, a ray of sight is shot ("cast") through the volume, and its intersection with the volume is computed.

Sampling The ray is sampled along the path within the volume. Because in general the volume is not aligned with the ray, sampling points usually will be located in between voxels. The value of each sample is computed by interpolating its surrounding voxels.

Shading The gradient of each sampling point, i.e., the surface orientation at the sampling point, is computed. The samples are then shaded, i.e. colored and lighted, according to their surface orientation and the source of light in the scene.

Compositing The final color of the pixel, where the ray passes through, is composited from the sampling point's shaded values with the rendering equation.

Volumetric Ray Tracing Function Under Lambertian Assumption

In this work, the most simple shading model: the Lambert model, is assumed. In the Lambert model, since the reflection property of the material is assumed to be purely diffusive, the light reflected from the surface is equal in all directions. Under this Lambertian assumption, a color value can be assigned to each surface point, to denote the light received at the sensor. This color value is the product of surface albedo and the environmental light. The scene is represented with a volume X, which is divided into a set of discrete voxels x_i, where i is the index of that voxel. i.e.,

$$X = \{x_i, \quad i = 1, 2, \cdots, n\}.$$

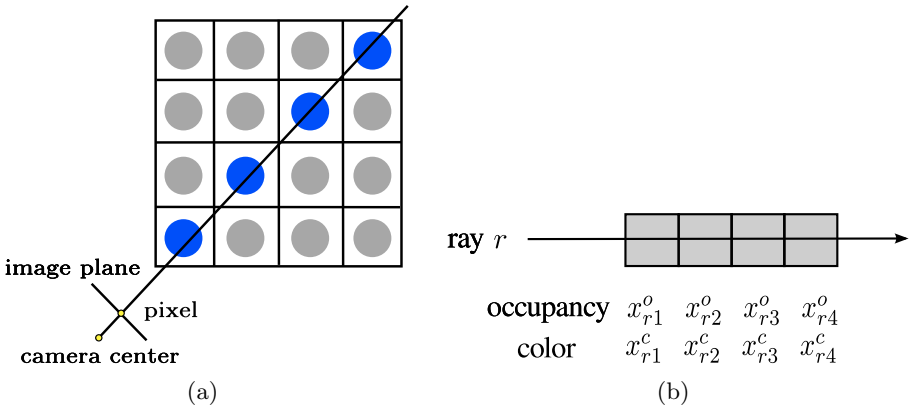


Figure 3.5: Illustration of the relationship between pixel, ray, and voxels: (a) a pixel ray passes through 4 voxels of the 2-d volume; (b) the voxels are ordered with the ray traversal order.

For each voxel, two properties are assigned: binary occupancy x_i^o and real-valued 3-component RGB color x_i^c, i.e.,

$$x_i = (x_i^o, x_i^c).$$

For the occupancy variable x_i^o, it is assigned value 0 when the voxel is empty, and assigned value 1 when the voxel is solid.

Under the Lambertian assumption, the volumetric ray tracing function can be simplified, because the reflected and refractured rays does not need to be further traced. For a given 3-d volume X, the rendered pixel value of the ray r is denoted as $\bar{I}_r$. The volumetric rendering equation is denoted as $\Phi_r(X)$, then $\bar{I}_r = \Phi_r(X)$. Denote the set of voxels that the ray r passes through as X_r. The voxels in X_r are ordered in the viewing ray traversal order. x_{ri} denotes the ith voxel on the ray r; x_{ri}^o denotes its binary occupancy property; x_{ri}^c denotes its RGB color property. Since the rendered pixel value only depends on the voxels that the ray passes through, the volumetric rendering equation can be further written as $\Phi_r(X_r)$. How to compute $\Phi_r(X_r)$? Let's look at a simple example: a ray passes through 4 voxels as shown in Fig. 3.5. In this example, if the first voxel is solid (i.e., $x_{r1}^o = 1$), then the rendered pixel value is the first voxel's color, i.e., $\Phi_r(X_r) = x_{r1}^c$, no matter what the other voxels' occupancy are. If the first voxel is empty (i.e., $x_{r1}^o = 0$), but the second voxel is solid ($x_{r2}^o = 1$), then the rendered pixel value is the second voxel's color, i.e., $\Phi_r(X_r) = x_{r2}^c$, no matter what the 3rd and 4th voxels' occupancy are. And so on. In summary, the rendered

pixel value of a ray is the first solid voxel's color. This can be summarized formally as

$$\Phi_r(X_r) = \begin{cases} x^c_{r1}, & X^o_r = 1, \times, \times, \times, \times, \cdots \\ x^c_{r2}, & X^o_r = 0, 1, \times, \times, \times, \cdots \\ \cdots & \\ x^c_{ri}, & X^o_r = \underbrace{0, \cdots, 0}_{i-1}, 1, \times, \cdots \\ \cdots & \end{cases} \tag{3.5}$$

where $\times$ denotes either value of 0 or 1, X^o_r is a vector of occupancy values for the voxels on the ray r sorted in the traversal order of the viewing ray.

3.2 Statistical Inverse Ray Tracing

3.2.1 Analysis by Synthesis Framework

In some sense, computer vision, which tries to understand the scenes from images by extracting a description of the scene (low-level intrinsic representation or high-level semantic description), is an inverse problem of computer graphics, which generates images from a description of the scene. Here we are only interested in recovering the low-level intrinsic representation of the scene – geometry and appearance, instead of its high-level semantic description, from images. This image-based 3-d recovery problem corresponds to the rendering problem in computer graphics. As discussed above, inverse ray tracing as an inverse problem, is ill-posed. Its solution needs to combine information from both the observation and the prior knowledge. This fits to the statistical framework of pattern theory [55], where one of the essential ideas is "analysis by synthesis". Inverse ray-tracing is the "analysis" of the scene through studying the "synthesis" of observations. The image synthesis process is modeled by ray-tracing.

The basic idea of statistical analysis by synthesis is to model the posterior probability $P(X|Y)$ (where X is the unknown to be estimated, Y is the observation) through Bayes' theorem, i.e., through modeling the prior distribution $P(X)$ and the data generation process $P(Y|X)$:

$$P(X|Y) \propto P(X)P(Y|X). \tag{3.6}$$

For example, the most probable estimation of X can be solved as

$$X^* = \arg\max_X P(X|Y) \Leftrightarrow X^* = \arg\max_X P(X)P(Y|X). \tag{3.7}$$

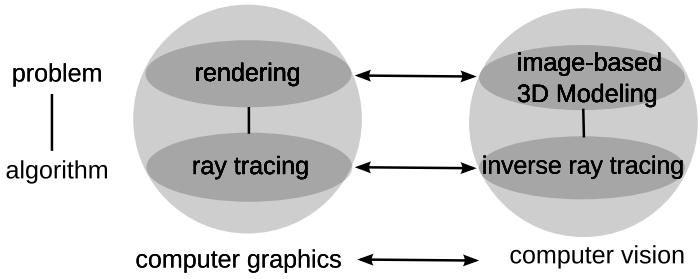


Figure 3.6: Connections amongst rendering, multi-view stereo, ray tracing and inverse ray tracing

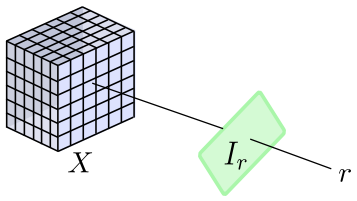


Figure 3.7: Illustration of ray constraint in inverse ray tracing

It is equivalent to the following energy formulation (the $-log$ form):

$$X^* = \arg\min_X E(X|Y) \Leftrightarrow X^* = \arg\min_X \{E(X) + E(Y|X)\}, \tag{3.8}$$

where $E(X)$ is the prior energy of X, $E(Y|X)$ is the conditional energy of observation Y given X.

3.2.2 Photo-Consistency Optimization

Given a 3-d model X and a ray r, the observation of the ray $\bar{I}_r$ can be rendered with ray tracing, i.e.,

$$\bar{I}_r = \Phi_r(X), \tag{3.9}$$

where $\Phi_r(\cdot)$ is the volumetric ray tracing function. The goal is to find the best 3-d model(s) X^*[1], given the observed values $\{I_r\}$, which minimize(s) the difference between the rendered value $\bar{I}_r$ and observed value I_r:

$$X^* = \arg\min_X \sum_{r \in R} ||I_r - \bar{I}_r||^2 = \arg\min_X \sum_{r \in R} ||I_r - \Phi_r(X)||^2. \tag{3.10}$$

[1]The model X^* may not be unique, because of the ambiguities in the solution. It is one of the reasons that probabilistic modeling is necessary and why the prior model of the surface is helpful to reduce the ambiguities.

Because $\bar{I}_r$ only depends on X_r — the set of voxels through which the ray r passes through — then the volumetric rendering equation can be denoted as $\Phi_r(X_r)$. Correspondingly, the optimization problem is simplified as

$$X^* = \arg\min_X \sum_{r \in R} ||I_r - \Phi_r(X_r)||^2. \tag{3.11}$$

Before diving into the details of solving the optimization problem, some properties of the formulation is summarized.

1. The number of random variables is large (e.g., 100^3–1000^3 or larger, depending on the volume resolution).

2. The interaction between random variables is complex: the rendering function is a semi-global function of the entire set of voxels.

3. The solution to the problem may not be unique.

4. It is equivalent to the Bayesian formulation with non-informative prior (i.e., $P(X)$ is a uniform distribution):

$$X^* = \arg\max_X P(X|\{I_r\}) = \arg\max_X P(X)P(\{I_r\}|X) = \arg\max_X P(\{I_r\}|X). \tag{3.12}$$

The above properties (1-3) make the optimization problem hard, especially the complex interaction between voxels along a ray. The mathematical tools on Markov random field (MRF) provides a way to model large scale statistical systems, together with the prior regularity model. MRF has been successfully applied to solving many high-dimensional inverse problems in image processing and low-level vision. In the next chapter, the image-based 3-d modeling problem will be formulated in MRF, and the existing work on solving similar problems will be leveraged to solve the problem.

3.2.3 Related Work

Before talking about the related work, let's clarify one commonly asked question: why the proposed approach is named as "inverse ray tracing"? Isn't all the work on 3-d reconstruction doing the work of "inverse ray tracing"? Broadly speaking, all the 3-d reconstruction can be seen as "inverse ray tracing" or "inverse rendering". But in this thesis, the term is reserved for the process of literally inverting the ray tracing based image generation process, i.e., the process of solving the image-based 3-d reconstruction problem by solving an inverse problem of "ray tracing".

The idea of inverse ray tracing emerges as a result of accumulating research. Key concepts in the idea of inverse ray tracing include 1) the representation of the whole scene as unknown variables, 2) explicit formulation of the ray constraints, 3) the combination of both prior knowledge and the observation with the Bayesian principle, and 4) the formulation of the problem as a global optimization problem. Some of the ingredients have already been studied in the existing literatures. The iterative surface evolution approach optimizes the re-projection error and takes the scene as unknown variables. The re-projection formulation limits their optimization to be local (e.g., gradient descent and its variants), because it needs to work with a 3-d surface, from which a surface point can be projected to images and get updated. As a result, this approach needs a good initial surface to start with. The space carving approach is a general approach that can handle large varieties of scenes, without assuming a good initial surface. It was first proposed as a projection approach (i.e., projecting a voxel to an image). Then in [19, 18], Broadhurst, Drummond and Cipolla modeled the probabilistic visibility of a voxel along a "ray", which goes from the camera center to that voxel. The visibility is then computed as a product of proceeding voxels' non-existing probabilities. The combination of different rays' information is computed with a Bayesian rule, which suffers from the combinatorial explosion. A local greedy solution is adopted by introducing a local threshold. In the pioneering work of [106, 105, 30], Pollard, Crispell and Mundy developed a (online) Bayesian updating rule to estimate voxel surface probabilities. In these work, the ray constraint is formally defined as the relationship between the pixel observation and the voxels that the back-projected ray passes through for the first time. Similarly to overcome the high computational cost of exact Bayesian updating, the updating is performed independently for each ray. This work is inspired by the pioneering ideas in [18, 105, 30]. The problem is first formulated as a global optimization problem, optimally combining the prior knowledge and high-fidelity image generation process based on volumetric ray tracing. Then as shown in the later chapters, an efficient approximate global solution is also developed to solve the optimization problem in high quality, as verified by extensive experiments later.

Besides the work in multi-view stereo, there is another thread of closely related work: "inverse rendering" or "relighting", which estimates scene's lighting and material properties, given the scene's geometry. In [91], Marschner studied the problem of estimating attributes of the scene from photographs. Specifically, they have studied the problem of 1) estimating the lighting, given the scene's geometry, reflectance and photographs; 2) estimating reflectance, given scene's geometry, lighting and photographs; and 3) estimating the surface's BRDF, given geometry, lighting and photographs. In summary, [91] assumes that

Figure 3.8: Relighting objects from image collections [56]. From left to right: samples of the input images downloaded from flickr [1], sparse point cloud reconstruction and camera visualization, reconstructed surface model, re-lighted object.

the geometry is given, and then estimates the lighting and surface material interchangeably. Different from [91], this work estimates surface's geometry and appearance at the same time. In [146], Yu et al. presented a novel relighting approach that does not assume that illumination is known or controllable. In their approach, the illumination and texture are estimated together from given multi-view images captured under a single illumination setting, given object shape. There has been some recent work on estimating scene's lighting and material from community photo collections with the geometry recovered through multi-view stereo algorithms [56]. It can be seen that "inverse ray tracing" and "inverse rendering" are studies of the same subject with different focuses. It is a natural idea to combine these techniques to estimate geometry, lighting and material together in one formulation. This approach could improve the reconstruction accuracy of both geometry and material. Multi-view stereo is sensitive to the lighting change and non-Lambertian surface material, and the relighting has been shown to be sensitive to the inaccurate geometry reconstruction [56]. Estimating them together seems be a promising direction of solving these problems, as illustrated in Fig. 3.10.

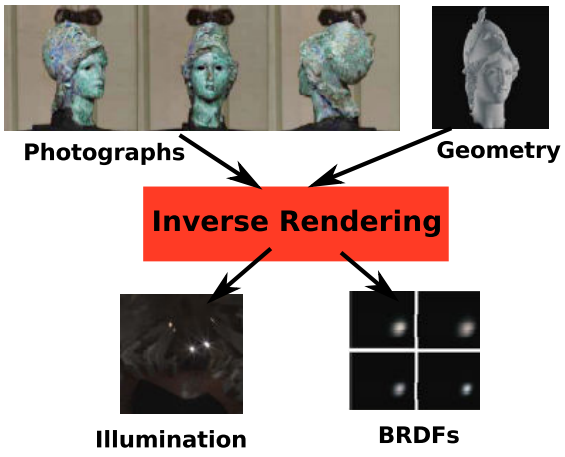


Figure 3.9: Diagram of inverse rendering [56]

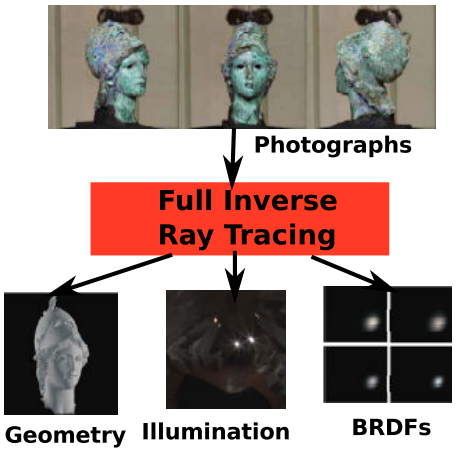


Figure 3.10: Diagram of full inverse ray tracing

Chapter 4

Ray Markov Random Field

At the intersection of statistical physics and probability theory, Markov Random Fields (MRFs) emerged in the early eighties as powerful tools for modeling images and coping with high-dimensional inverse problems in low-level vision. Since then, lots of computer vision problems have been formulated as MRF inference problems. Moreover, the quality and speed of MRF inference algorithms have been dramatically improved over the last decades. The last chapter discussed the "Analysis by Synthesis" framework for handling image-based 3-d modeling as an inverse problem. Here the problem is formally modeled with the Ray Markov Random Field (RayMRF). The RayMRF's graphical structure and unique characteristics are analyzed. The end of this chapter discusses related works on using MRF in multi-view stereo and their connections with this work.

4.1 Review of Markov Random Fields

Markov random fields, which have been used as powerful tools for statistical physics [8] and statistics [6, 13] since the 1960s, were first applied to image processing in the 1980s [49, 26]. Since then, their theoretical richness and practical versatility have been widely explored

notation	meaning
Ω	the collection of indices
x_a, $a \in \Omega$	a random variable with index a
X_A, $A \subset \Omega$	the set $\{x_a, a \in A\}$, A is a subset of Ω
X	$X = X_\Omega = \{x_a, a \in \Omega\}$, i.e., the entire set of random variables
$\mathcal{N}_a$	the index set whose elements are neighbors of index a
$\mathcal{N}$	the neighborhood set, i.e., $\mathcal{N} = \{\langle i, j\rangle\}$, where $\langle i, j\rangle$ is a neighbor pair

Table 4.1: MRF notations

[25, 28, 22, 82, 24, 149, 112]. This section reviews the essential concepts and terminologies of MRF for later development. This review is not intended to be comprehensive, please refer to books (e.g., [82], [14], [22]) for more complete and rigorous accounts.

The implicit assumption behind probabilistic approaches to image analysis is that, for a given problem, there exists a probability distribution that can capture (to some extent) the variability and the interactions between image attributes. Consequently, these attributes are modeled as a set of random variables $X = \{x_i, i = 1, \cdots, n\}$ with the probability measure P. The following properties and operations are used to construct and analyze the probability P.

Markov Property Here the Markov chain defined on a sequence is generalized to the Markov property on a general graph. For a set of random variables and the probability measure P, the Markov property is defined as

$$P(x_a|X_{\Omega\setminus a}) = P(x_a|X_{\mathcal{N}_a}), \forall a \in \Omega. \tag{4.1}$$

Literally, the conditional probability of a random variable x_a over all the other random variables $X_{\Omega\setminus a}$ is only a function of its neighbor random variables $X_{\mathcal{N}_a}$. This local Markov property indicates that the whole joint probability has strong conditional independences, which are characterized by the neighbor sets of each random variable. But how does one define a joint probability $P(X)$ that is consistent with the local Markov property? This problem is better answered from the factorization perspective.

Factorization Property One simple way of writing down the whole distribution is to decompose it as a product of factors, with each factor depending on just a few variables. This decomposed distribution can then be expressed simply by specifying these local "interaction" factors. Formally speaking, $P(X)$ exhibits a factorized form:

$$P(X) \propto \prod_{C\in\mathcal{C}} F_C(X_C), \tag{4.2}$$

where $\mathcal{C}$ consists of small index subsets C, and the factor F_C depends only on the variable subset $X_C = \{x_i, i \in C\}$. A factor is also called a clique of the factorization graph; and an index set C is also called a clique set. From now on, we use the term "factor" and "clique" interchangeablly.

It can be observed that there is a dual relationship between the neighbor sets and the clique sets. Neighbor sets are centered around each random variable, and clique sets are centered around each factor. However, they are mutually translatable: from a clique set, its corresponding neighbor set can be constructed, and vice versa.

Equivalence between Markov Property and Factorization Property The formal equivalence between the above two properties were established by Hammersley and Clifford [57]: *"A probability distribution P with positive and continuous density satisfies the Markov property with respect to an undirected graph G if and only if it factorizes according to G."*

Energy Formulation If the product of factors is positive, then it can be written in exponential form. Denote

$$E_C := -\log F_C, \tag{4.3}$$

then $P(X)$ can be written as

$$P(X) = \frac{1}{Z} e^{-\sum_{C \in \mathcal{C}} E_C(X_C)}. \tag{4.4}$$

Well known from physics, this distribution is the **Gibbs (or Boltzman) distribution** with the interaction potential $\{E_C, C \in \mathcal{C}\}$. The total energy of the system is defined as

$$E(X) := \sum_{C \in \mathcal{C}} E_C(X_C), \tag{4.5}$$

and the partition function (of parameters) $Z(X)$ is defined as

$$Z := \sum_{X} \exp\{-E(X)\}. \tag{4.6}$$

Configurations of lower energies have higher probabilities, and high energies correspond to low probabilities.

Inference Algorithms After the construction of the MRF model, two natural questions about the model present themselves: (1) what's the marginal probability distribution of a particular random variable? (2) what's the most probable state of the system? The high dimensionality of the problem usually excludes any direct method for performing both tasks. In general, these inference problems are computationally NP-hard [82], and as a result approximate or local methods are usually employed. Fortunately, the local decomposition of the probability measure P allows one to devise smart deterministic or stochastic iterative algorithms. Over the past few years, there have been several exciting advances in the development of algorithms for solving early vision problems, such as stereo and image restoration using Markov random field (MRF) models. These methods are good in the sense that the converged "strong local minimum" is "close" to optimal.

Typical MRF Model in Computer Vision In computer vision, lots of problems can be formulated as labeling problems. Let Ω be the set of pixels in an image and L be a finite set of labels. The labels correspond to the pixel attributes to be estimated, such as intensity or disparities. A labeling X assigns a label $x_i \in L$ to each pixel $i \in \Omega$. It is usually assumed that the neighbor labels have some regularities, for example, neighbor labels have similar values. The quality of a labeling is measured by an energy function,

$$E(X) = \sum_{i \in \Omega} D_i(x_i) + \sum_{\langle i,j \rangle \in \mathcal{N}} W(x_i, x_j), \tag{4.7}$$

where $\mathcal{N}$ is the nearest neighborhood clique set on the image grid; $D_i(x_i)$ is the data clique energy, defined as the cost of assigning label x_i to pixel i; $W(x_i, x_j)$ is the pairwise clique energy, measuring the cost of assigning labels x_i and x_j to two neighboring pixels i and j.

In low-level vision problems, the discontinuity cost W is usually defined based on the difference between labels, rather than their absolute values. Thus $W(x_i, x_j) = V(x_i - x_j)$ is commonly used, yielding an energy minimization problem of the form,

$$E(X) = \sum_{i \in \Omega} D_i(x_i) + \sum_{\langle i,j \rangle \in \mathcal{N}} V(x_i - x_j). \tag{4.8}$$

This model is called the standard MRF model in computer vision.

Factor Graph Representation In MRF, each variable only *directly* depends on a few other "neighboring" variables. From a more global point of view, all variables are mutually dependent, but only through the combination of successive local interactions. This key notion can be represented by a graph where i and j are neighbors if x_i and x_j appear within the same local component of the chosen factorization. Each random variable is denoted by a circle and each factor denoted by a square. A circle and a square are connected if and only if the circle's corresponding random variable participates in the square's factor. There is no edge link between random variable nodes; nor is there an edge link between factor nodes. One example factor graph is shown in Fig. 4.1. A factor graph is a generalization of the MRF dependency graph, making the connection between random variables and factor functions explicit. In other words, this graph neatly captures Markov-type conditional independence among the random variables. It can be used to understand and develop efficient inference algorithms by exploring an MRF's graphical structure.

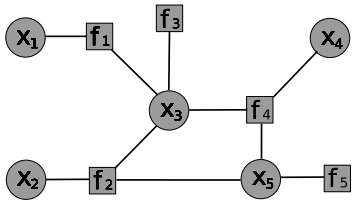


Figure 4.1: Factor graph representation of probability distribution $P(x_1, x_2, x_3, x_4, x_5) \propto f_1(x_1, x_3) f_2(x_2, x_3, x_5) f_3(x_3) f_4(x_3, x_4) f_5(x_5)$, or equivalently energy $E(x_1, x_2, x_3, x_4, x_5) = E_1(x_1, x_3) + E_2(x_2, x_3, x_5) + E_3(x_3) + E_4(x_3, x_4) + E_5(x_5)$.

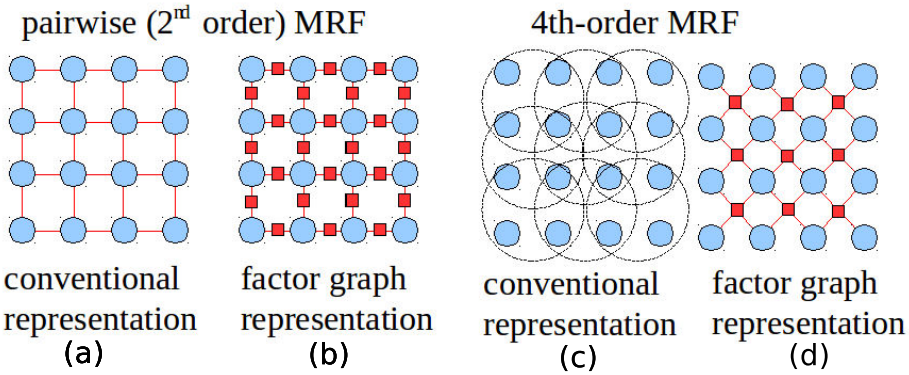


Figure 4.2: Conventional representation and factor graph representation of the pair-wise MRF and the 4th-order MRF

Higher Order Cliques A higher order clique consists of more than 2 random variables. How to represent the higher-order clique graphically? Traditionally, the 4th-oder MRF is represented by drawing a circle around the random variable nodes to denote a clique. As can be seen in Fig. 4.2(c), such a representation is not explicit in revealing the graphical structure, nor is a network representation. The factor graph representation shown in Fig. 4.2(d) solves these problems by creating a new node, the clique node (represented by a square graphically), and connecting it to all the random variable nodes. The higher-order MRF is then represented by a bipartite graph with two kinds of nodes: random variable nodes and clique nodes. The pairwise clique can also be represented in this way, as shown in Fig. 4.2 (a, b).

Higher-order cliques pose hard problems for learning and inference algorithms. The first problem is how to specify the large number of parameters in the higher-order clique. This learning problem has attracted researchers' attention and several solutions have been proposed, such as [149, 112]. There has also been some works reported on solving the inference problem for higher-order cliques [76, 68, 110]. Most of these

works assume some properties of the higher-order clique energy. Solving the inference problem for higher-order cliques is therefore still a very challenging research problem.

4.2 Ray Markov Random Field

4.2.1 Ray Clique

Section 3.1.1 studied the rendering equation and shading models for computing the rendered value $\bar{I}_r$ of a viewing ray r. And as discussed in section 3.2.2, in inverse ray tracing, given the observed pixel value I_r, the goal is to find the best 3-d model X that optimizes $\sum_{r\in R} ||I_r - \Phi_r(X_r)||^2$. Here define the term

$$E_r(X_r) = ||I_r - \Phi_r(X_r)||^2 \tag{4.9}$$

as the ray clique energy, which measures the 3-d model's consistency with the pixel on a specific ray r. As the first step towards inverse ray tracing, the simplest shading model, Lambert model, is assumed. As discussed in section 2.1, the 3-d scene is modeled with a volume X. By volumetric ray tracing with a Lambert model discussed in section 3.1.2,

$$\Phi_r(X_r) = \begin{cases} x^c_{r1}, & X^o_r = 1, \times, \times, \times, \times, \cdots \\ x^c_{r2}, & X^o_r = 0, 1, \times, \times, \times, \cdots \\ \cdots & \\ x^c_{ri}, & X^o_r = \underbrace{0, \cdots, 0}_{i-1}, 1, \times, \cdots \\ \cdots & \end{cases} \tag{4.10}$$

where $\times$ denotes either value of 0 or 1, X^o_r is a vector of occupancy values for the voxels on the ray r sorted in the traversal order of the viewing ray. The above equation says that the color of the viewing ray is equal to the color of the first solid voxel that is visible along the ray.

4.2.2 Ray Markov Random Field Model

Besides the ray clique energy, which models the observation process, there are two other clique energies: unary clique energy and pair-wise clique energy, which model the prior distribution of the discrete volume. Unary clique energy models the prior knowledge of a voxel being solid or empty. Pair-wise clique energy models the continuity of the volume, which is equivalent to the minimum surface prior [16]. Formally, the whole MRF can be

expressed in the following energy form:

$$E(X) = \sum_{r \in R} E_r(X_r) + w_p^o \sum_{\langle i,j \rangle \in \mathcal{N}} E_p^o(x_i^o, x_j^o) + w_p^c \sum_{\langle i,j \rangle \in \mathcal{N}} E_p^c(x_i^c, x_j^c) + w_{uv}^o \sum_{k \in \Omega} E_{uv}^o(x_k^o) \quad (4.11)$$

where $E_r(X_r)$ is the clique energy for the ray r, and X_r is the set of voxels that ray r passes through; $E_p^o(x_i^o, x_j^o)$ is the pair-wise occupancy clique energy, and w_p^o is its weight; $E_p^c(x_i^c, x_j^c)$ is the pair-wise color clique energy, with w_p^c being its corresponding weight; and $E_{uv}(x_k^o)$ is the unary voxel ("uv") occupancy clique energy, and w_{uv}^o is its weight. The functional form of each energy function is defined as follows:

$$E_r(X_r) = ||I_r - \Phi_r(X_r)||^2 \quad (4.12)$$

$$E_p^o(x_i^o, x_j^o) = \begin{cases} 0, & x_i^o = x_j^o \\ 1, & x_i^o \neq x_j^o \end{cases} \quad (4.13)$$

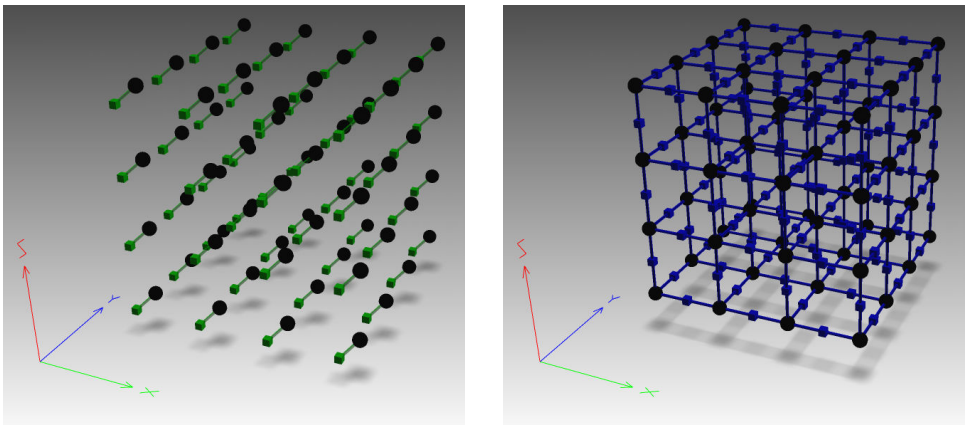
$$E_p^c(x_i^c, x_j^c) = ||x_i^c - x_j^c||^2 \quad (4.14)$$

$$E_{uv}^o(x_k^o) = (x_k^o - 1)^2 \quad (4.15)$$

The graphical structure of the MRF is illustrated in Fig. 4.5, where, following the convention of factor graph [71], each voxel is denoted by a ball, and each ray clique is denoted by a tube connecting the random variables (i.e., voxels) on the ray.

4.2.3 Comparison between Ray MRF and Standard MRF

There are several significant differences between the Ray MRF model (4.11) and the standard MRF model (4.8). The difference with respect to the data term is most obvious. In the standard model, the data term has just one random variable x_i. However, in Ray MRF, the data term involves a large number of random variables X_r, literally, the set of voxels that the ray r passes through. The number of variables involved in the ray clique ranges from hundreds to thousands and more, depending on the resolution of the volume. This difference suggests a potential problem with the traditional approaches to solving the inference of Ray MRF. In the typical MRFs, the data terms are usually a quadratic function for modeling the noisy corruption of the pixel intensity/disparity. MRF is in such approaches used to de-noise the image or to fill in missing values. However, in Ray MRF, the data term (ray clique energy) is not a simple quadratic function, but a complex step-wise function involving the occlusion and interpolation (which will be discussed later) of two kinds of random variables: binary occupancy variables and real-valued color variables. Here the ray clique is not for smoothing: but for *inferring the unknown values through an indirect observation*



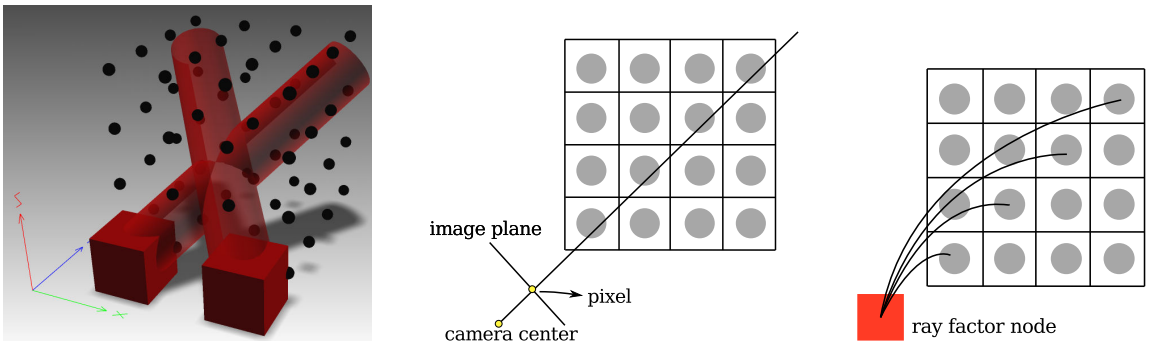
(a) unary cliques (b) pair-wise cliques

Figure 4.3: Visualization of the unary and pair-wise cliques in Ray MRF

process, as opposed to standard MRF's direct observation of each pixel intensity/disparity value. This casts yet another doubt on the Ray MRF inference, as will be shown later, the biggest obstacle is that *the size of the ray clique is large and semi-global* (100–1000) compared to the pair-wise cliques in the standard MRF model. These differences (i.e., the large clique size and the indirect measurement) are the unique characteristics of the Ray MRF. These unique properties of Ray MRF pose challenges to the traditional inference algorithms, which become computationally infeasible when applied to the inference of Ray MRF. These problems will be addressed in the next chapters.

4.2.4 Factor Graph Representation of Ray MRF

The factor graph representation of Ray MRF is shown in Figs. 4.3, 4.4 and 4.5. Fig. 4.3 (a, b) visualize the unary cliques and pair-wise cliques respectively, where the clique nodes are represented by cubes. For the ray clique, a red cube is used to denote the clique node, and a tube along the ray is used to denote the connection between the ray clique node and the voxels on the ray, as shown in Fig. 4.4(a). However, because the ray clique is hard to visualize in 3-d, it is better shown in 2-d (Fig. 4.4(b, c)). Then these three clique factor graphs are combined together to get the whole factor graph of the Ray MRF model, shown in Fig. 4.5.



(a) 3-d ray clique visualization (b) 2-d ray-voxel intersection (c) 2-d ray clique factor graph

Figure 4.4: Ray clique visualization

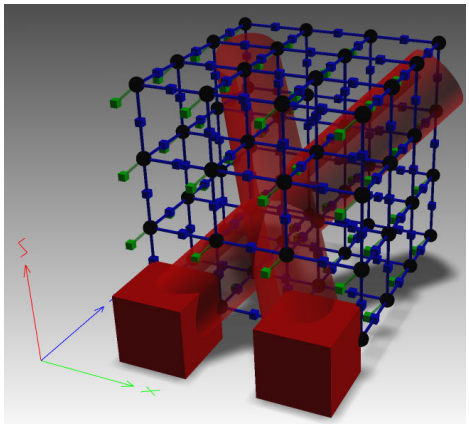


Figure 4.5: Ray MRF visualization (with the factor graph representation [71]): black balls represent voxels, cubes represent clique factors and tubes represent clique connections: green for unary clique, blue for pair-wise clique, and red for ray clique. (best viewed in color)

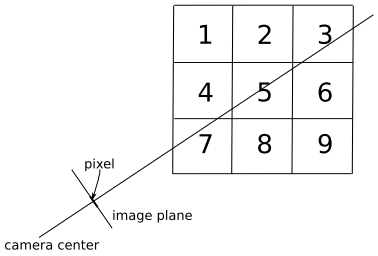


Figure 4.6: Ray line model

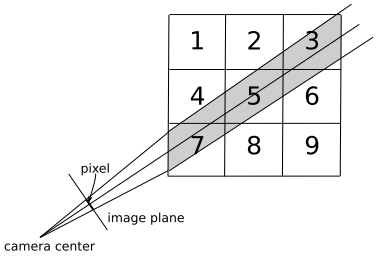


Figure 4.7: Ray tube model

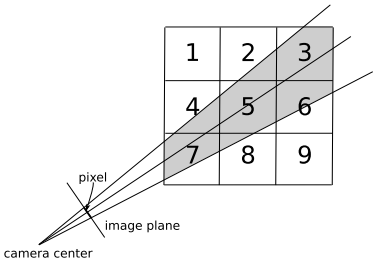


Figure 4.8: Ray cone model

4.2.5 Ray-Voxel Intersection Relationship

One of the key steps of building the Ray MRF is to compute the ray-voxel intersection relationship. There are several choices. The ray can be modeled as a line, cube or cone. The ray-voxel intersection relationship can be treated as binary (intersected or not intersected) or fractional (partially intersected). These choices affect the computation cost and accuracy of the reconstruction.

Ray-Line Model In this simple model, the ray is treated as an infinitesimal thin line. And the ray-voxel intersection is simply a line passing through a set of voxels. In this model, the size of the pixel is not considered; and the camera is modeled as a point sensor, instead of an area sensor. As shown in Fig. 4.6, the ray in the figure passes through voxels $\{7, 4, 5, 6, 3\}$ in order. One obvious problem is that the ray just marginally passes through voxel no. 4 and 6. Should they be included or not? One solution is to use the length of the segment on the ray that's inside each voxel as a weight. This approach has been adopted by Crispell and Mundy [30].

In practice, a pixel is not of infinitesimal size – each pixel has a finite area. The next two models account for this.

Ray-Tube Model If the object size is much smaller than the distance from the camera to the object, the ray cone can be approximated as a tube. This can be used for small-middle size object reconstruction. As shown in Fig. 4.7, treating the ray as a tube, can better capture the ray-voxel intersection relationship, for example, the contributions from voxels 8 and 2 are counted in this model.

Ray-Cone Model When reconstructing large-scale scenes, or when the camera is very close to the object, the above ray-tube model is no longer a good approximation. In general, the ray for a pixel is a cone, as shown in Fig. 4.8.

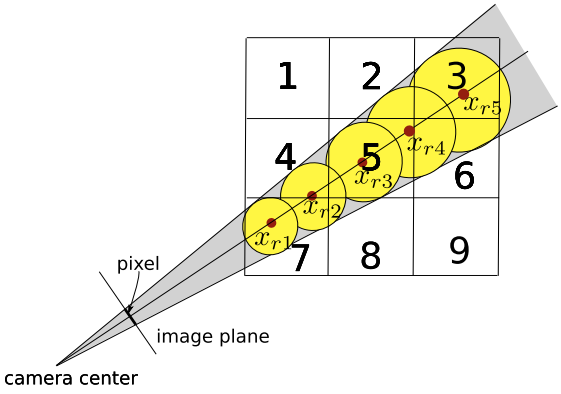


Figure 4.9: Even sampling in ray-cone model

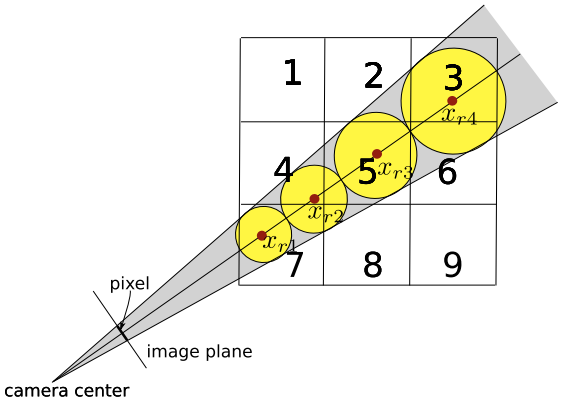


Figure 4.10: Proportional sampling in ray-cone model

The problem with the ray-tube model and ray-cone model is that the order of voxels each ray passes through, which is required by the volumetric ray tracing, is not well-defined. For example, in Fig. 4.7 and 4.8, the ray passes through voxels $\{7, 4, 8, 5, 2, 6, 3\}$. But what's the order between voxel no.4 and voxel no.8? To both maintain the intersection order and account for the fractional intersection, the idea of "sampling" and "interpolation" is introduced. The ray is sampled along the line, and the value of the sample is computed through interpolation of its neighbors. Fig 4.9 shows a scheme of sampling, where the ray is evenly sampled from the first intersection point to the last intersection point. Each sample has an area proportional to the diameter of the cone. The sampling area inside each voxel serves as the weight of that voxel on the ray. The problem with even hsampling is that it does not account for the near-far resolution problem. As the ray moves further, the sampling does not need to be as dense as the near points. So a better scheme is to distribute the samples unevenly as shown in Fig. 4.10. In this sampling scheme, the distance between consecutive sampling points is proportional to the cone diameter. Here the sampling scheme not only solve the voxel ordering problem but also improve the accuracy of intersection — sub-voxel accuracy. For example, in the example shown in Fig. 4.10, the value of x_{r2} can be interpolated as

$$x_{r2} = w_{r2}^4 x_4 + w_{r2}^5 x_5 + w_{r2}^7 x_7 + w_{r2}^8 x_8, \tag{4.16}$$

where w_{r2}^4, w_{r2}^5, w_{r2}^7, and w_{r2}^8 are the weights, defined as the portion of intersected area of the voxel cube in the sampling ball.

4.3 Related Work

There are several other works on using Markov random fields for image-based 3-d reconstruction, e.g.[125, 137, 136, 122, 133]. All these works use the standard MRF models as defined in Eq. (4.8) to enforce the continuity and smoothness of the volume occupancy labels. The observation value of each voxel is defined based on some measure of the voxel's photo-consistency with all the views. The photo-consistency is defined approximately either by ignoring the visibility information [125] or based on the visibility computed from some initial shape [137, 136, 122, 133]. However, in this work, the semi-global ray clique is introduced to better model the image generation process and to remove the dependency on some initial shape. It is the ray clique that differentiates the proposed Ray MRF from the standard MRF models, and makes the corresponding inference problem challenging to solve.

The introduced semi-global ray clique model is inspired by the the seminal work on MRF modeling of computed tomography by Geman, McClure and Manbeck [52, 51], where the single photon emission tomography (SPET) is modeled with MRF. In that model, the detected photo intensity is a summation of all the photons emitted along all possible direction through an attenuation process from the body tissue. It can be seen that the observation process is also semi-global. However, under the assumption of dense uniform sampling, the data observation can be modeled with a linear relationship with the unknown concentration of the radiopharmaceutical, which makes the corresponding inference problem easier. In this problem, because the observation process is directional and the location of the sensors (i.e., cameras) is un-controllable and usually far from densely uniformly distributed, the data observation can not be approximated with a linear process, which makes this problem different.

Chapter 5

Voxel Occupancy and Color Estimation

The art of doing mathematics consists in finding that special case which contains all the germs of generality.
— David Hibert (1862-1943)

In the previous chapters, image-based 3-d modeling is formulated as a Ray MRF inference problem. The problem is very challenging because (1) there is no direct observation for each hidden random variable (i.e., voxel, here) as in usual MRFs, serving as a good initial estimate; (2) there is a semi-global clique in the Ray MRF, and higher-order cliques are hard to deal with in general. There are many optimization methods for MAP estimation of MRF, including local algorithms such as ICM, gradient descent and their variants, sampling methods such as Markov Chain Monte Carlo (MCMC), and approximate algorithms such as graph cuts and loopy belief propagation (LBP), etc. Among them, graph cuts and LBP are the most popular MRF inference algorithms in computer vision for their good convergence quality and speed. However, the semi-global nature of the ray clique in the proposed model prohibits the direct application of these inference algorithm. In this chapter, the problem is studied in detail and an efficient algorithm is developed to estimate voxel occupancies (i.e., geometry) and colors (i.e., appearance), by exploring the special structure of the ray clique. At the end of this chapter, related works are discussed.

As discussed in Chapter 4, each voxel has two properties — occupancy and color. Mathematically, the occupancy is denoted by a binary discrete variable, and the color is represented by a three-component continuous variable (or quantized into 256 levels in each component). These two types of random variables are so different that it is natural to estimate

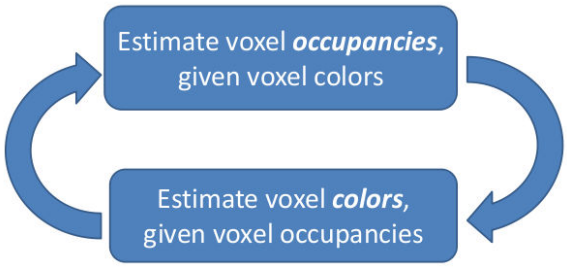


Figure 5.1: Alternating estimation of voxel occupancies and colors

notation	meaning
x	a random variable
f	a factor/clique
x_i	a random variable with index i
f_i	a factor/clique with index i
X_f	the set of random variables adjoining to factor f
x_{fi}	the ith random variable in X_f
$\mathcal{C}(x)$	the set of cliques adjoining to random variable x
$M_{x\to f}(x)$	the message from random variable x to factor f
$M_{f\to x}(x)$	the message from factor f to random variable x

Table 5.1: Notations in belief propagation

them alternately (in an iterative way), as shown in Fig. 5.1. The idea is similar to the coordinate-alternating optimization, where each variable is estimated separately given the other variables are fixed, and the procedure is iterated by alternating the variables. In the following, voxel occupancies are first estimated given voxel colors, and then voxel colors are estimated by fixing voxel occupancies. By doing these alternately in a recursive fashion, the whole energy function (4.11) defined in the Ray MRF can be optimized. In this chapter, the basic concepts in loopy belief propagation are review, and then LBP is directly applied to the voxel occupancy estimation, where computational challenges are analyzed and overcome by studing the sparse problem structure. The voxel color estimation is studied at the end of this chapter.

5.1 Review of Loopy Belief Propagation

Loopy belief propagation is an algorithm to compute the marginal or maximal distribution of a large system by solving a set of smaller integration or optimization problems. Assume $X = \{x_i, i = 1, \cdots, N\}$ is a set of discrete random variables of a system with a joint mass function P. Given this probability model, there are several natural questions to ask about

the system:

Marginal Distribution and Mean Estimation: What's the **marginal probability distribution** of a random variable x_i? By definition, it can be computed as [1]

$$P(x_i = a) = \sum_{X:x_i=a} P(X) \tag{5.1}$$

What is the **mean estimate** of a random variable x_i? It can be computed as

$$\overline{x_i} = \sum_{x_i} x_i P(x_i) \tag{5.2}$$

Conditional Distribution and MAP Estimation: What is the most probable state X^* of the system, given observation Y? By definition, it can be computed as

$$X^* = \arg\max_X P(X|Y). \tag{5.3}$$

X^* is also called the **MAP (maximum a posteriori) estimate** of X.

$$P^*(x_i = a) = \max_{X:x_i=a} P(X) \tag{5.4}$$

is called the conditional distribution of x_i, given the other variables taking their MAP estimates, or **MAP conditional distribution** of x_i, for short. It can be easily verified that the ith element value of X^* is the same as $\arg\max_{x_i} P^*(x_i)$.

When the number of variables is small, the summation and maximization operations can be carried out easily even in a brute-force way. Both of these two fundamental problems in statistical estimation quickly become computationally intractable thanks to "the curse of dimensionality": if there are 100 binary variables, then one needs to summarize or maximize over 2^{100} possible values — an astronomical figure. To break the "curse", the structures of the probabilistic model are usually exploited. Efficient algorithms, like dynamic programming, can take advantage of these structures to avoid the brute-force computation. Belief propagation is one of these algorithms.

Loopy Belief Propagation (LBP) is a message passing algorithm for performing inference on probabilistic graphical models. It can be used to (approximately) compute the marginal distribution or maximal distribution for each random variable or group of random variables. Empirical successes have been reported in numerous applications including turbo codes [90], low level vision (e.g., stereo [131], optical flow [38], tracking [129]), etc. Belief

[1] Here after, x_i is assumed to be a discrete variable, for simplicity.

propagation was first proposed as a message-passing implementation of marginal integral on tree structured probabilistic models by Judea Pearl in 1982 [103]; it was later extended to polytrees [66] and has further been shown to work well on general graphs [104, 95]. However, there has been little understanding of the nature of the solutions to which BP converges for a general graph until the work by Yedidia, Freeman and Weiss [143]. They showed that BP converges to a stationary point of an approximate free energy, known as Bethe free energy in statistical physics, if it converges. Based on this connection, there have been other variants of message-passing algorithms proposed based on other (better) free energy approximations, e.g., Kikuchi free energy [143, 144].

However, belief propagation is only efficient for lower-order MRFs. It quickly becomes computationally infeasible as the size of the clique increases. Here in this thesis, It is shown that by factorizing the high-order cliques further (not in the traditional sense, as discussed later), dynamic programming can once again help accelerate the computation dramatically. For the central role of dynamic programming in belief propagation and the later development, its basic ideas are first reviewed and its connection with message passing and belief propagation is built. These connections will lead naturally to the core ideas in deep belief propagation.

Dynamic programming is a key concept in many high-efficiency algorithms. In this chapter, message-passing algorithms are interpreted as a parallel implementation of dynamic programming, and belief propagation is a message-passing implementation for computing marginal/maximal distributions on tree-structured probabilistic models. Then the deep belief propagation proposed in this thesis is developed as a further extension in the same spirit.

5.1.1 Dynamic Programming and Message Passing

Dynamic programming is a widely used method to accelerate computation. "It has an amazing history of discovery and rediscovery in different fields with different forms under different names" [50]. To list a few, backward induction for discrete-time dynamic optimization [138], Cocke-Yonger-Kasami algorithm for context-free grammar parsing [63, 67], Viterbi algorithm for hidden Markov models [135], Needleman-Wunsch algorithm for sequence alignment [96], Floyd's shortest path algorithm [39], Bellman-Ford algorithm for shortest distance in a graph [41], pixel matching in stereo vision [29, 99], seam carving in content aware image resizing [7], dynamic programming for error correcting codes [50], belief propagation for probabilistic inference [104], are all specific algorithms under the dynamic

programming principle. In this sense, dynamic programming is more like a meta-algorithm, or an algorithm design methodology.

The basic idea of the dynamic programming principle is to break a complicated problem into simpler sub-problems recursively. By tackling simpler sub-problems, the solution to the original big problem can be computed efficiently by re-using stored intermediate sub-problem results. The process of breaking a big problem into smaller problems can be represented as a tree graphically.

Message passing is a parallel implementation of dynamic programming by sending messages (information) between neighbor nodes on the tree. An interesting fact is that a global problem can be computed by sending messages among neighbors locally. Take a look at a simple counting example from [90]. Given a queue of soldiers, the goal is to let all the soldiers count the total number of soldiers in the queue (a global function of individuals). Instead of picking one soldier and letting him do the counting and then telling others, the message passing solution is simply asking each soldier to communicate with his neighbors by following the rules in Fig. 5.2. Each soldier can compute the total number of soldiers by adding the two messages received from each of its neighbors and adding one (to count himself). Fig. 5.3 shows an example of the messages computed for each soldier. The message passing rule for counting can be extended to the tree structured configuration, as shown in Fig. 5.4 and Fig. 5.5. As a side-note, similar ideas have been used to reduce the communication cost in a distribution algorithm for image-based 3-d reconstruction, reported in [86]. From this example, it can be observed that message-passing schemes can compute global summations of a set of numbers. Similarly, by replacing the local "add" operation, other global properties can be computed including min/max (for finding the shortest/tallest soldier), product and other more complex functions. In nature, many animals have developed similar strategies to communicate, for example, migratory geese can form a "V" shape by just coordinating with their neighbors without the governance of a leader.

In the above, it is shown that the dynamic programming structure can be revealed with a directed acyclic graph or a tree structured undirected graph. Message passing is a parallel distributed algorithm to compute a global function on trees. Combining these two techniques could potentially result in an efficient, parallel and distributed algorithm for computing some large-scale global quantities, including summation, integral, min, max, etc. As will be shown in the next section, belief propagation is a good example.

For each soldier,

1. If you are the front or the end of the line, say the number "1" to the soldier next to you.
2. If a soldier next to you says a number to you, add one to it, and say the new number to the soldier on the other side.

Figure 5.2: Message-passing rule for soldier counting in a queue

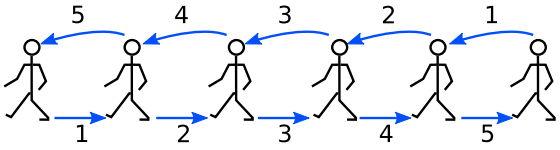


Figure 5.3: Counting soldiers in a queue with a message passing rule

For each soldier,

1. If you have only one neighbor, send message "1" to it.
2. Send a message to one neighbor, when you have received messages from all the other neighbors. The message value is the summation of all the messages from all the other neighbors + 1.

Figure 5.4: Message-passing rule for soldier counting in a tree configuration

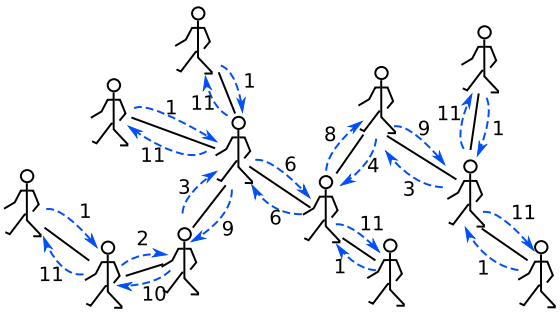


Figure 5.5: Counting soldiers in a tree configuration with a message passing rule

5.1.2 Dynamic Programming and Belief Propagation

BP was first developed as an *exact* inference algorithm for *tree-structured* graphical models [103, 66]. The same algorithm is then shown to work well for for *approximate* inference of *loopy* graphical probabilistic models [104, 95], in which case the algorithm is usually referred as Loopy Belief Propagation (LBP). As a simple example, consider the graphical model in Fig. 5.6 (adapted from [127]). The MAP estimation of x_3 can be computed as following:

$$x_3^* = \arg\min_{x_3} \left\{ \min_{x_1,x_2,x_4,x_5} \left[f_3(x_3) + f_1(x_1,x_3) + f_2(x_2,x_3) + f_4(x_3,x_4,x_5) + f_5(x_5) \right] \right\}. \tag{5.5}$$

Suppose $x_i, i = 1, \cdots, 5$ have K states. The above sum takes K^5 operations. By changing the minimization order, x_3^* can be computed more efficiently as

$$x_3^* = \arg\min_{x_3} \left\{ f_3(x_3) + \left[\min_{x_1} f_1(x_1,x_3) \right] + \left[\min_{x_2} f_2(x_2,x_3) \right] + \left[\min_{x_4,x_5} f_4(x_3,x_4,x_5) + f_5(x_5) \right] \right\}. \tag{5.6}$$

This new order of computing the sum takes $K^2 + 3K$ operations. Compared with the approach in Eq. (5.5), the computational cost is reduced. The order of minimization in Eq. (5.6) may not be easy to identify, but the factor graph shown in Fig. 5.6 reveals it clearly. In the graph, there are 4 edges connected to variable node x_3, corresponding to the 4 terms in the square bracket in Eq. (5.6). The above minimization grouping and order change is actually a form of dynamic programming; and it can be implemented with a simple message passing rule. The terms in Eq. (5.6) can be encoded with messages, defined as below.

1. The message from a random variable node to a factor node:

$$M_{x_1 \to f_1}(x_1) := \vec{0}, \tag{5.7}$$

$$M_{x_2 \to f_2}(x_2) := \vec{0}, \tag{5.8}$$

$$M_{x_4 \to f_4}(x_4) := \vec{0}, \tag{5.9}$$

$$M_{x_5 \to f_4}(x_5) := f_5(x_5). \tag{5.10}$$

2. The message from a factor node to a random variable node:

$$M_{f_1\to x_3}(x_3) := \min_{x_1} f_1(x_1, x_3) + M_{x_1\to f_1}(x_1) = \min_{x_1} f_1(x_1, x_3), \tag{5.11}$$

$$M_{f_2\to x_3}(x_3) := \min_{x_2} f_2(x_2, x_3) + M_{x_2\to f_2}(x_2) = \min_{x_2} f_2(x_2, x_3), \tag{5.12}$$

$$M_{f_3\to x_3}(x_3) := f_3(x_3), \tag{5.13}$$

$$\begin{aligned} M_{f_4\to x_3}(x_3) &:= \min_{x_4,x_5} \left[f_4(x_3, x_4, x_5) + M_{x_4\to f_4}(x_4) + M_{x_5\to f_4}(x_5)\right] \\ &= \min_{x_4,x_5} \left[f_4(x_3, x_4, x_5) + f_5(x_5)\right]. \end{aligned} \tag{5.14}$$

By combining the above messages, the marginal distribution of the variable x_3 can be computed as

$$\begin{aligned} x_3^* &= \arg\min_{x_3} \{M_{f_1\to x_3} + M_{f_2\to x_3}(x_3) + M_{f_3\to x_3}(x_3) + M_{f_4\to x_3}(x_3)\} \\ &= \arg\min_{x_3} \left\{ \left[\min_{x_1} f_1(x_1, x_3)\right] + \left[\min_{x_2} f_2(x_2, x_3)\right] + [f_3(x_3)] + \left[\min_{x_4,x_5} \left[f_4(x_3, x_4, x_5) + f_5(x_5)\right]\right] \right\}. \end{aligned} \tag{5.15}$$

It can be seen that the result from the message passing implementation is exactly the same as Eq. (5.6).

The message-passing scheme is visualized in Fig. 5.7. Take a careful look at the patterns in the above messages. It can be observed that there are two kinds of messages: messages from random variable nodes to factor nodes and messages from factor nodes to the random variables. And to compute the marginal distributions for all the random variables, it is needed to compute messages along each edge of the tree, and along both directions of the edges, as shown in Fig. 5.8. From the above, it can be observed that the belief propagation not only reduces the computational cost for estimating the marginal distribution for one random variable, but also saves computation for estimating all the marginal distributions, by re-using the messages. Here, the messages are the intermediate values in dynamic programming. Using an induction argument, this approach can be generalized to show that BP computes the exact marginal integral in any tree-structured model [104]. The exact formula is summarized in section 5.1.3

In summary, for tree-structured probabilistic graphical model, BP provides a distributed computation of the marginal or maximal distribution based on dynamic programming. Its recursive, local decomposition of maximization can provide dramatic computational savings. For example, N variables, taking one of K states, are connected by pair-wise compatibility functions, forming a tree-structured graph. If ordered appropriately, each message must

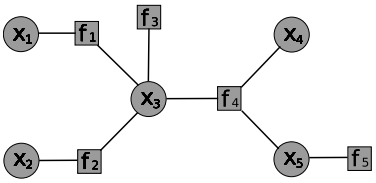


Figure 5.6: Tree-structured joint probability

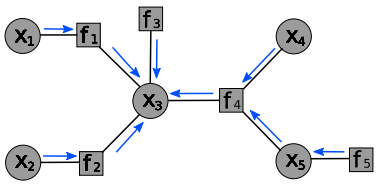


Figure 5.7: Messages for computing $P(x_3)$ on the tree-structured probabilistic graphical model

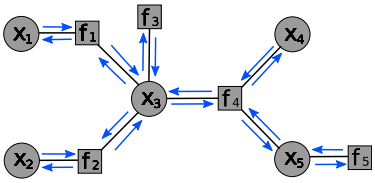


Figure 5.8: Messages for computing all the marginal distributions on the tree-structured probabilistic graphical model

only be computed once, and exact marginal distributions can be determined in $O(NK^2)$ operations. In contrast, brute force enumeration of all joint states has computational complexity of $O(K^N)$.

5.1.3 Canonical Form of Belief Propagation

With the energy formulation of the problem, the message passing algorithm for MAP estimation can be expressed in the min-sum form, as shown below.

Message sent from a clique f to its ith random variable:

$$M_{f\to x_{fi}}(x_{fi}=a)=\min_{X_f:x_{fi}=a}\left\{f(X_f)+\sum_{j\neq i}M_{x_{fj}\to f}(x_{fj})\right\} \tag{5.16}$$

Message sent from clique f's ith random variable to the clique f:

$$M_{x_{fi}\to f}(x_{fi})=\sum_{g\in C_{x_{fi}}\setminus\{f\}}M_{g\to x_{fi}}(x_{fi}) \tag{5.17}$$

MAP conditional distribution of random variable x:

$$E^*(x)=\sum_{g\in C_x}M_{g\to x}(x) \tag{5.18}$$

$$P^*(x)\propto\exp(-E^*(x)). \tag{5.19}$$

Conditional distribution of a set of random variables belonging to clique f, given the remaining variables taking their MAP states:

$$E^*(X_f)=f(X_f)+\sum_i M_{x_{fi}\to f}(x_{fi}) \tag{5.20}$$

$$P^*(X_f)\propto\exp(-E^*(X_f)). \tag{5.21}$$

notation	meaning
x_i	the voxel with index i in the volume
x_i^o	the occupancy of voxel x_i ("o" for occupancy)
x_i^c	the RGB color of voxel x_i ("c" for color)
X_r	the set of voxels on the ray r
x_{ri}	the ith voxel on the ray r
x_{ri}^o	the occupancy of voxel x_{ri}
x_{ri}^c	the color of voxel x_{ri}
R_x	the set of rays that pass through voxel x
$\ell_r(x)$	the index of voxel x on the ray r, given that $r \in R_x$
$M_{i\to r}^o(x_{ri}^o)$	i.e., $M_{x_{ri}^o \to r}(x_{ri}^o)$, the message from ray r's ith voxel occupancy variable to the ray clique
$M_{i\to r}^o$	normalized voxel occupancy to ray message
$M_{r\to i}^o(x_{ri}^o)$	the message from ray clique r to its ith voxel occupancy variable, i.e., $M_{r\to x_{ri}^o}(x_{ri}^o)$
$M_{r\to i}^o$	normalized ray to voxel occupancy message i.e., $M_{r\to i}^o(1) - M_{r\to i}^o(0)$
$M_{r\to i}^c$	the message sent from ray clique r to the its ith voxel's color variable

Table 5.2: Notations in message passing for Ray MRF

5.2 Voxel Occupancy Estimation, Given Voxel Colors

First, look at the voxel occupancy estimation problem, assuming that the voxel colors have been estimated. Here the direct application of belief propagation is adopted. The unique challenges will be exposed, and proposed solutions will be discussed later in this section.

5.2.1 Belief Propagation for Voxel Occupancy Estimation

There are three kinds of cliques in Ray MRF: the unary clique, pair-wise clique, and ray clique. Since the message from each random variable node to its adjoining factor node is the same as the typical MRFs, the focus is put on the message sent from factor node to its random variable nodes. The messages sent from the three kinds of cliques to their attending random variables are summarized as following.

unary message Since the unary clique has only one random variable involved, so its message passing is simple.

$$\text{unary clique energy:} \quad E_{uv}(x_i^o) = w_{uv}^o(1 - x_i^o)^2$$

The message from the voxel unary clique node (denoted by uv with abuse of notation) to its participating voxel occupancy variable node is:

$$M_{uv\to x_i^o}(x_i^o = 0) = w_{uv}^o(1 - x_i^o)^2 = w_{uv}^o, \tag{5.22}$$

$$M_{uv\to x_i^o}(x_i^o=1) = w_{uv}^o(1-x_i^o)^2 = 0. \tag{5.23}$$

Notice that the message sent to a binary variable x_i^o, for example, $M_{uv\to x_i^o}(x_i^o)$, is a two-dimensional vector. However only the relative difference between these two values is important, which will become clear later on. So similar as that the *normalized* distribution of a binary variable can be denoted with one parameter, the 2-d message can also be encoded with one scalar value – the difference. In the energy formulation, the *normalization* of the binary message is defined as

$$\begin{aligned} M_{uv\to x_i^o} &= M_{uv\to x_i^o}(x_i^o=1) - M_{uv\to x_i^o}(x_i^o=0) \\ &= -w_{uv}^o. \end{aligned} \tag{5.24}$$

Here the scalar-valued normalized message is represented by the same symbol as the vector-valued message by the abuse of notation. The meaning of the symbol should be clear from the context. The normalized message will be shown to be an effective way of saving computation costs (both time and memory) in the later derivations.

pair-wise message The pair-wise clique is the standard clique used in 3-d Ising models.

$$\text{pair-wise clique energy:}\quad E_p^o(x_i^o,x_j^o) = \begin{cases} 0, & (x_i^o,x_j^o)=(0,0); \\ w_p^o, & (x_i^o,x_j^o)=(0,1); \\ w_p^o, & (x_i^o,x_j^o)=(1,0); \\ 0, & (x_i^o,x_j^o)=(1,1). \end{cases}$$

The message from the pair-wise clique node (denoted by p with the abuse of notation) to one of its variable nodes x_i^o is

$$M_{p\to x_i^o}(x_i^o=0) = \min\left\{E_p^o(0,1)+M_{x_j^o\to p}(x_j^o=1), E_p^o(0,0)+M_{x_j^o\to p}(x_j^o=0)\right\} \tag{5.25}$$

$$= \min\left\{w_p^o + M_{x_j^o\to p}(x_j^o=1), 0+M_{x_j^o\to p}(x_j^o=0)\right\}, \tag{5.26}$$

$$M_{p\to x_i^o}(x_i^o=1) = \min\left\{E_p^o(1,0)+M_{x_j^o\to p}(x_j^o=0), E_p^o(1,1)+M_{x_j^o\to p}(x_j^o=1)\right\} \tag{5.27}$$

$$= \min\left\{w_p^o + M_{x_j^o\to p}(x_j^o=0), 0+M_{x_j^o\to p}(x_j^o=1)\right\}. \tag{5.28}$$

Similarly, the message can be normalized as:

$$\begin{aligned} & M_{p\to x_i^o} \\ = \; & M_{p\to x_i^o}(1) - M_{p\to x_i^o}(0) \\ = \; & \min\left\{w_p^o + M_{x_j^o\to p}(0), M_{x_j^o\to p}(1)\right\} - \min\left\{w_p^o + M_{x_j^o\to p}(1), M_{x_j^o\to p}(0)\right\} \\ = \; & \min\left\{w_p^o, M_{x_j^o\to p}(1) - M_{x_j^o\to p}(0)\right\} - \min\left\{w_p^o + M_{x_j^o\to p}(1) - M_{x_j^o\to p}(0), 0\right\} \\ = \; & \min\left\{w_p^o, M_{x_j^o\to p}\right\} - \min\left\{w_p^o + M_{x_j^o\to p}, 0\right\}. \end{aligned} \tag{5.29}$$

ray message

$$\text{ray clique energy:} \quad E_r(X_r) = ||I_r - \Phi_r(X_r)||^2$$

For simplicity, let's introduce two short notations:

$$M_{r\to i}^o(x_{ri}^o) := M_{r\to x_{ri}^o}(x_{ri}^o), \tag{5.30}$$

$$M_{i\to r}^o(x_{ri}^o) := M_{x_{ri}^o\to r}(x_{ri}^o), \tag{5.31}$$

to denote the message sent from ray clique node r to voxel occupancy variable x_{ri}^o, and the message in the reverse direction respectively. With the abuse of notations, the corresponding scalar-valued normalized message is defined as

$$M_{r\to i}^o := M_{r\to i}^o(1) - M_{r\to i}^o(0), \tag{5.32}$$

$$M_{i\to r}^o := M_{i\to r}^o(1) - M_{i\to r}^o(0). \tag{5.33}$$

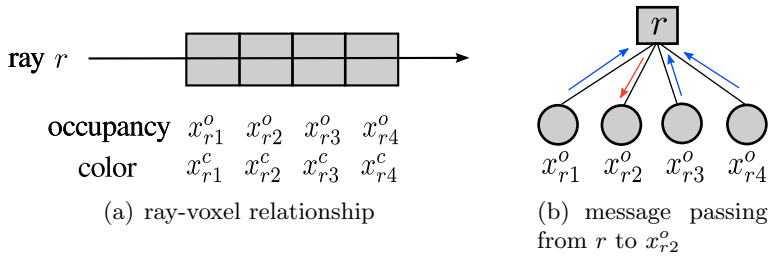
The message from a ray clique r to its ith voxel's occupancy variable is computed as:

$$M_{r\to i}^o(0) = \min_{X_r : x_{ri}^o = 0} \left\{ E_r(X_r) + \sum_{j\neq i} M_{j\to r}^o(x_{rj}^o) \right\}, \tag{5.34}$$

$$M_{r\to i}^o(1) = \min_{X_r : x_{ri}^o = 1} \left\{ E_r(X_r) + \sum_{j\neq i} M_{j\to r}^o(x_{rj}^o) \right\}. \tag{5.35}$$

To better understand the message passing formula for the ray clique, let's start from a simple example — a ray clique with 4 voxels: $E_r(x_{r1}^o, x_{r2}^o, x_{r3}^o, x_{r4}^o; x_{r1}^c, x_{r2}^c, x_{r3}^c, x_{r4}^c)$, as illustrated in Fig. 5.9. Let's derive the message sent from the ray clique node r to voxel occupancy variable x_{r2}^o, for example. When the 2nd voxel is empty, i.e., $x_{r2}^o = 0$,

$$\begin{aligned} & M_{r\to 2}^o(0) \\ = & \min_{x_{r1}^o, x_{r3}^o, x_{r4}^o} \left\{ E_r(x_{r1}^o, 0, x_{r3}^o, x_{r4}^o; x_{r1}^c, x_{r2}^c, x_{r3}^c, x_{r4}^c) + M_{1\to r}^o(x_{r1}^o) + M_{3\to r}^o(x_{r3}^o) + M_{4\to r}^o(x_{r4}^o) \right\}. \end{aligned} \tag{5.36}$$



(a) ray-voxel relationship

(b) message passing from r to x^o_{r2}

Figure 5.9: Toy ray clique example

When the 2nd voxel is solid, i.e., $x^o_{r2} = 1$,

$$M^o_{r\to 2}(1) = \min_{x^o_{r1},x^o_{r3},x^o_{r4}} \{E_r(x^o_{r1}, 1, x^o_{r3}, x^o_{r4}; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(x^o_{r1}) + M^o_{3\to r}(x^o_{r3}) + M^o_{4\to r}(x^o_{r4})\}. \tag{5.37}$$

To better understand the structure of the messages, all the possible combinations of the other three voxel occupancy states $\{x^o_{r1}, x^o_{r3}, x^o_{r4}\}$ are enumerated.

$$\begin{aligned} M^o_{r\to 2}(0) &= \min_{x^o_{r1},x^o_{r3},x^o_{r4}} \{E_r(x^o_{r1}, 0, x^o_{r3}, x^o_{r4}; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(x^o_{r1}) + M^o_{3\to r}(x^o_{r3}) + M^o_{4\to r}(x^o_{r4})\} \\ &= \min \left\{ \begin{array}{l} E_r(0,0,0,0; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(0,0,0,1; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(0,0,1,0; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(0,0,1,1; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(1), \\ E_r(1,0,0,0; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(1,0,0,1; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(1,0,1,0; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(1,0,1,1; x^c_{r1}, x^c_{r2}, x^c_{r3}, x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(1) \end{array} \right\}, \end{aligned} \tag{5.38}$$

$$
\begin{aligned}
&M^o_{r\to 2}(1)\\
&= \min_{x^o_{r1},x^o_{r3},x^o_{r4}} \{E_r(x^o_{r1},1,x^o_{r3},x^o_{r4};x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(x^o_{r1}) + M^o_{3\to r}(x^o_{r3}) + M^o_{4\to r}(x^o_{r4})\}\\
&= \min \left\{ \begin{array}{l}
E_r(0,1,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0),\\
E_r(0,1,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(1),\\
E_r(0,1,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(0),\\
E_r(0,1,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(1),\\
E_r(1,1,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(0),\\
E_r(1,1,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(1),\\
E_r(1,1,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(0),\\
E_r(1,1,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(1)
\end{array} \right\}. \qquad (5.39)
\end{aligned}
$$

As seen in Eq. (5.38) and (5.39), for each case, there are $2^{4-1} = 8$ configurations. So totally $2 \times 8 = 16$ configurations. Because the same operations need to be done for each voxel on the ray, the number of configurations to be computed is $4 \times 16 = 64$. It seems that the belief propagation can be used to solve the problem and produce a good quality result from the past experience on the quality of belief propagation for vision problems [130]. But to reconstruct a volume with 1M (1 million) voxels, it will take forever to finish. That is, the computational cost is too high. The problem comes from the minimization problem in Eqs. (5.34) and (5.35). It is a combinatorial optimization problem with n random variables (the number of voxels passed by a ray is about $100 - 1000$ in normal case). The number of combinations is $n2^n$ (i.e. $100\times 2^{100} - -1000\times 2^{1000}$), which is too large to enumerate in finite time. The inspiration for solving this problem comes from more intuitive understanding of the problem and further exploration of the problem structure, as discussed in the next subsection.

5.2.2 Deep Belief Propagation for Voxel Occupancy Estimation

Again the 8 configurations of the three variables $x^o_{r1}, x^o_{r3}, x^o_{r4}$ in Eq. (5.38) are listed in Eq. (5.43), but they are grouped into 4 cases, based on their $E_r(X_r)$ values. Recall that $E_r(X_r)$ depends only on the first non-empty voxel's color, which makes the following equations hold:

$$
E_r(0,0,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) = E_r(0,0,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) = E_r(0,0,1,\times;x^c_{r3}), \quad (5.40)
$$

and

$$\begin{aligned} & E_r(1,0,0,0;x_{r1}^c,x_{r2}^c,x_{r3}^c,x_{r4}^c) &=& E_r(1,0,0,1;x_{r1}^c,x_{r2}^c,x_{r3}^c,x_{r4}^c) \\ = \; & E_r(1,0,1,0;x_{r1}^c,x_{r2}^c,x_{r3}^c,x_{r4}^c) &=& E_r(1,0,1,1;x_{r1}^c,x_{r2}^c,x_{r3}^c,x_{r4}^c) \\ = \; & E_r(1,\times,\times,\times;x_{r1}^c). \end{aligned} \tag{5.41}$$

Substituting the above equations into Eq. (5.43) results in Eq. (5.44).

$$M^o_{r\to 2}(0)$$

$$= \min_{x^o_{r1},x^o_{r3},x^o_{r4}} \{E_r(x^o_{r1},0,x^o_{r3},x^o_{r4};x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{0\to r}(x^o_{r1}) + M^o_{2\to r}(x^o_{r3}) + M^o_{3\to r}(x^o_{r4})\} \tag{5.42}$$

$$= \min \left\{ \begin{array}{l} E_r(0,0,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ \\ E_r(0,0,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ \\ \boxed{\begin{array}{l} E_r(0,0,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(0,0,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(1), \end{array}} \\ \\ \boxed{\boxed{\begin{array}{l} E_r(1,0,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(1,0,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(1,0,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(1,0,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(1) \end{array}}} \end{array} \right\} \tag{5.43}$$

$$= \min \left\{ \begin{array}{l} E_r(0,0,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ \\ E_r(0,0,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ \\ \min \left\{ \boxed{\begin{array}{l} E_r(0,0,1,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(0,0,1,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(1) \end{array}} \right\}, \\ \\ \min \left\{ \boxed{\boxed{\begin{array}{l} E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(1) \end{array}}} \right\} \end{array} \right\} \tag{5.44}$$

The structures in the above two boxes can be compressed as following.

$$\min\left\{\begin{array}{|l}\hline E_r(0,0,1,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(0)+M^o_{3\to r}(1)+M^o_{4\to r}(0),\\ E_r(0,0,1,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(0)+M^o_{3\to r}(1)+M^o_{4\to r}(1)\\ \hline\end{array}\right\} \tag{5.45}$$

$$=\boxed{E_r(0,0,1,\times;x^c_{r3})+M^o_{3\to r}+\min\{0,M^o_{4\to r}\}+M^o_{1\to r}(0)+M^o_{3\to r}(0)+M^o_{4\to r}(0)} \tag{5.46}$$

$$\min\left\{\begin{array}{|l}\hline\hline E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(1)+M^o_{3\to r}(0)+M^o_{4\to r}(0),\\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(1)+M^o_{3\to r}(0)+M^o_{4\to r}(1),\\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(1)+M^o_{3\to r}(1)+M^o_{4\to r}(0),\\ E_r(1,\times,\times,\times;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4})+M^o_{1\to r}(1)+M^o_{3\to r}(1)+M^o_{4\to r}(1)\\ \hline\hline\end{array}\right\} \tag{5.47}$$

$$=\boxed{\boxed{E_r(1,0,\times,\times;x^c_{r1})+M^o_{1\to r}+\min\{0,M^o_{3\to r}\}+\min\{0,M^o_{4\to r}\}+M^o_{1\to r}(0)+M^o_{3\to r}(0)+M^o_{4\to r}(0)}} \tag{5.48}$$

Putting the above two compressed terms back into the formula for $M^o_{r\to 2}(0)$ (i.e., Eq. (5.44)), then

$$M^o_{r\to 2}(0) \tag{5.49}$$

$$=\min\left\{\begin{array}{|l}\hline \infty,\\ E_r(0,0,0,1;x^c_{r4})+M^o_{4\to r},\\ \boxed{E_r(0,0,1,\times;x^c_{r3})+M^o_{3\to r}+\min\{0,M^o_{4\to r}\},}\\ \boxed{\boxed{E_r(1,0,\times,\times;x^c_{r1})+M^o_{1\to r}+\min\{0,M^o_{3\to r}\}+\min\{0,M^o_{4\to r}\}}}\\ \hline\end{array}\right\}\begin{array}{l}+M^o_{1\to r}(0)\\ +M^o_{3\to r}(0)\\ +M^o_{4\to r}(0)\end{array} \tag{5.50}$$

$$=\min\left\{\begin{array}{|l}\hline \infty,\\ (I_r-x^c_{r4})^2+M^o_{4\to r},\\ \boxed{(I_r-x^c_{r3})^2+M^o_{3\to r}+\min\{0,M^o_{4\to r}\},}\\ \boxed{\boxed{(I_r-x^c_{r1})^2+M^o_{1\to r}+\min\{0,M^o_{3\to r}\}+\min\{0,M^o_{4\to r}\}}}\\ \hline\end{array}\right\}\begin{array}{l}+M^o_{1\to r}(0)\\ +M^o_{3\to r}(0)\\ +M^o_{4\to r}(0)\end{array}. \tag{5.51}$$

Similarly, the messages from the ray clique node r to the 2nd voxel, when it is solid, can be reduced as:

$$M^o_{r\to 2}(1)$$
$$= \min_{x^o_{r1},x^o_{r3},x^o_{r4}} \{E_r(x^o_{r1},1,x^o_{r3},x^o_{r4};x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(x^o_{r1}) + M^o_{3\to r}(x^o_{r3}) + M^o_{4\to r}(x^o_{r4})\} \tag{5.52}$$

$$= \min \left\{ \begin{array}{l} \boxed{\begin{array}{l} E_r(0,1,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(0,1,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(0,1,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(0,1,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(0) + M^o_{3\to r}(1) + M^o_{4\to r}(1), \end{array}} \\ \\ \boxed{\boxed{\begin{array}{l} E_r(1,1,0,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(0), \\ E_r(1,1,0,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(0) + M^o_{4\to r}(1), \\ E_r(1,1,1,0;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(0), \\ E_r(1,1,1,1;x^c_{r1},x^c_{r2},x^c_{r3},x^c_{r4}) + M^o_{1\to r}(1) + M^o_{3\to r}(1) + M^o_{4\to r}(1) \end{array}}} \end{array} \right\} \tag{5.53}$$

$$= \min \left\{ \begin{array}{l} \boxed{E_r(0,1,\times,\times;x^c_{r2}) + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\},} \\ \\ \boxed{\boxed{E_r(1,1,\times,\times;x^c_{r1}) + M^o_{1\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}}} \end{array} \right\} \begin{array}{l} +M^o_{1\to r}(0) \\ +M^o_{3\to r}(0) \\ +M^o_{4\to r}(0) \end{array} \tag{5.54}$$

$$= \min \left\{ \begin{array}{l} \boxed{(I_r - x^c_{r2})^2 + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\},} \\ \\ \boxed{\boxed{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}}} \end{array} \right\} \begin{array}{l} +M^o_{1\to r}(0) \\ +M^o_{3\to r}(0) \\ +M^o_{4\to r}(0) \end{array} . \tag{5.55}$$

Notice that both $M^o_{r\to 2}(0)$ and $M^o_{r\to 2}(1)$ have the term $M^o_{1\to r}(0) + M^o_{3\to r}(0) + M^o_{4\to r}(0)$, which will be cancelled in the normalization. To make the expression look cleaner, from now on, similar terms (that can be cancelled during normalization) will be ignored.

Through the same derivation procedure, the message passing formulas for $M^o_{r\to 1}(0)$, $M^o_{r\to 2}(0)$, $M^o_{r\to 3}(0)$, and $M^o_{r\to 4}(0)$ can be computed as:

$$M^o_{r\to 1}(0) = \min \left\{ \boxed{\begin{array}{l} \infty, \\ (I_r - x^c_{r4})^2 + M^o_{4\to r}, \\ (I_r - x^c_{r3})^2 + M^o_{3\to r} + \min\{0, M^o_{4\to r}\}, \\ (I_r - x^c_{r2})^2 + M^o_{2\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\} \end{array}} \right\} \tag{5.56}$$

$$M^o_{r\to 2}(0) = \min\left\{ \begin{array}{l} \boxed{\begin{array}{l}\infty,\\ (I_r - x^c_{r4})^2 + M^o_{4\to r},\\ (I_r - x^c_{r3})^2 + M^o_{3\to r} + \min\{0, M^o_{4\to r}\},\end{array}} \\ \\ \boxed{\boxed{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r}\} + \underline{\min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{2\to r}\}}} \end{array} \right\} \tag{5.57}$$

$$M^o_{r\to 3}(0) = \min\left\{ \begin{array}{l} \boxed{\begin{array}{l}\infty,\\ (I_r - x^c_{r4})^2 + M^o_{4\to r},\end{array}} \\ \\ \boxed{\boxed{\begin{array}{l}\underline{(I_r - x^c_{r2})^2 + M^o_{2\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{3\to r}\},\\ \underline{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{3\to r}\}\end{array}}} \end{array} \right\} \tag{5.58}$$

$$M^o_{r\to 4}(0) = \min\left\{ \begin{array}{l} \boxed{\infty,} \\ \\ \boxed{\boxed{\begin{array}{l}\underline{(I_r - x^c_{r3})^2 + M^o_{3\to r} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{4\to r}\},\\ \underline{(I_r - x^c_{r2})^2 + M^o_{2\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{4\to r}\},\\ \underline{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r} + \min\{0, M^o_{4\to r}\}\}} - \min\{0, M^o_{4\to r}\}\end{array}}} \end{array} \right\}. \tag{5.59}$$

Similarly, through the same derivation procedure, the message passing formulas for $M^o_{r\to 1}(1)$, $M^o_{r\to 2}(1)$, $M^o_{r\to 3}(1)$, and $M^o_{r\to 4}(1)$ can be computed as:

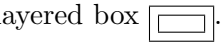
$$M^o_{r\to 1}(1) = \min\left\{ \boxed{(I_r - x^c_{r1})^2 + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} \right\} \tag{5.60}$$

$$M^o_{r\to 2}(1) = \min\left\{ \begin{array}{l} \boxed{(I_r - x^c_{r2})^2 + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\},} \\ \\ \boxed{\boxed{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r}\} + \underline{\min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{2\to r}\}}} \end{array} \right\} \tag{5.61}$$

$$M^o_{r\to 3}(1) = \min\left\{ \begin{array}{l} \boxed{(I_r - x^c_{r3})^2 + \min\{0, M^o_{4\to r}\},} \\ \\ \boxed{\boxed{\begin{array}{l}\underline{(I_r - x^c_{r2})^2 + M^o_{2\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{3\to r}\},\\ \underline{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{3\to r}\}\end{array}}} \end{array} \right\} \tag{5.62}$$

$$M^o_{r\to 4}(1) = \min\left\{ \begin{array}{l} \boxed{(I_r - x^c_{r4})^2} \\ \\ \boxed{\boxed{\begin{array}{l}\underline{(I_r - x^c_{r3})^2 + M^o_{3\to r} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{4\to r}\},\\ \underline{(I_r - x^c_{r2})^2 + M^o_{2\to r} + \min\{0, M^o_{3\to r}\} + \min\{0, M^o_{4\to r}\}} - \min\{0, M^o_{4\to r}\},\\ \underline{(I_r - x^c_{r1})^2 + M^o_{1\to r} + \min\{0, M^o_{2\to r}\} + \min\{0, M^o_{3\to r} + \min\{0, M^o_{4\to r}\}\}} - \min\{0, M^o_{4\to r}\}\end{array}}} \end{array} \right\} \tag{5.63}$$

By examining the above equations from (5.56) to (5.63), the following patterns can be observed:

- $M^o_{r\to i}(0)$ and $M^o_{r\to i}(1)$ share exactly the same component, highlighted by the double layered box  .
- The underlined structures in  for $M^o_{r\to 2}(\cdot)$, $M^o_{r\to 3}(\cdot)$ and $M^o_{r\to 4}(\cdot)$ have a recursive structure. More specifically, the underlined structures in  for $M^o_{r\to 2}(\cdot)$ is a sub-component of the corresponding structure in $M^o_{r\to 3}(\cdot)$, and the same relationship exists in between $M^o_{r\to 3}(\cdot)$ and $M^o_{r\to 4}(\cdot)$.
- The structures in  for $M^o_{r\to 1}(0)$, $M^o_{r\to 2}(0)$, $M^o_{r\to 3}(0)$ and $M^o_{r\to 4}(0)$ have the recursive structure. More specifically, for $M^o_{r\to 3}(0)$ is a sub-component of the corresponding structure in $M^o_{r\to 2}(0)$, and the same relationship exists for $M^o_{r\to 2}(0)$ and $M^o_{r\to 1}(0)$.
- The structures in  and  also share most of the terms.

The above repeated patterns in the messages reminds us of the dynamic programming algorithm. Based on dynamic programming, an efficient algorithm is developed to accomplish the computation efficiently. Consider a general ray clique involving n voxels: $E_r(x^o_{r1}, x^o_{r2}, \cdots, x^o_{rn}; x^c_{r1}, x^c_{r2}, \cdots, x^c_{rn})$. Define $E^i_r := (I_r - x^c_{ri})^2, \quad i = 1, 2, \cdots, n$. By generalizing the formulas for the toy example, the message from ray clique node r to its ith

voxel, $M^o_{r\to i}(\cdot)$ can be written as:

$$M^o_{r\to i}(0) = \min\left\{\begin{array}{l}\boxed{\begin{array}{l}\infty,\\ E^n_r + M^o_{n\to r},\\ E^{n-1}_r + M^o_{n-1\to r} + \min\{0, M^o_{n\to r}\},\\ \cdots\\ E^{i+1}_r + M^o_{i+1\to r} + \sum_{k=i+1+1}^{n}\min\left\{0, M^o_{k\to r}\right\},\end{array}}\\ \\ \boxed{\boxed{\begin{array}{l}E^{i-1}_r + M^o_{i-1\to r} + \sum_{k=i-1+1}^{n}\min\left\{0, M^o_{k\to r}\right\} - \min\left\{0, M^o_{i\to r}\right\},\\ \cdots,\\ E^1_r + M^o_{1\to r} + \sum_{k=1+1}^{n}\min\left\{0, M^o_{1\to r}\right\} - \min\left\{0, M^o_{i\to r}\right\}\end{array}}}\end{array}\right\} \tag{5.64}$$

$$= \min\left\{\begin{array}{l}\boxed{\min\left\{\begin{array}{l}\infty,\\ E^n_r + M^o_{n\to r},\\ E^{n-1}_r + M^o_{n-1\to r} + \min\{0, M^o_{n\to r}\},\\ \cdots\\ E^{i+1}_r + M^o_{i+1\to r} + \sum_{k=i+1+1}^{n}\min\left\{0, M^o_{k\to r}\right\}\end{array}\right\}}\\ \\ \boxed{\boxed{\min\left\{\begin{array}{l}E^{i-1}_r + M^o_{i-1\to r} + \sum_{k=i-1+1}^{n}\min\left\{0, M^o_{k\to r}\right\},\\ \cdots\\ E^1_r + M^o_{1\to r} + \sum_{k=1+1}^{n}\min\left\{0, M^o_{1\to r}\right\}\end{array}\right\} - \min\left\{0, M^o_{i\to r}\right\}}}\end{array}\right\} \tag{5.65}$$

$$M^o_{r\to i}(1) = \min\left\{ \begin{array}{l} E^i_r + \sum_{k=i+1}^n \min\{0, M^o_{k\to r}\}, \\ \\ \boxed{\begin{array}{l} E^{i-1}_r + M^o_{i-1\to r} + \sum_{k=i-1+1}^n \min\{0, M^o_{k\to r}\} - \min\{0, M^o_{i\to r}\}, \\ \cdots \\ E^1_r + M^o_{1\to r} + \sum_{k=1+1}^n \min\{0, M^o_{k\to r}\} - \min\{0, M^o_{i\to r}\} \end{array}} \end{array} \right\} \tag{5.66}$$

$$= \min\left\{ \begin{array}{l} E^i_r + M^o_{i\to r} + \sum_{k=i+1}^n \min\{0, M^o_{k\to r}\} - M^o_{i\to r}, \\ \\ \boxed{\min\left\{ \begin{array}{l} E^{i-1}_r + M^o_{i-1\to r} + \sum_{k=i-1+1}^n \min\{0, M^o_{k\to r}\}, \\ \cdots \\ E^1_r + M^o_{1\to r} + \sum_{k=1+1}^n \min\{0, M^o_{k\to r}\} \end{array} \right\} - \min\{0, M^o_{i\to r}\}} \end{array} \right\} \tag{5.67}$$

Denote

$$E^{oi}_r := \min\{0, M^o_{i\to r}\}, \tag{5.68}$$

and

$$E^{*i}_r := E^i_r + M^o_{i\to r} + \sum_{k=i+1}^n E^{ok}_r \tag{5.69}$$

Then $M^o_{r\to i}(0)$ and $M^o_{r\to i}(1)$ can be simplified as

$$M^o_{r\to i}(0) = \min\left\{ \begin{array}{l} \boxed{\min\{E^{*n}_r, E^{*(n-1)}_r, \cdots, E^{*(i+1)}_r\}} \\ \\ \boxed{\boxed{\min\{E^{*1}_r, E^{*2}_r, \cdots, E^{*(i-1)}_r\} - E^{oi}_r}} \end{array} \right\} \tag{5.70}$$

$$M^o_{r\to i}(1) = \min\left\{ \begin{array}{l} E^{*i}_r - M^o_{i\to r}, \\ \\ \boxed{\boxed{\min\{E^{*1}_r, E^{*2}_r, \cdots, E^{*(i-1)}_r\} - E^{oi}_r}} \end{array} \right\} \tag{5.71}$$

Denote

$$E^{*\uparrow}_r[i] := \min\{E^{*n}_r, E^{*(n-1)}_r, \cdots, E^{*(i+1)}_r\}, \tag{5.72}$$

$$E^{*\downarrow}_r[i] := \min\{E^{*1}_r, E^{*2}_r, \cdots, E^{*(i-1)}_r\}, \tag{5.73}$$

$$E^{o\uparrow}_r[i] := \sum_{k=i+1}^n E^{ok}_r, \tag{5.74}$$

$M^o_{r\to i}(0)$ and $M^o_{r\to i}(1)$ can be further simplified as

$$M^o_{r\to i}(0) = \min \left\{ \begin{array}{l} \boxed{E^{*\uparrow}_r[i],} \\ \\ \boxed{\boxed{E^{*\downarrow}_r[i] - E^{oi}_r}} \end{array} \right\}, \tag{5.75}$$

$$M^o_{r\to i}(1) = \min \left\{ \begin{array}{l} E^{*i}_r - M^o_{i\to r}, \\ \\ \boxed{\boxed{E^{*\downarrow}_r[i] - E^{oi}_r}} \end{array} \right\}. \tag{5.76}$$

Then the normalized message $M^o_{r\to i}$ is computed as

$$M^o_{r\to i} = M^o_{r\to i}(1) - M^o_{r\to i}(0) \tag{5.77}$$

$$= \min\left\{E^{*i}_r - M^o_{i\to r}, E^{*\downarrow}_r[i] - E^{oi}_r\right\} - \min\left\{E^{*\uparrow}_r[i], E^{*\downarrow}_r[i] - E^{oi}_r\right\}. \tag{5.78}$$

The above operation is summarized in Algorithm 1. In the algorithm, there are 5 sweeps of the n voxels on the ray (3 forward sweeps and 2 backward sweeps). That is, the computational complexity is $5n$*, or* $O(n)$*. For a clique with n binary random variables, the direct implementation of belief propagation takes* $O(n \times 2^n)$ *computation. Here for optimized message passing of ray clique, the computation is reduced to* $O(n)$*. The dramatic reduction of the computational complexity makes the Ray MRF inference from intractable to efficient.*

5.3 Voxel Color Estimation, Given Voxel Occupancies

The last section has discussed the estimation of voxel occupancies with the deep belief propagation algorithm, given that voxel colors are known. Here the focus turns to the voxel color estimation. A natural question is: can the voxel colors be estimated with a similar approach, i,e., deep belief propagation? One obvious obstacle towards this direction is that the color is a 3-dimensional real-valued random variable, instead of binary variable. Treating it as a discrete variable is also problematic, because the number of states is too large ($256^3 = 16777216$). Approximating the messages with mixture of Gaussians and using the non-parametric belief propagation (e.g., [128]) may be a solution. Is there a better solution?

As shown in Fig. 5.10, the voxel is visible in 4 views (y_1, y_2, y_4, y_5). Then the marginal mean estimation of the color is the mean of the color of these 4 visible rays. What does this mean to the color estimation problem? It means that *given the voxel occupancies, the voxel color can be estimated in a closed form.* In Ray MRF, the visibility of each voxel along a

Algorithm 1 Message passing from ray cliques to voxel occupancy variables, i.e., computing $M^o_{r\to i}$

for each ray r **do**
  for $i = 1$ to n **do**
    $M^o_r[i] = M^o_{i\to r}$
    $E^o_r[i] = \min(0, M^o_r[i])$
    $E^c_r[i] = (I_r - x^c_{ri})^2$
  end for
  $E^{o\uparrow}_r[n] = 0$
  $E^*_r[n] = E^c_r[n] + M^o_r[n]$
  for $i = n-1$ to 1 **do**
    $E^{o\uparrow}_r[i] = E^{o\uparrow}_r[i+1] + E^o_r[i+1]$
    $E^*_r[i] = E^c_r[i] + M^o_r[i] + E^{o\uparrow}_r[i]$
  end for
  $E^{*\uparrow}_r[n] = \infty$
  for $i = n-1$ to 1 **do**
    $E^{*\uparrow}_r[i] = \min(E^{*\uparrow}_r[i+1], E^*_r[i+1])$
  end for
  $E^{*\downarrow}_r[1] = \infty$
  for $i = 2$ to n **do**
    $E^{*\downarrow}_r[i] = \min(E^{*\downarrow}_r[i-1], E^*_r[i-1])$
  end for
  for $i = 1$ to n **do**
    $M^o_{r\to i}(0) = \min(E^{*\downarrow}_r[i] - E^o_r[i], E^{*\uparrow}_r[i])$
    $M^o_{r\to i}(1) = \min(E^{*\downarrow}_r[i] - E^o_r[i], E^*_r[i] - M^o_r[i])$
    $M^o_{r\to i} = M^o_{r\to i}(1) - M^o_{r\to i}(0)$
  end for
end for

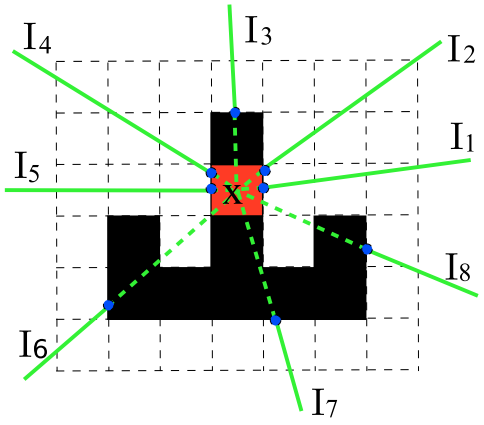


Figure 5.10: Voxel's visibilities in different views, given binary voxel occupancies. White squares denote empty voxels; dark squares denote solid voxels.

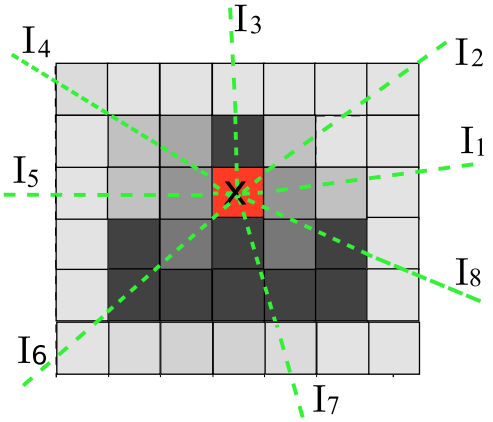


Figure 5.11: Voxel's visibilities in different views, given probabilistic voxel occupancies. Darker squares denote higher probability of the voxel being solid.

ray is measured in probability, as shown in Fig. 5.11. Formally the color of a voxel, x^c, is estimated as

$$\overline{x^c} = \frac{\sum_{r\in R_x} \overline{\mathsf{vis}}_r(x) I_r}{\sum_{r\in R_x} \overline{\mathsf{vis}}_r(x)} \tag{5.79}$$

where

$$\begin{aligned} \overline{\mathsf{vis}}_r(x) &= \mathbb{E}_{x^o_{r(\ell_r(x)+1)}}[\cdots \mathbb{E}_{x^o_{rn}}[P_r(\underbrace{0,0,\cdots,0}_{\ell_r(x)-1},1,x^o_{r(\ell_r(x)+1)},\cdots,x^o_{rn})]\cdots] \\ &= P_r(\underbrace{0,0,\cdots,0}_{\ell_r(x)-1},1,\times,\cdots,\times). \end{aligned} \tag{5.80}$$

is the mean visibility of voxel x along ray r, R_x is the set of rays that pass through the voxel x, $\ell_r(x)$ is the index of voxel x on the ray r, $\mathbb{E}_{x^o_{ri}}[\cdot]$ is the mathematical expectation operator over x^o_{ri}.

The color estimate, based on the maximal visibility, is computed as

$$x^{c*} = \frac{\sum_{r\in R_x} \mathsf{vis}^*_r(x) I_r}{\sum_{r\in R_x} \mathsf{vis}^*_r(x)} \tag{5.81}$$

where

$$\begin{aligned} \mathsf{vis}^*_r(x) &= \max_{x^o_{r(\ell_r(x)+1)}}\{\cdots \max_{x^o_{rn}}\{P_r(\underbrace{0,0,\cdots,0}_{\ell_r(x)-1},1,x^o_{r(\ell_r(x)+1)},\cdots,x^o_{rn})\}\cdots\} \\ &= P^*_r(\underbrace{0,0,\cdots,0}_{\ell_r(x)-1},1,\times,\cdots,\times). \end{aligned} \tag{5.82}$$

is the maximal visibility of voxel x along ray r. To be consistent with the occupancy estimation, the maximal visibility is used, instead of the expected (mean) visibility.

There are two approaches to compute the visibility probability $\mathsf{vis}_r^*(x)$.

product of marginals: In this approach, the visibility probability is computed as the product of marginal probability of each voxel along the ray, i.e.,

$$P_r(x_{r1}^o, x_{r2}^o, \cdots, x_{rn}^o) = P_r(x_{r1}^o)P_r(x_{r2}^o)\cdots P_r(x_{rn}^o). \tag{5.83}$$

$$\begin{aligned} \mathsf{vis}_r^*(x) &= \max_{x^o_{r(\ell_r(x)+1)}} \cdots \max_{x^o_{rn}} P_r(\underbrace{0,0,\cdots,0}_{\ell_r(x)-1}, 1, x^o_{r(\ell_r(x)+1)}, \cdots, x^o_{rn}) \\ &= P(x_{r1}^o = 0)P(x_{r2}^o = 0)\cdots P(x^o_{r(\ell_r(x)-1)} = 0)P(x^o_{r\ell_r(x)} = 1) \\ &\quad \max_{x^o_{r(\ell_r(x)+1)}} P(x^o_{r(\ell_r(x)+1)}) \cdots \max_{x^o_{rn}} P(x^o_{rn}). \end{aligned} \tag{5.84}$$

The marginal distribution of each voxel can be computed by the deep belief propagation algorithm as discussed in last section.

joint distribution: In this approach, the visibility probability is computed as the true joint probability without the independence approximation. Recall that in belief propagation, not only each variable's marginal/maximal distribution can be computed, but also a group of variables' joint marginal/maximal distribution, or more accurately speaking, a clique (or factor)'s marginal distribution.

Toy Example: Again the toy example $E_r(x_{r1}^o, x_{r2}^o, x_{r3}^o, x_{r4}^o; x_{r1}^c, x_{r2}^c, x_{r3}^c, x_{r4}^c)$ (Figs. 5.9) can be used to show the essential ideas. The maximal visibility of the 2nd voxel can be computed as

$$P_r^*(0, 1, \times, \times) = \frac{e^{-E_r^*(0,1,\times,\times;x_{r2}^c)}}{e^{-E_r^*(\times,\times,\times,\times)}} \tag{5.85}$$

$E_r^*(0, 1, \times, \times; x_{r2}^c)$ can be computed as

$$\begin{aligned} & E_r^*(0, 1, \times, \times; x_{r2}^c) \\ = \;& \min(E_r^*(0, 1, 0, 0; x_{r2}^c), E_r^*(0, 1, 0, 1; x_{r2}^c), E_r^*(0, 1, 1, 0; x_{r2}^c), E_r^*(0, 1, 1, 1; x_{r2}^c)) \\ = \;& \min\left\{ \begin{array}{l} E_r(0, 1, 0, 0; x_{r2}^c) + M^o_{2\to r}, \\ E_r(0, 1, 0, 1; x_{r2}^c) + M^o_{2\to r} + M^o_{4\to r}, \\ E_r(0, 1, 1, 0; x_{r2}^c) + M^o_{2\to r} + M^o_{3\to r}, \\ E_r(0, 1, 1, 1; x_{r2}^c) + M^o_{2\to r} + M^o_{3\to r} + M^o_{4\to r} \end{array} \right\} \\ = \;& E_r(0, 1, \times, \times; x_{r2}^c) + M^o_{2\to r} + \min(0, M^o_{3\to r}) + \min(0, M^o_{4\to r}) \end{aligned} \tag{5.86}$$

$E_r^*(\times, \times, \times, \times)$ can be computed as

$$\begin{aligned} & E_r^*(\times, \times, \times, \times) \\ = \;& \min\{E_r^*(1, \times, \times, \times; x_{r1}^c), E_r^*(0, 1, \times, \times; x_{r2}^c), E_r^*(0, 0, 1, \times; x_{r3}^c), E_r^*(0, 0, 0, 1; x_{r4}^c)\} \end{aligned} \tag{5.87}$$

General Case: For the case with n voxels on a ray, the maximal visibility of the ith voxel along a ray can be computed as following. Denote $P_r^{*i} = P_r^*(\underbrace{0, 0, \cdots, 0}_{i-1}, 1, \times, \cdots, \times)$. The associated minimal energy distribution of the ith voxel along the ray is $E_r^{*i} = E_r^*(\underbrace{0, 0, \cdots, 0}_{i-1}, 1, \times, \cdots, \times)$. It can be computed as

$$E_r^{*i} = E_r^i + M_{i\to r} + \sum_{j=i+1}^{n} \min(0, M_{j\to r}). \tag{5.88}$$

Denote $E_r^{*0} = E_r^*(\times, \cdots, \times)$. It can computed as

$$E_r^{*0} = E_r^*(\times, \cdots, \times) = \min_{i=1,\cdots,n} E_r^{*i}. \tag{5.89}$$

Then the maximal visibility can be computed as

$$P_r^{*i} = \frac{e^{-E_r^{*i}}}{e^{-E_r^{*0}}} = e^{-(E_r^{*i} - E_r^{*0})}. \tag{5.90}$$

A good news is that E_r^{*i} is exactly the term in Eq. (5.69), which has been computed in the occupancy estimation in Algorithm 1 (the term $E_r^*[i]$ in line 11), which can be re-used here.

The above operation can also be implemented with a message passing scheme. Denote $M_{r\to i}^c$ as the message sent from the ray r to voxel color variable x_i^c. Note that the message $M_{r\to i}^c$ is not a distribution as in the belief propagation. It has two values: visibility of the voxel x_{ri} on the ray r, and ray color I_r weighted by the visibility.

$$M_{r\to i}^c := (\mathsf{vis}_r^{*i}, \mathsf{vis}_r^{*i} * I_r) \tag{5.91}$$

The value of each voxel color can be computed by accumulating the messages received from all the rays that passes through the voxel:

$$x^{*c} = \frac{\sum_{r\in R_x} M_{r\to \ell_r(x)}^c[2]}{\sum_{r\in R_x} M_{r\to \ell_r(x)}^c[1]} \tag{5.92}$$

where $M_{r\to \ell_r(x)}^c[1]$ and $M_{r\to \ell_r(x)}^c[2]$ denote respectively the first and second value of the 2-d vector $M_{r\to \ell_r(x)}^c$. The above voxel color updating algorithm is summarized in Algorithm 2.

5.4 Algorithm Summary

The core algorithms for computing the ray messages have been summarized in Algorithms 1 and 2. Here the whole information flow from input images to the reconstructed surface mesh is illustrated with a system diagram in Fig.5.12. In the diagram, on the left are

Algorithm 2 Computing the message sent from ray clique to voxel variables (including both occupancy and color), i.e., computing $M^o_{r\to i}$ and $M^c_{r\to i}$

for each ray r **do**
  compute the occupancy messages as in Algorithm 1.
  $E^{*0}_r = \infty$
  for $i = 1$ to n **do**
    $E^{*0}_r = \min(E^{*0}_r, E^*_r[i])$
  end for
  for $i = 1$ to n **do**
    $\mathsf{vis}^*_r[i] = e^{-(E^*_r[i]-E^{*0}_r)}$;
    $M^c_{r\to i} = (\mathsf{vis}^*_r[i], \mathsf{vis}^*_r[i] * I_r)$
  end for
end for

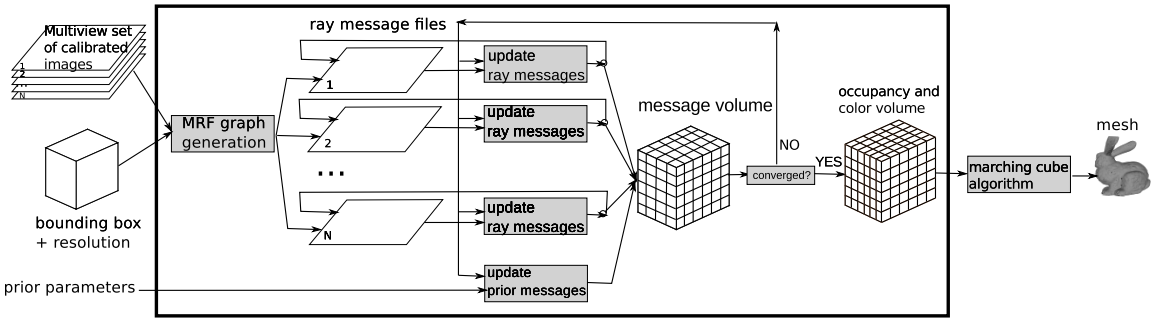


Figure 5.12: Information flow in the inverse ray tracing algorithm

the inputs of the system: a multi-view image set, a bounding box of the scene, and a set of parameters controlling the Ray MRF (e.g., w_r, w^o_p, w^o_{uv}, etc.) On the right is the output of the system: an occupancy-color volume or a polygon mesh converted from the volume by an iso-thresholding algorithm, such as marching cube. Inside the processing box, there are two significant parts: one is the Ray MRF graph generation, which computes the ray-voxel relationship. This step is done one time only. The other part is the core of the whole procedure: iterative message passing between voxels and cliques, in particular the ray cliques. The message passing for the ray cliques are totally in parallel to the rays. Combined with the message passing for unary and pairwise cliques, the whole algorithm can run in parallel. By comparing the proposed system diagram with the pipeline of the depth fusion approach as shown in Fig. 2.1 and the pipeline of the feature expansion approach as shown in Fig. 2.2, it can be seen that the traditional multi-step multi-view pipeline has been replaced with an one-step reconstruction with all the constraints processed together and no sub-optimal hard decisions made in the middle steps.

5.5 Algorithm Convergence

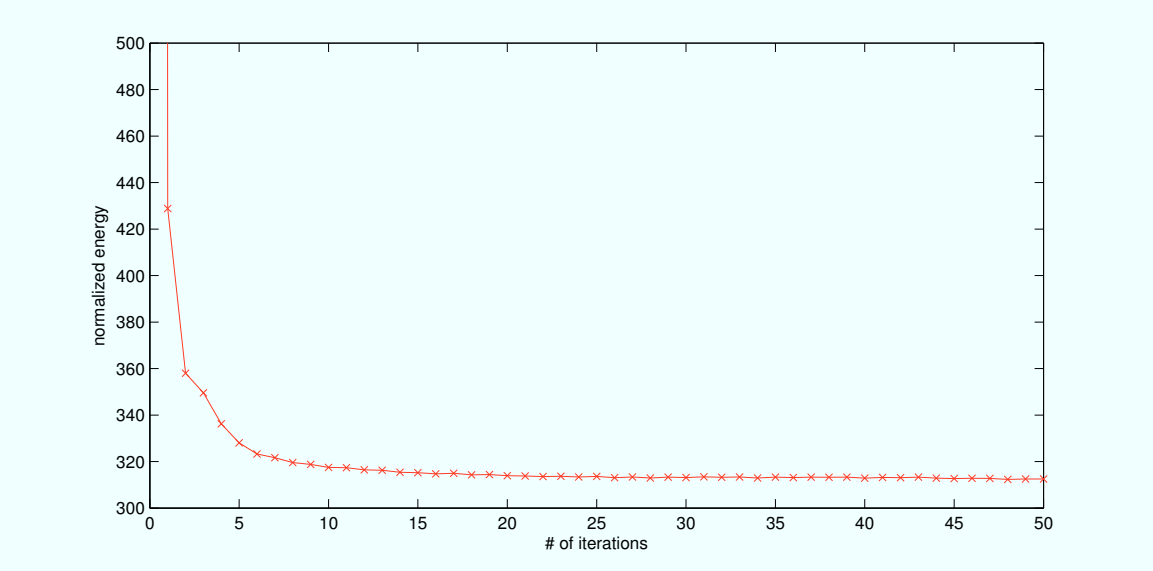
The theoretical convergence property of the proposed algorithm is unknown and is yet to be analyzed in the future. Here its typical convergence behavior is reported empirically. The experiment is conducted on the kermit dataset, which is provided by the Bundler software package [123]. The dataset consists of 11 VGA-resolution images taken under normal indoor lighting condition. Fig. 5.13 (a) shows the change of the normalized energy (defined as the whole Ray MRF energy divided by the number of rays) as the number of iterations increases. Fig. 5.13 (b) shows the rendered images of the reconstructed 3-d model from three different views and their changes as the algorithm iterates. It can be seen that 1) the algorithm converges in about 10-20 iterations; 2) the energy decreases exponentially; and 3) the rendered images become increasingly sharper as the number of iterations increases.

Here the convergence of the two ways of estimating the visibility, i.e., product of marginal and joint distribution, is also compared. Fig. 5.14 plots the normalized energy change with respect to the number of iterations. It can be seen that the two methods' performances are close, with the joint distribution method having slightly lower energy, i.e., better estimation.

5.6 Related Work

In this chapter, an algorithm for the MAP (maximum a posteriori) estimation of voxel occupancies and colors is developed based on loopy belief propagation (LBP) and dynamic programming (DP). The high computational cost problem in directly applying LBP has been solved by exploring the recursive chain structure of the volumetric ray tracing function with DP. The recursive chain structure of the problem comes from the visibility property: only the first visible voxel accounts for the pixel/ray observation. This work is not the first one to use this visibility property, it has been explored by other works in different contexts to acclerate the computation, especially the volumetric methods, including space carving [118, 119, 73], probabilistic space carving [19, 18] and voxel world model [106, 105, 30].

In space carving, the visibility property is utilized to carve out the voxels from outside to inside in a special order to get a linear complexity algorithm. In probabilistic space carving, the visibility is modeled with probability. The probabilistic visibility is computed under the voxel independence assumption. Each voxel is updated based on its visibility weighted Bayesian decision of the voxel-existence model and voxel-nonexistence model. Then the voxel world model takes the visiblity property into full use by expanding the pixel/ray



(a) The change of Ray MRF's normalized energy as the number of iterations increases

(b) From left to right: rendered images from 3 different views at iteration no. 1, 2, 5, 10, and 50.

Figure 5.13: Convergence behavior of the proposed algorithm

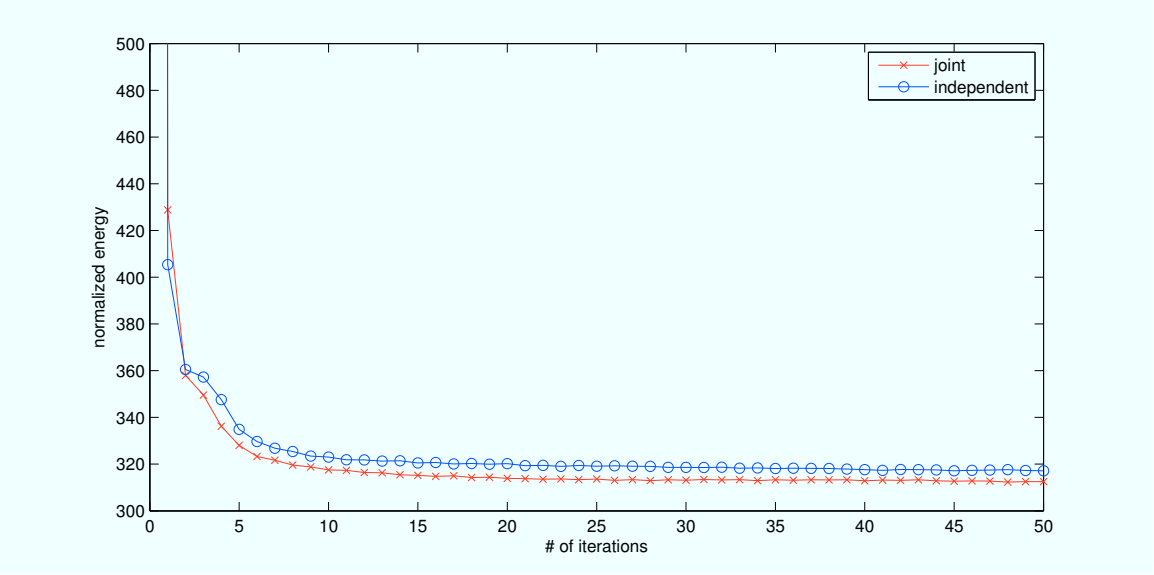


Figure 5.14: Convergence comparison of two visibility estimation methods: product of marginal and joint distribution

intensity probability distribution over the first visible surface voxels. This work and the voxel world model work use similar kind of factorization to accelerate the computation in different estimation framework.

Chapter 6

Modeling Background and Transient Clutter

A weed is simply a plant that wants to grow where people want something else.
Weeds are people's idea, not nature's.
— Author Unknown

In the last chapter, the estimation of a 3-d volume from multiple images has been studied. Can the current model and algorithm handle most kinds of scenes? Frist take a careful look at the typical datasets in Fig. 1.6, especially images (e) and (f). It can be seen that (1) besides the "object of interest", there are other scene structures, e.g., trees, background buildings, moving vehicles, sky, etc.; (2) these scene structures could span a large area. How to handle these structures? Put them in the volume or ignore them? In this chapter, these problems are formally studied. A compromise solution is proposed by modeling some structures and discarding others, balancing the computational cost, reconstruction accuracy and robustness of the system. The boundary between what to be modeled and what to be discarded is not fixed. It depends on the development of what can be modeled efficiently.

6.1 "Volume + Bubble" Model

6.1.1 Domain of Localization (DOL)

The "object of interest" is a subjective concept. Without additional information, it is almost impossible for the algorithm to figure out which part of the scene is of interest and which part is not. Exceptional cases include: 1) the user intentionally sets up a simple

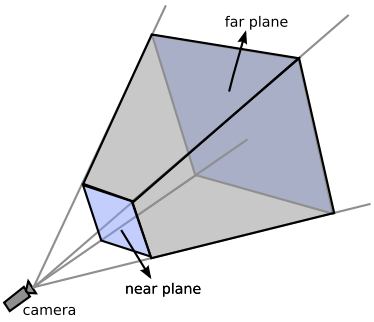


Figure 6.1: Camera frustum

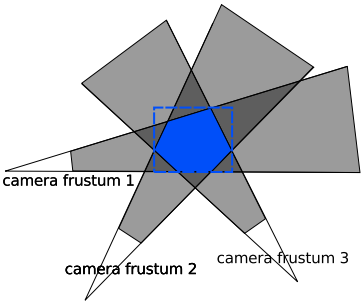


Figure 6.2: Intersection of camera frustums

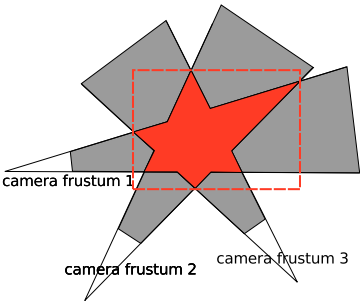


Figure 6.3: Domain of Localization (DOL)

background, which enables easy segmentation of the object of interest from the background, as shown in the Middlebury dataset (Fig. 7.2 and 7.5); 2) or, the user intentionally captures the full object in every shot, then the intersection of the viewing cones can serve as a fairly tight bounding box of the object of interest, as shown in Fig. 6.2. The second case is more difficult than the first one to deal with, because the projected bounding box to each image also contains pixel values from objects of no interest. But still in most applications, the object of interest cannot be easily segmented out and its tight bounding box cannot be easily figured out.

To make the concept operable, here a new concept "domain of localization (DOL)" is introduced, substituting for the subjective concept "object of interest". The DOL for a configuration of cameras is defined as the smallest domain outside of which no point can be localized. The DOL can be used to find the optimal bounding box for 3-d reconstruction as shown in Fig. 6.3.

DOL depends on several parameters, including cameras' location and orientation, cameras' depth of view, and baseline among cameras, etc. It can be computed through intersection of the camera frustums, as shown in Fig. 6.3. Camera frustum or viewing frustum, is traditionally defined as the region of space that has the object in focus. The planes that cut the frustum perpendicular to the viewing direction are called the near plane and the far plane, as shown in Fig. 6.1. Objects closer to the camera than the near plane or beyond the far plane are too out-of-focus (blurred) to be reconstructed. This concept is related to depth of field (DOF). DOF is the portion of a scene that appears acceptably sharp in the image. The DOF is determined by the camera-to-subject distance, focal length, f-number, and the format size or circle of confusion criterion. There is an empirical equation for computing the DOF [111]. Here the concept of "camera frustum" is extended to "camera localization

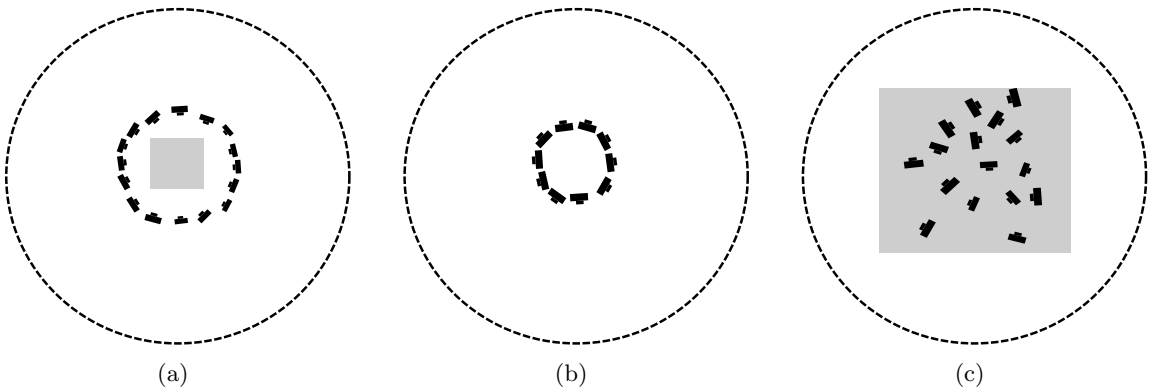


Figure 6.4: Typical camera configurations. (a) circular camera motion around an object; (b) camera rotating around a center to create panoramic view of the surrounding scene; (c) irregular camera motion.

frustum", which is defined as the region of space that has the object localized. To be localized, the object should stay in focus and should have enough baseline between this camera and other cameras. Technically, the determination of the far plane is extended to account for the case of small stereo baseline. With small baseline, the further the point is, the larger the localization uncertainty. For points beyond some specific point, the uncertainty is too large to carry any useful information. So the far plane is constrained by the baseline. In practice, the far plane of one camera can be set as $f * d_{max}$, where f is the focal length, d_{max} is the largest baseline between this camera and the other cameras.

6.1.2 "Bubble" Background Model

There is a large number of pixel observations coming from structures outside DOL. Although these structures cannot be localized by the set of captured photos, a model is needed to explain the corresponding pixel observations. Otherwise, artifacts will be introduced in the volume to explain these pixel observations. How to model these scene structures? Here a spherical background is used to explain all the pixels that cannot be localized within DOL. Let's consider a special case of the camera configuration: all the cameras are around a common center, as shown in Fig. 6.4(b). From the definition of DOL, since the baseline is zero, the DOL is empty, so everything should be explained with the background at infinity. This is exactly what the panoramic photography does.

To represent the spherical image with an image requires a 2-d parametrization of the sphere, which maps a spherical globe of the earth onto a flat piece of paper. Since distortion

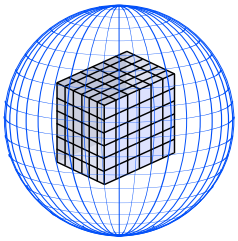
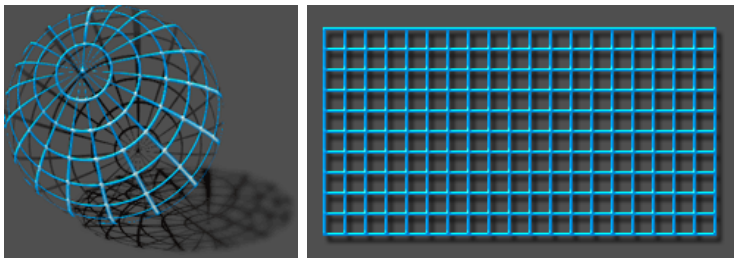


Figure 6.5: 3-d representation of scene: volume + background panorama



(a) (b)

Figure 6.6: Spherical panorama [93]

is inevitable when trying to map a spherical image into a flat surface, several parametrization/projections exist, with each one trying to minimize one type of distortion at the expense of others. For example, the cylindrical image projection maps the latitude and longitude coordinates of a spherical globe directly onto horizontal and vertical coordinates of a grid, as shown in Fig. 6.6. Horizontal stretching therefore increases further from the poles, with the north and south poles being stretched across the entire upper and lower edges of the flattened grid. [93] provides a good summary of these projections. As for choosing the right spherical image representation, one concern is to build prior MRF models for the spherical background easily. For this purpose, each pixel on the flat image should roughtly correspond to approximately the same size of area on the spherical image, and the neighborhood between pixels should be kept. Due to the strong stretching effect on the pole region, the cylindrical representation is not a good choice. In this thesis, the cube approximation of the sphere is used, as shown in Fig. 6.7. In the cube approximation, the spherical image is represented with 6 face images on the cube. These 6 face images can then be expanded to a long 2-d image from face 0 to 5, as shown in Fig. 6.7.

6.1.3 Spherical Background Estimation

Here the same two properties – occupancy and color – are assigned to the background pixel. The occupancy of the background is an abuse of terminology, by voxels and background pixels having the same properties to simplify the later notations. The occupancy of the background pixel indicates whether this pixel is being observed. If it has been observed, then it has a value 1, otherwise value 0. With the new scene representation, how to update the equations in Chapter 5 to estimate both volume and background? First the problem formulation is updated to include the spherical background variables.

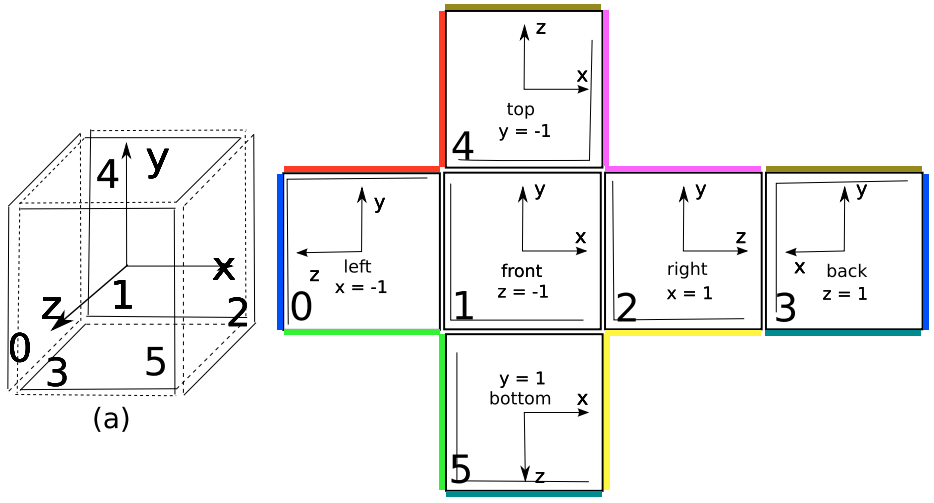


Figure 6.7: Cube representation of the 360° panorama

$$E(X) = \sum_{r \in R} E_r(X_r) + w^o_{uv} \sum_{k \in V} E^o_{uv}(x^o_k) + w^o_{ub} \sum_{k \in B} E^o_{ub}(x^o_k) + w^o_p \sum_{\langle i,j \rangle \in \mathcal{N}} E^o_p(x^o_i, x^o_j) + w^c_p \sum_{\langle i,j \rangle \in \mathcal{N}} E^c_p(x^c_i, x^c_j). \tag{6.1}$$

where V denotes the set of voxels, B denotes the set of background pixels, X denotes the set of voxels and background pixels, i.e., $X = \{V, B\}$, X_r denotes the set of voxels and background pixels passed by the ray r. x^o_k (and x^o_i, x^o_j) denotes the occupancy of both voxels and pixels. And the neighborhood $\mathcal{N}$ is defined as 6-neighborhood for voxel and 4-neighborhood for background pixel.

As shown in Fig. 6.10, a ray passes through a series of voxels and finally hits the background. Since the background pixels and the volume voxels share the same attributes (i.e., occupancy and color), they can be treated equally. In this way, the ray tracing equation and ray message passing can be extended naturally, by having the background pixel on the ray serving as the last voxel.

The intersection between the ray and infinite sphere. The intersection is only dependent on the ray direction. Similar to the ray-voxel intersection, there are two models.

Ray Line Model In this model, the ray is treated as a line. So its intersection with the background infinite sphere is a point. And the pixel containing the point is taken as the intersected pixel. The intersection of the ray with the infinite sphere can be computed as a ray from the origin and a sphere with center at the origin and radius 1.

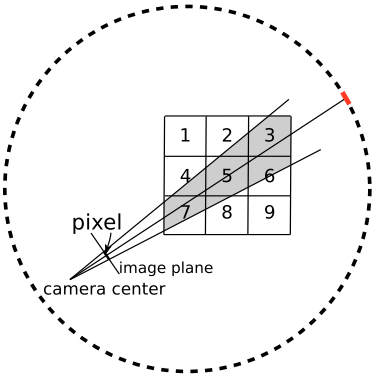
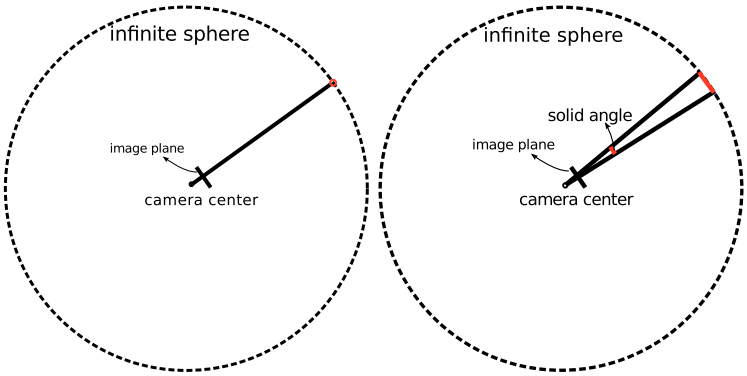


Figure 6.8: Ray-volume-background intersection relationship



(a) ray line model (b) ray cone model

Figure 6.9: Ray-background intersection model

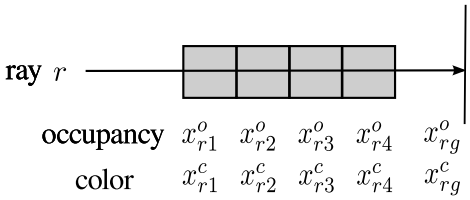


Figure 6.10: A ray passes through a line of voxels, and finally hits the background pixel (x_{rg}).

Ray Cone Model In this model, the ray is treated as a cone. The intersection of the ray cone with the infinite sphere can be computed as a ray cone with solid angle θ centered at the origin and a sphere with center at the origin and radius 1.

6.1.4 Related Work

In [30], Crispell introduced a new term vis_∞ to model "the probability of the ray passing unoccluded through the model", which can be thought of as a "background" appearance model. Equivalently, each ray hit an independent background pixel at the end, besides the voxels it passes through. This work follows the same idea of modeling not just the objects in the bounding box, but also the objects outside of the bounding box. Different from [30], in this work, a spherical panoramic background is used as the background model, where different rays can hit the same pixel on the background panoramic image and the background pixels are regularized to model color regularities. In this work, the selection of the bounding box based on the camera configurations is also studied.

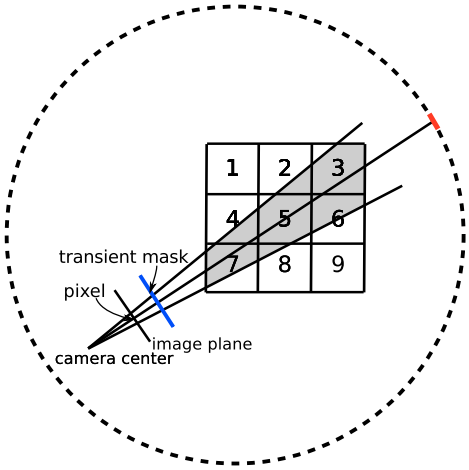


Figure 6.11: Ray-mask-volume-background intersection relationship

6.2 Handling Transient Clutter

In the above study, it is assumed that the scene is static, surfaces are Lambertian and the volume is large enough to explain all pixel observations (together with the spherical background). But the world is not as ideal as (and also not as boring as!) assumed. The real-world scenes are full of dynamic objects, lots of surfaces are reflective, and the spatial span of the scene captured in the image could be very large. All these are factors that need to be taken into consideration if a reconstruction system with high quality and strong robustness is desired. Here the transient clutter is defined as the observations that cannot be modeled and explained by the volume and background. The dynamic objects, non-Lambertian material, and out-of-volume objects are considered as transient clutter.

6.2.1 Transient Clutter Modeling and Estimation

One simple idea of removing the transient clutter is to create a mask to remove the bad pixels. But how to get the mask? Instead of manually creating the mask, here the previous model is extended to automatically estimate the mask. As shown in Fig. 6.11 and Fig. 6.12, a mask with unknown mask values is put right in front of the camera. The viewing ray from each pixel on the image will first hit the corresponding pixel on the mask, and then some voxels in the volume, and finally hit the background. Because the mask is to model the transient clutter, here it is called transient mask.

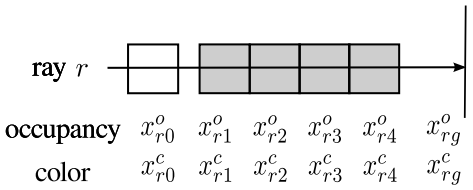


Figure 6.12: A ray passes through the transient mask, a line of voxels and finally hits the spherical background

Here the model in Eq. (6.1) is further extended to include the contribution of the transient pixels.

$$E(X) = \sum_{r\in R} E_r(X_r) + w^o_{uv} \sum_{k\in V} E^o_{uv}(x^o_k) + w^o_{ub} \sum_{k\in B} E^o_{ub}(x^o_k) + w^o_{ut} \sum_{k\in T} E^o_{ut}(x^o_k)$$
$$+ w^o_p \sum_{\langle i,j\rangle\in\mathcal{N}} E^o_p(x^o_i, x^o_j) + w^c_p \sum_{\langle i,j\rangle\in\mathcal{N}} E^c_p(x^c_i, x^c_j) \tag{6.2}$$

where T denotes the set of transient pixels, $X = \{V, B, T\}$, the neighborhood $\mathcal{N}$ is extended similarly to include the 4-neighborhood in the transient mask.

Because in the ray clique the transient pixels (on the mask) are treated similarly as the voxels and background pixels, the same deep belief propagation algorithm can be easily extended to handle them.

6.2.2 Related Work

There are similar works on handling transient outliers in the feature-based approach. Basically it is easy for the feature-based approach to handle the problem, because the transient clutter pixels can not be matched in high confidence across images. The clutter pixels can be discarded through thresholding the matching confidence [43, 54].

Chapter 7

Experiments and Evaluation

The previous chapters have discussed the Ray MRF formulation of the inverse ray tracing problem, and an efficient algorithm for estimating the volumetric 3-d model and spherical background. How well does the proposed solution work? What are the advantages and weaknesses of the proposed approach compared with other approaches? To answer these questions, the algorithm is first evaluated on the Middlebury benchmark datasets, from which quantitative evaluations with ground truth geometry provided by laser scanner are analyzed. To evaluate the performance and robustness of the algorithm under more general working conditions, experiments on less-controlled datasets are provided. And the comparison with one of the state-of-the-art algorithms, PMVS2 [48] + PoissonRecon [65], is provided. At last, the algorithm is compared with one of the state-of-the-art volumetric reconstruction algoirthms, the voxel-world-model algorithm [30], in an image-based rendering experiment.

7.1 Dino Sparse Ring Dataset (dinoSR16)

The dinoSR16 dataset consists of 16 views captured in a circle around a plaster dinosaur (stegosaurus) [116], as shown in Fig. 7.2. The dinosaur object is of size 7cm x 9cm x 7cm. The images, with resolution 640x480, were carefully captured using the Stanford Spherical Gantry which enables moving a camera on a sphere to specified latitude/longitude angles to an accuracy of approximately 0.01 degrees (see Fig. 7.1) [117]. The cameras are calibrated using Jean-Yves Bouguet's Matlab toolbox and their own software with a planar grid from 68 viewpoints. The ground truth is captured with a Cyberware Model 15 laser scanner. The resolution of the ground truth model is 0.25mm.

Fig. 7.4 shows two different views of the reconstructed 3-d model rendered in textured and non-textured modes. It can be seen that the textureless object (with some fixed shadow

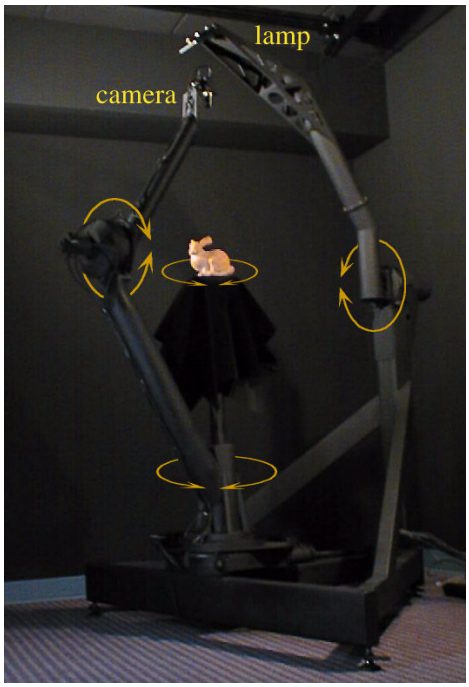


Figure 7.1: Stanford Spherical Gantry image capture and camera calibration system. `http://graphics.stanford.edu/projects/gantry/`

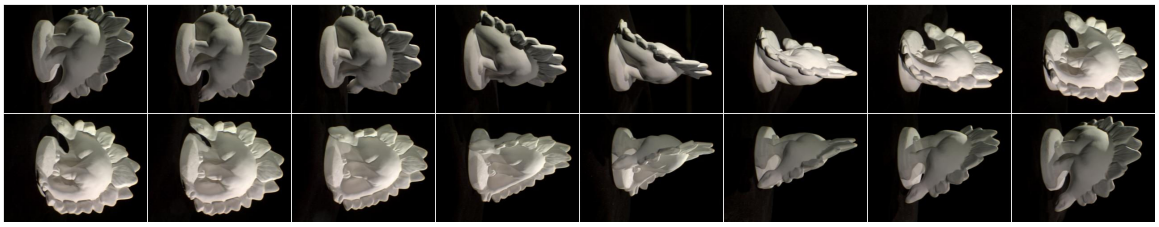


Figure 7.2: dinoSR16: thumbnails of all 16 images in the dataset

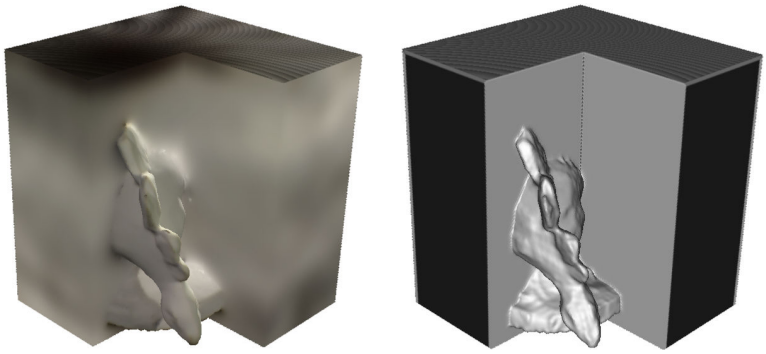


Figure 7.3: dinoSR16: 1/4 cut of the occupancy-color volume: colored and non-colored model

Figure 7.4: dinoSR16: two different views of the textured and non-textured reconstruction models generated by IRAY (the proposed approach)

on the body under the constant illumination condition) is reconstructed in fairly high quality in both geometry and appearance. The textured rendering of the object is hardly distinguishable from the real object. Fig. 7.3 shows the reconstructed 3-d occupancy-color volume, one quarter of which is cut off to show the estimated color of the empty voxels. Quantative comparision of the reconstruction accuracy and completeness between PMVS2, voxel world model, and IRAY (the proposed approach) is shown in Tab. 7.1. In the quantative evaluation, an accuracy threshold of 90% is used, i.e., an accuracy of 1.0mm means that 90% of the points are within 1.0mm of the ground-truth model. For completeness, an inlier threshold of 1.25mm, i.e., a completeness of 95% means that 95% of the points are within 1.25mm of the ground-truth model. From Tab. 7.1, IRAY achieves sub-millimeter accuracy (0.63mm), far better than the geometry accuracy achieved by the voxel-world-model algorithm (2.61mm), while falling behind the best accuracy (0.37mm) achieved by PMVS2+PoissonRecon. The completeness of the reconstructed surface by IRAY is close to PMVS2+PoissonRecon, better than the voxel-world-model algorithm. In summary, the geometric accurac of the proposed algorithm is good, but still have room for improvement to match the performance of the best ones.

Algorithm	Accuracy at 90% (mm)	Completeness (%)
PMVS2+PoissonRecon[48]	0.37	99.2
IRAY	0.63	96.7
voxel world model (with 40 images) [30]	2.61	91.4

Table 7.1: Accuracy and completeness comparison between PMVS2, IRAY and voxel world model for the dinoSR16 dataset. Notice that the voxel world algorithm has only been evaluated on the dinoR (40 images) datasets. Since the dinoSR dataset is a subset of dinoR dataset, it is reasonable to assume that the performance of voxel world model will not be better than the performance reported here.

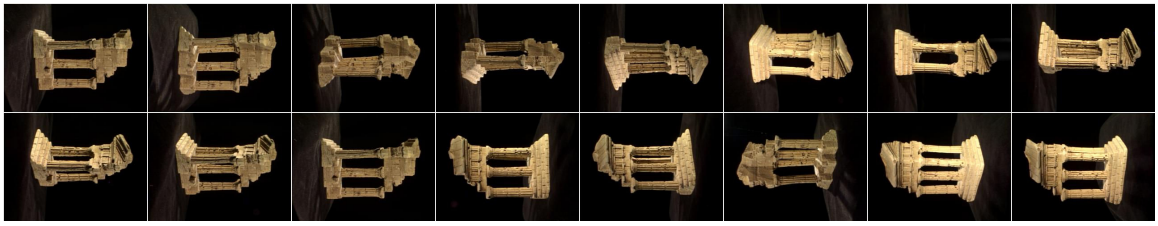


Figure 7.5: templeSR16: thumbnails of all 16 images in the dataset

7.2 Temple Sparse Ring Dataset (templeSR16)

The templeSR16 dataset consists of 16 views sampled on a ring around a plaster reproduction of a temple. The temple object is of size $10cm \times 16cm \times 8cm$. The images and ground-truth 3-d models are captured similarly as the dinoSR16 dataset. Fig. 7.7 shows two different views of the reconstructed 3-d model in textured and non-textured modes. The object is reconstructed in fairly high quality in both geometry and apprance. The textured rendering of the object is almost indistinguishable from the real object. Fig. 7.6 shows the reconstructed occupancy-color volume, where one quarter of the volume is cut off. Quantative comparision of the reconstruction accuracy and completeness between PMVS2, voxel world model, and IRAY (the proposed approach) is shown in Tab. 7.2. The characteristics of the comparision between the three methods are similar as the ones in dinoSR16 dataset.

7.3 Elephants Dataset (elephants40)

From the above two Middlebury experiments, it can seen that the objects can be reconstructed in high quality, given accurate calibration and carefully controlled capturing condition. However under normal working conditions, the expensive equipment and complex

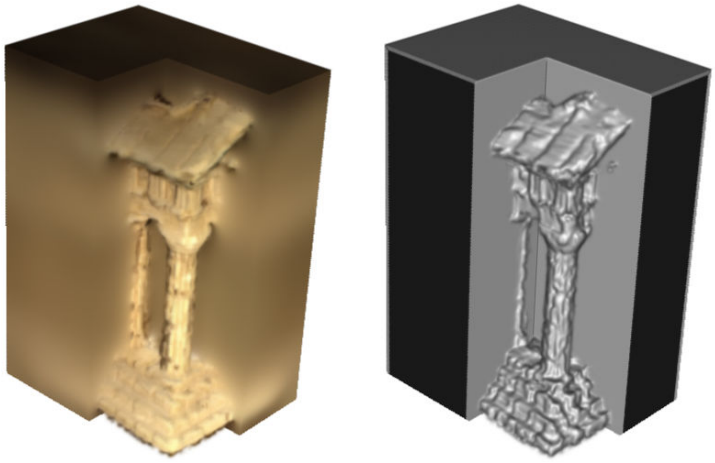


Figure 7.6: templeSR16: 1/4 cut of the occupancy-color volume: colored and non-colored model

Figure 7.7: templeSR16: two different views of the textured and non-textured reconstruction models of IRAY (the proposed approach)

Algorithm	Accuracy at 90% (mm)	Completeness (%)
PMVS2+PoissonRecon [48]	0.63	99.3
IRAY	1.20	95.1
voxel world model (with 40 images) [30]	1.89	92.1

Table 7.2: Accuracy and completeness comparison between PMVS2, IRAY and voxel world model for the templeSR16 dataset. Notice that the voxel world model algorithm has only been evaluated on the templeR (40 images) datasets. Since the templeSR dataset is a subset of templeR dataset, it is reasonable to assume that the performance of voxel world model will not be better than the performance reported here.

setup procedure used in the controlled environment are not affordable and even not possible, for example the out-door object capture scenario. And in addition to the geometric accuracy, there are other aspects to be evaluated, such as algorithms' performance in less controlled environment, the completeness of the reconstruction in normal conditions, and the ability to handle objects of complex topology, etc. The following experiments focus on evaluating the proposed algorithm's performance under much less controlled environments, for example, objects of more complex geometry, more complex backgrounds, less accurate camera calibration, out-door scenes, etc. The performance of the algorithm will be compared with the PMVS2+PoissonRecon, because 1) PMVS2 is one of the best MVS algorithms and its implementation is publicly available; 2) the combination of PMVS2 and PoissonRecon is a commonly used pack of algorithms, that has ranked top in several standard evaluations [6]. In the following experiments, all the parameters used for PMVS2 and PoissonRecon are chosen as suggested in [6], to create a dense point cloud and a detailed surface reconstruction.

The elephants dataset consists of 40 images taken around two carved elephants, one made of stone and the other one made of wood. Fig. 7.8 shows thumbnail views of the entire dataset. Inside each carved elephant's body, there is a smaller baby elephant. Fig. 7.9 better shows it with a large image. These two elephants are put on a round table, and a hand-held camera (Canon EOS Rebel XSi Digital SLR) moves around the table from the top. Fig. 7.10 shows the camera positions. The elephants are about 7cm in height. The images are captured with normal indoor lighting, auto-focused and fixed exposure settings (including aperture, shutter speed and ISO-value). The original resolution of each captured image is 4272x2848, which is used for the camera calibration with Bundler software. Then the images are resized to 15% , i.e., 641x427, for 3-d reconstruction (both PMVS2 and IRAY).

For comparison with IRAY (the proposed algorithm), PMVS2 software is used to generate a dense point cloud first. In PMVS2, the default parameters are set to work with high-resolution images. In the experiments, all the parameters used for PMVS2 and PoissonRecon are chosen as suggested in [48], except that two of the default parameters are modified to generate denser point clouds, given the relatively low resolutions of the input images: 'level' is changed from 1 to 0, and 'csize' is changed from 2 to 1, to create denser point cloud than the default parameters [1]. We do the same change on the default parameters

[1] PMVS2 "internally builds an image pyramid," and the 'level' parameter "pecifies the level in the image pyramid that is used for the computation. When level is 0, original (full) resolution images are used. When level is 1, images are halved (or 1/4 the number of pixels)." "csize (cell size) controls the density of reconstructions. The software tries to reconstruct at least one patch in every csize x csize pixel square

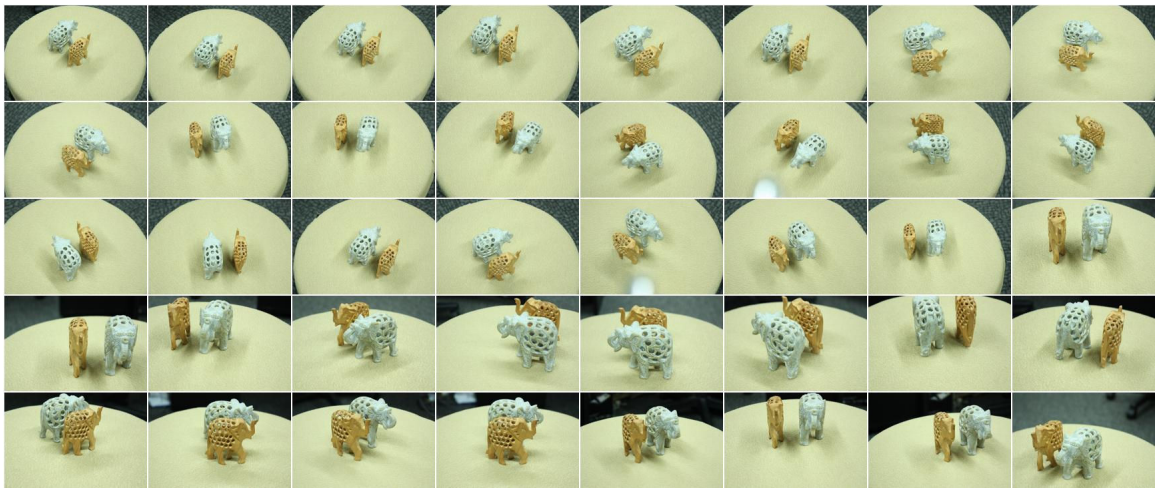


Figure 7.8: elephant40: thumbnails of all 40 images in the dataset

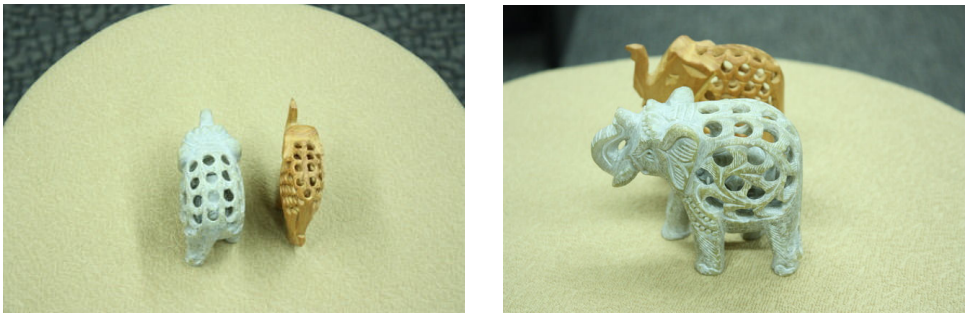


Figure 7.9: elephant40: two sample images

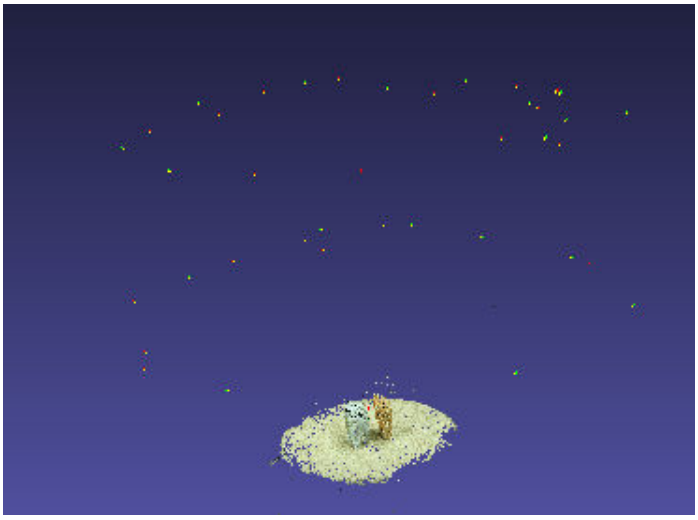


Figure 7.10: elephants40: visualization of camera poses and sparse point cloud of elephants40 dataset generated by Bundler software (each camera is represented by two color dots)

of PMVS2 for all the following experiments on different datasets.

Fig. 7.11 shows the dense oriented (each point is associated with an orientation normal) point cloud generated by PMVS2 from four different view angles. The sides of the elephants are reconstructed densely, while the rear part is almost completely missing, probably due to smaller number of visible views.

To get a surface reconstruction from the oriented point cloud generated by PMVS2, a surface reconstruction software is needed. In [48], Furukawa and Ponce used PoissonRecon software developed by Kazhdan and Bolitho [65, 64]. The same software is used to generate a mesh reconstruction [2], as shown in Fig. 7.12. This figure shows textured and non-textured 3-d surface models from different view angles. PoissonRecon does a good job of filling the rear part of the elephants. However, it incorrectly fills in the carved holes on the body of the elephants.

Here another popular surface reconstruction algorithm, ball pivoting [11], is tried, which functionality is provided by the Meshlab software [23]. The algorithm runs with the default parameter as suggested in Meshlab. Fig. 7.13 shows the reconstructed surface from different views. It can be seen that ball pivoting builds a surface conservatively — just connecting

region in all the target images specified by 'timages'. Therefore, increasing the value of 'csize' leads to sparser reconstructions." [45]

[2]The PoissonRecon runs with the following parameters as suggested in [48]: octree depth = 10, solver divide = 8, samples per node = 1, surface offsetting = 1.

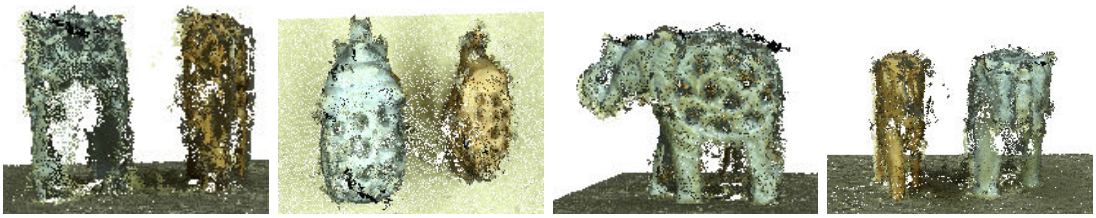


Figure 7.11: elephant40: PMVS2's oriented point cloud outputs

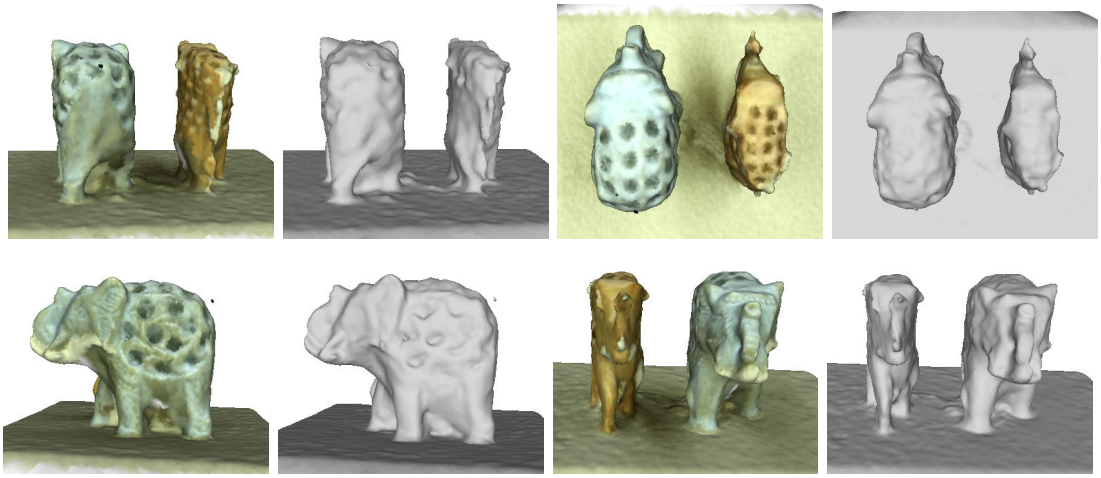


Figure 7.12: elephants40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon

existing points instead of doing interpolation aggressively as PoissonRecon. The resulting mesh leaves the whole rear part with a hole, while preserving the holes on the body. With these two surface reconstruction algorithms, it can be seen that surface reconstruction from an irregularly distributed point cloud generated from multi-view stereo is hard due to the loss of information in the point cloud representation.

Fig. 7.14 shows the reconstruction result from the proposed algorithm. The volume resolution in the experiment is 371x293x93 (about 10M voxels). By comparing Fig 7.12, 7.13 and 7.14, it can be seen that the proposed approach can preserve the carved holes correctly. The rear parts of the elephants are reconstructed reasonably. This is because the proposed algorithm better handles the occlusion relationship and is capable of reconstructing a region with a small number of views. Also the volumetric representation, as used in IRAY, contains more geometric information than unconnected points, which makes the surface reconstruction much easier when using a simple iso-thresholding algorithm (e.g., marching cube algorithm [88]) compared with the point cloud based approach.

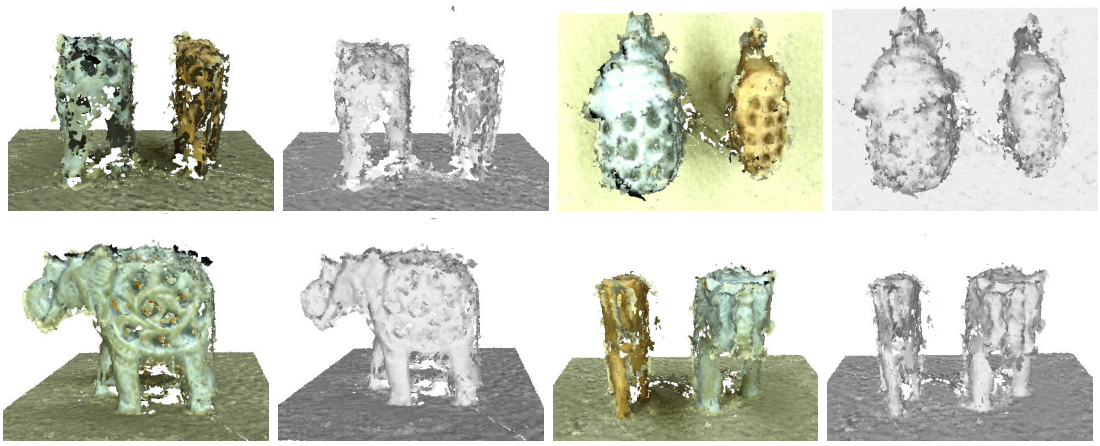


Figure 7.13: elephant40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + BallPivoting

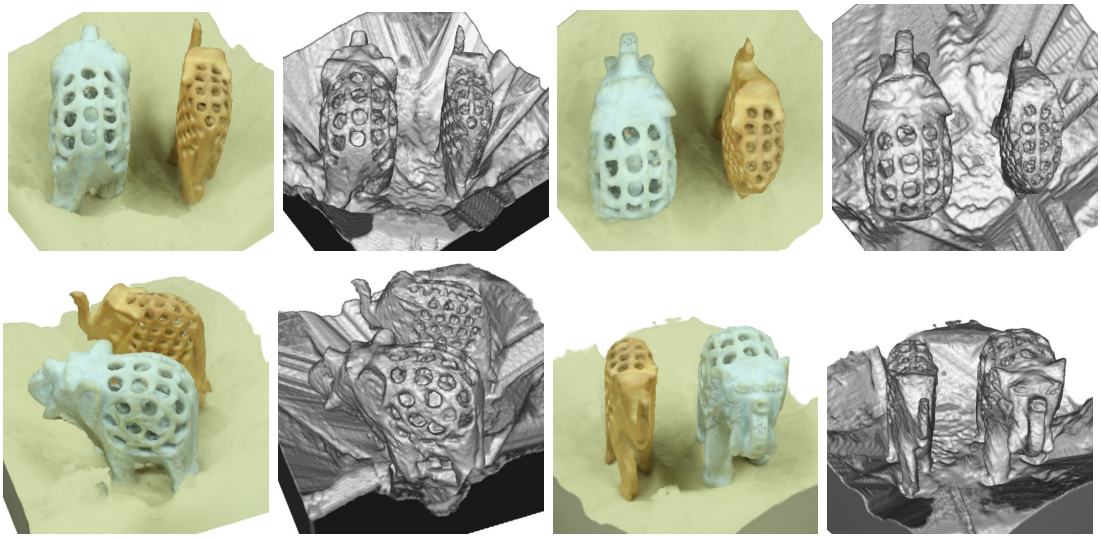


Figure 7.14: elephant40: four different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm)

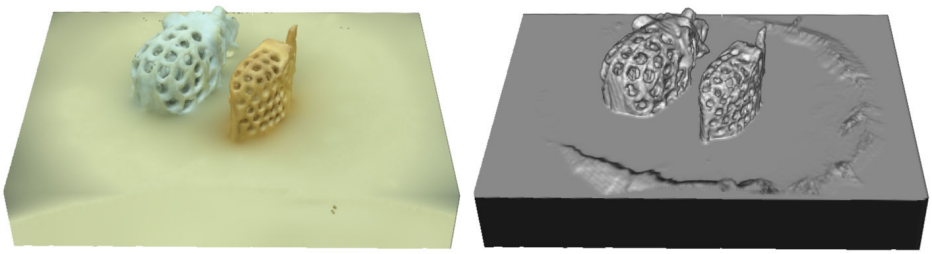


Figure 7.15: elephants40: 1/2 cut of the reconstructed occupancy-color volume

Figure 7.16: building14: thumbnails of all 14 images in the dataset

7.4 Building Dataset (building14)

The building14 dataset consists of 14 images of a building cluttered by trees and moving vehicles. The three-story building is the home of the Division of Applied Mathematics, Brown University. The images are captured with a hand-held camera (Canon EOS Rebel XSi Digital SLR Camera) with auto-focus and fixed exposure setting under normal outdoor lighting condition. The original image resolution is 4272x2848. Cameras are auto-calibrated with the Bundler software with the original image solution. Fig. 7.16 shows thumbnail views of all 14 images in the dataset. The original images are resized to 15% of their original size, i.e., 641x427, for the 3-d reconstruction. Fig. 7.17 shows an example of the resized image. Fig. 7.18 shows the camera positions and a sparse point cloud of the scene generated by Bundler software.

Fig. 7.19 shows different views of the dense point cloud generated by the PMVS2 software. It can be seen that most parts of the building are reconstructed densely, while the most of tree and grass parts of the scene are missing and the roof and chimney parts of the building are also missing. The absence of the trees and grass is very likely because of the homogeneity of the regions, which makes the pixel-wise matching problematic. The missing of the roof and chimney is likely due to self occlusions and few number of visible views. The point cloud generated by PMVS2 is further fed into the PoissonRecon software to generate a mesh surface representation. Fig. 7.20 shows three different views of the reconstructed surface model.

Fig. 7.21 shows the reconstruction result from IRAY. The volume resolution is 186x371x146 (about 10M voxels). Compared with the PMVS2+PoissonRecon result, it can be seen that IRAY generates a more complete reconstruction of the scene, including the trees, grass, roof, chimney, etc. The textured rendering of the reconstructed 3-d model is photo-realistic, given the limited volume resolution (186x371x146) used.

Figure 7.17: building14: two sample images

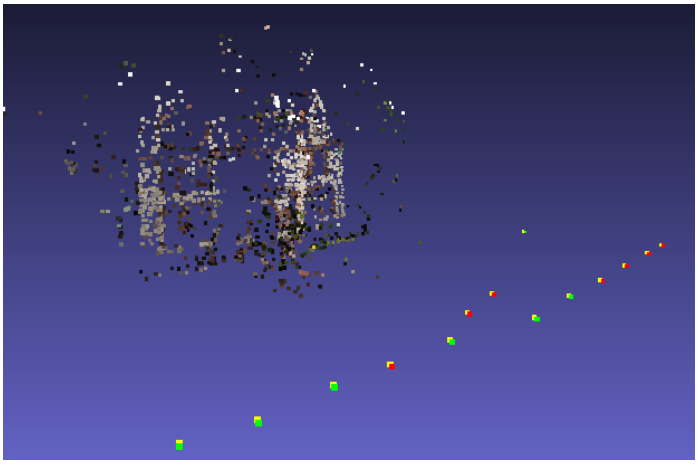


Figure 7.18: building14: visualization of camera poses and sparse point cloud generated by the Bundler software (each camera is represented by two close color dots)

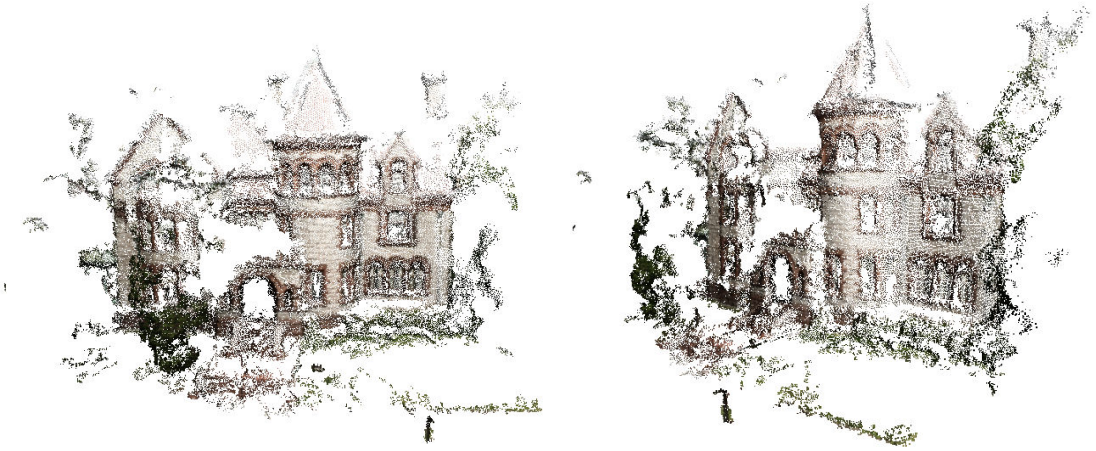


Figure 7.19: building14: PMVS2's oriented point cloud outputs

Figure 7.20: building14: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon

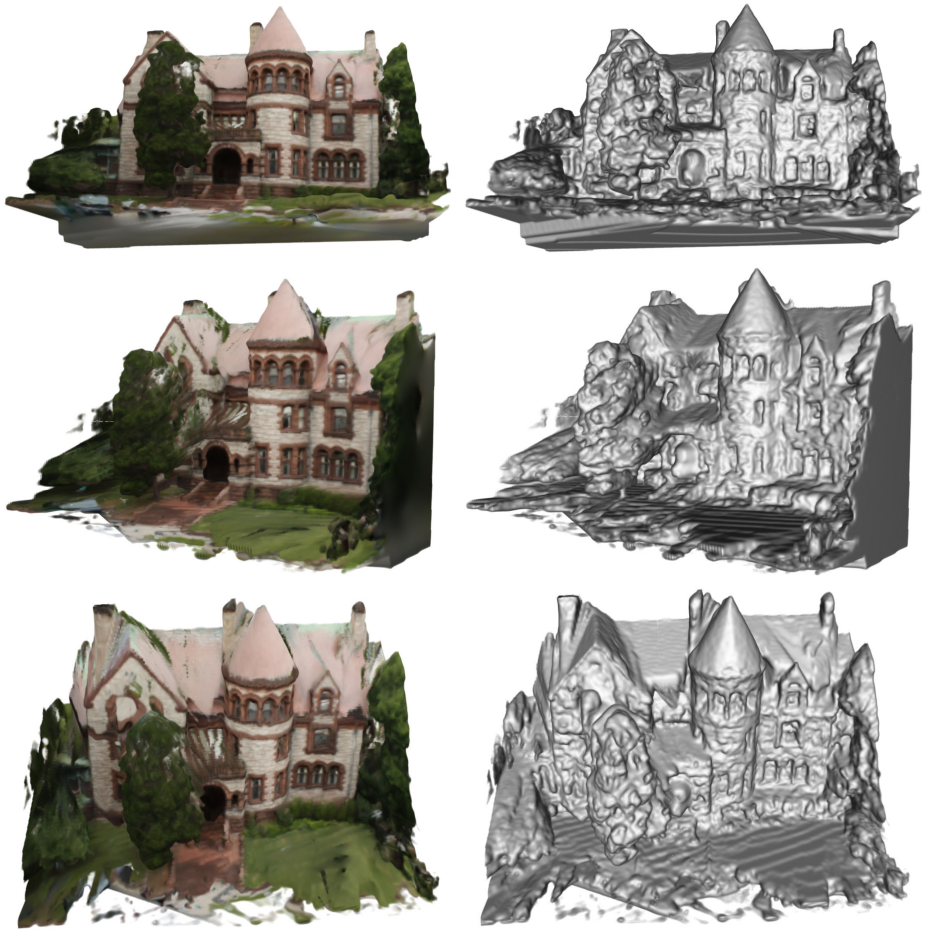


Figure 7.21: building14: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by from IRAY (the proposed algorithm)

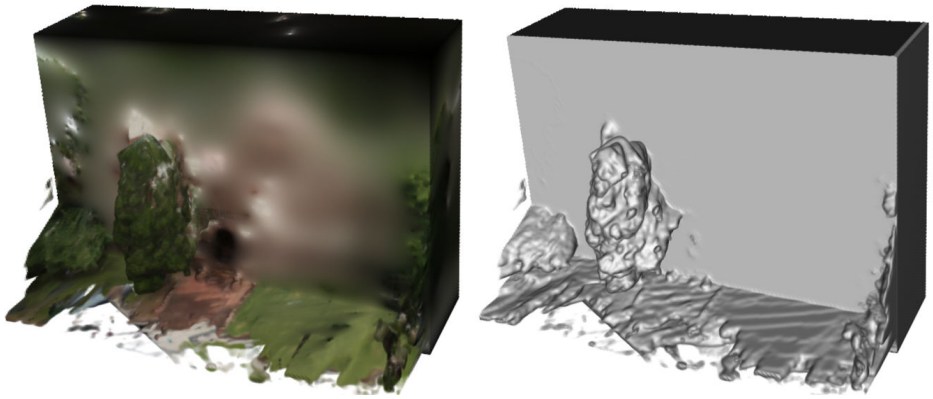
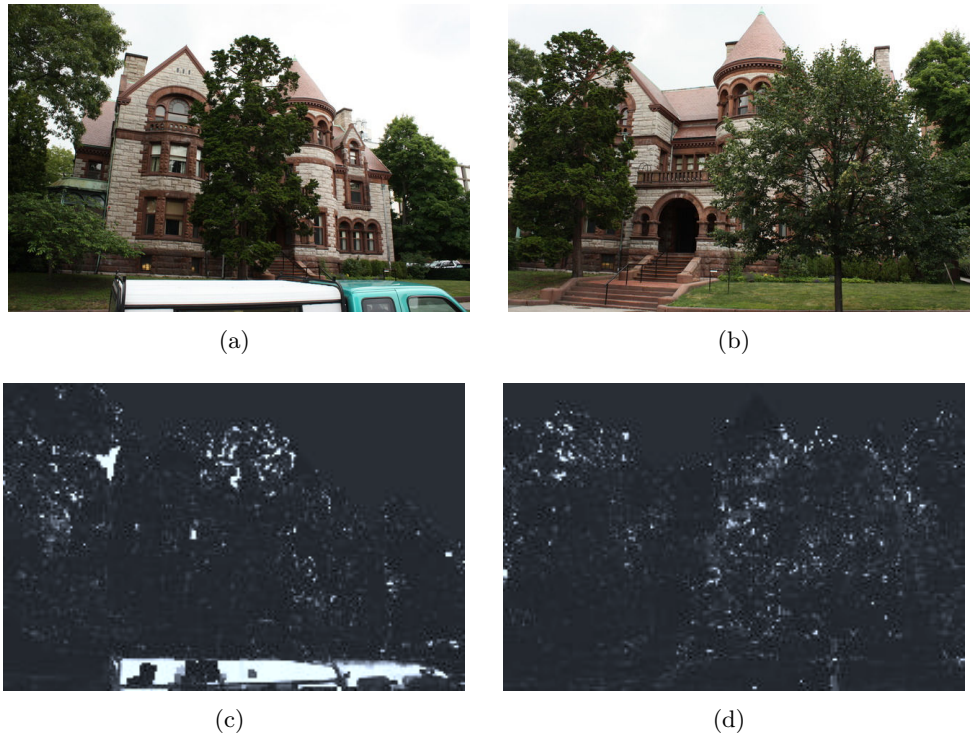


Figure 7.22: building14: 1/2 cut of the reconstructed occupancy-color volume



(a) (b)

(c) (d)

Figure 7.23: building14: two example transient masks generated automatically with IRAY. Top: two sample input images (the same two views as shown in Fig. 7.17); bottom: the corresponding transient masks generated by the proposed algorithm, where whiter pixels mean transient observations.

7.5 Horse Dataset (horse29)

The horse29 dataset is composed of 29 images, captured in a rough circle around a 6-meter tall horse statue at Brown campus. During capture, the camera is set to auto-focus and fixed exposure, and the camera points upwards to capture the upper region of the statue. Fig. 7.24 shows thumbnail views of the whole image dataset, and two larger sample images are shown in Fig. 7.25. The location of the cameras can be seen in Fig. 7.26, which also shows the sparse point cloud generated with the Bundler software, using the images with original resolution of 4272x2848. The image dataset used for 3-d reconstruction is resized to 15%, i.e., 641x427. Both PMVS2 and IRAY are tested on the resized image dataset (with resolution 641x427).

Fig. 7.27 shows different views of the dense point cloud generated by PMVS2. The body part of the horse is reconstructed densely, while the tops of the horse and the horse-rider and the base of the statue are missing because these regions are not observed in any views from the bottom of the statue.

Fig. 7.28 shows the surface reconstruction by PoissonRecon, taking PMVS2's point cloud as input. It can be seen that the generated surface totally departs from the true shape. This is very likely because the missing regions of the point cloud are too large, and the PoissonRecon tries to generate a closed surface. The ball-pivoting algorithm is then tried on the same point cloud. The generated surface is shown in Fig. 7.29. As shown in this figure, the ball-pivoting algorithm generates a surface in a less aggressive manner compared with PoissonRecon, producing less outliers while leaving the surface incomplete and noisy.

Fig. 7.30 shows the reconstruction result from IRAY. The volume resolution is 225x299x150 (about 10M voxels). Compared with PMVS2+PoissonRecon and PMVS2+BallPivoting, it can be observed that IRAY does a much better job of recovering the geometry of the horse, horse-rider and the base of the statue. It has successfully reconstructed the tops of the objects, which are non-visible in all views. It is possibly because although these regions are not visible, they are confined by the visibility constraints, and combined with the geometry smoothness prior, IRAY outputs a reasonable reconstruction of these regions. On the contrary, the visibility information is totally lost in the point cloud representation, which could lead to final surfaces violating the visibility constraints.

Figure 7.24: horse29: thumbnails of all 29 images in the dataset

Figure 7.25: horse29: two example images

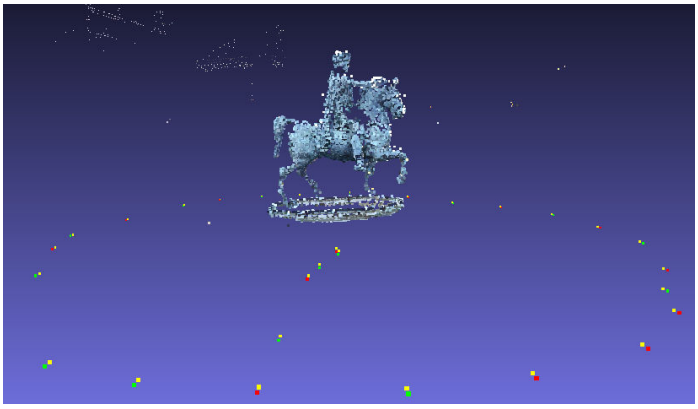


Figure 7.26: horse29: visualization of the camera poses and sparse point cloud generated by Bundler software (each camera is represented by two close color dots)

Figure 7.27: horse29: PMVS2's oriented point cloud outputs

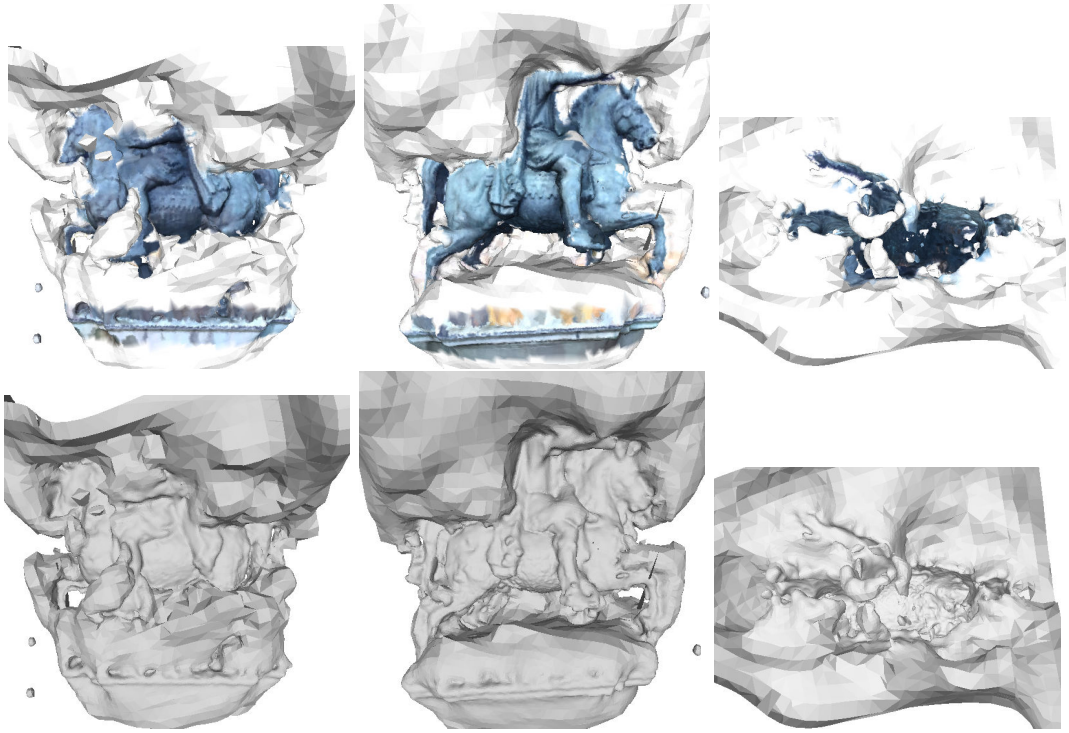


Figure 7.28: horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + PoissonRecon

Figure 7.29: horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by PMVS2 + BallPivoting

Figure 7.30: horse29: three different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm)

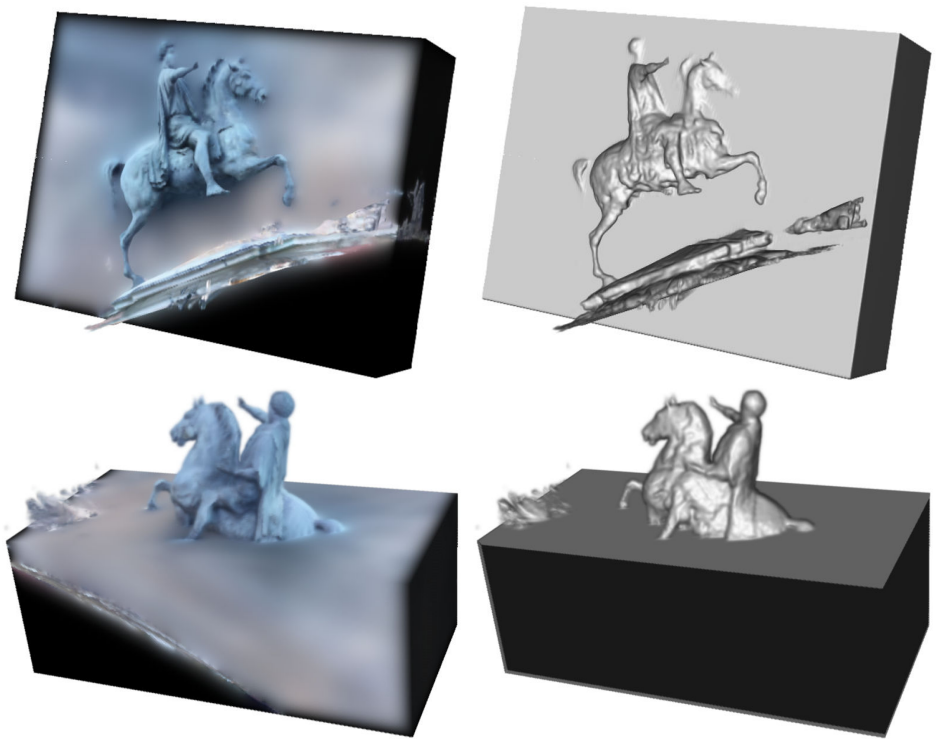


Figure 7.31: horse29: different 1/2 cuts of the reconstructed occupancy-color volume

7.6 Flower Dataset (flower31)

Fig. 7.32 shows thumbnail views of the 31 images of the dataset. The dataset is captured by a hand-held camera (Canon EOS Rebel XSi Digital SLR) shooting a pot of flower on a black table from the top. During capture, the camera is auto-focused, and the exposure setting is fixed. The original size of the image is 4272x2848. The original images are used for camera calibration with the Bundler software. Then the images are resized to 641x427 for 3-d reconstruction (both IRAY and PMVS2). Fig. 7.33 shows an example of the resized image in the dataset. Fig. 7.34 shows the camera positions and sparse point cloud of the dataset generated by the Bundler software.

In the experiment, the volume resolution is 247x247x165 (about 10M voxels). Fig. 7.35 shows the 3-d model generated by IRAY from different views. The textured model looks fairly good, even though the object is non-Lambertian, violating the Lambertian assumption. The non-textured rendering reveals that recovered geometry is not very accurate. To improve the geometry accuracy, more complex material properties need to be modeled. However, the result of the current quality could still be good enough to serve many applications, for instance, virtual reality, where the textured rendering quality is more important

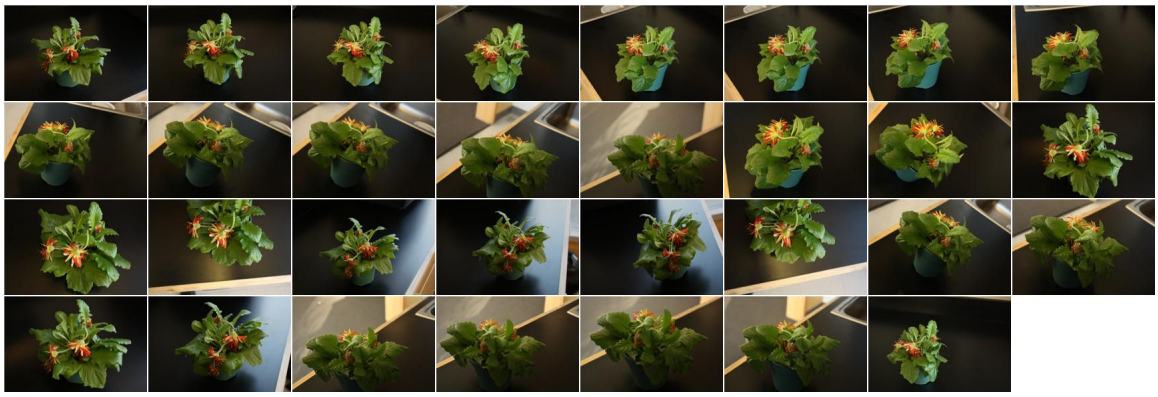


Figure 7.32: flower31: thumbnail views of all 31 images in the dataset

Figure 7.33: flower31: two example images

than geometric accuracy.

7.7 Capitol Dataset (capitol40)

In the above experiments, the accuracy and completeness of the the proposed algorithm's recovered geometry and topology have been evaluated and compared with the state-of-the-art methods. Here the focus turns to the image-based rendering performance. The proposed algorithm will be compared with one of the state-of-the-art methods — voxel world model (VWM) algorithm [106, 105, 30], on the photo-realistic quality of their rendered novel views.

In this experiment, the capitol40 dataset is used. The dataset is extracted from an aerial video sequence, captured by D. Crispell on a helicopter hovering over the Rhode Island Capitol building. This video sequence has been extensively used in the voxel world model study [106, 105, 30]. In this experiment, 49 frames are extracted from the video sequence. Among them, 40 frames are used for training (i.e., reconstructing) the volumetric 3-d model, and the remaining 9 frames are used for testing. Fig. 7.36 shows the thumnail

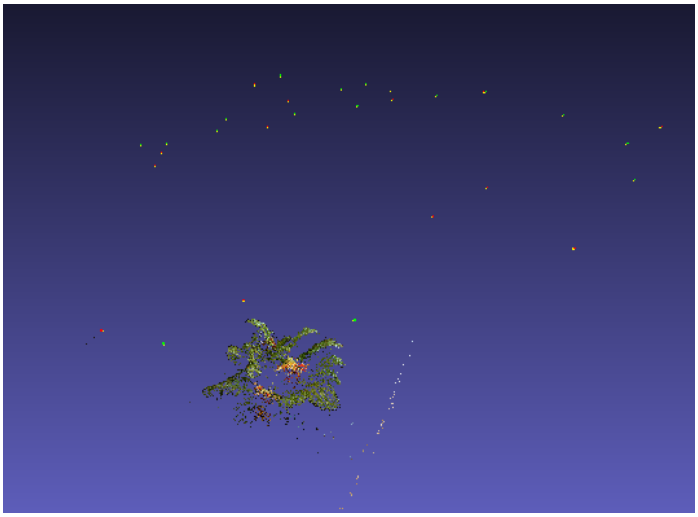


Figure 7.34: flower31: visualization of the camera positions and sparse point cloud generated by the Bundler software (each camera is represented by two color dots)

Figure 7.35: flower31: two different views of the textured and non-textured renderings of the reconstructed 3-d model generated by IRAY (the proposed algorithm)

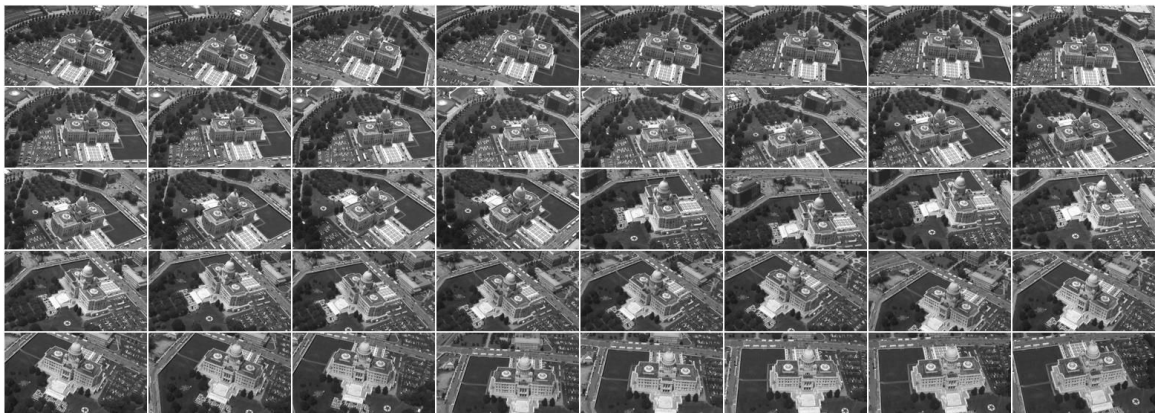


Figure 7.36: captiol40: thumbnail views of the 40 training images in the dataset

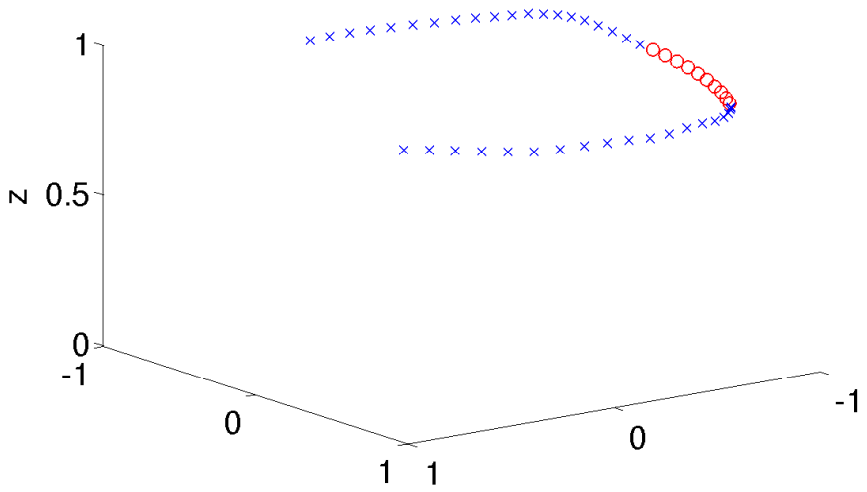
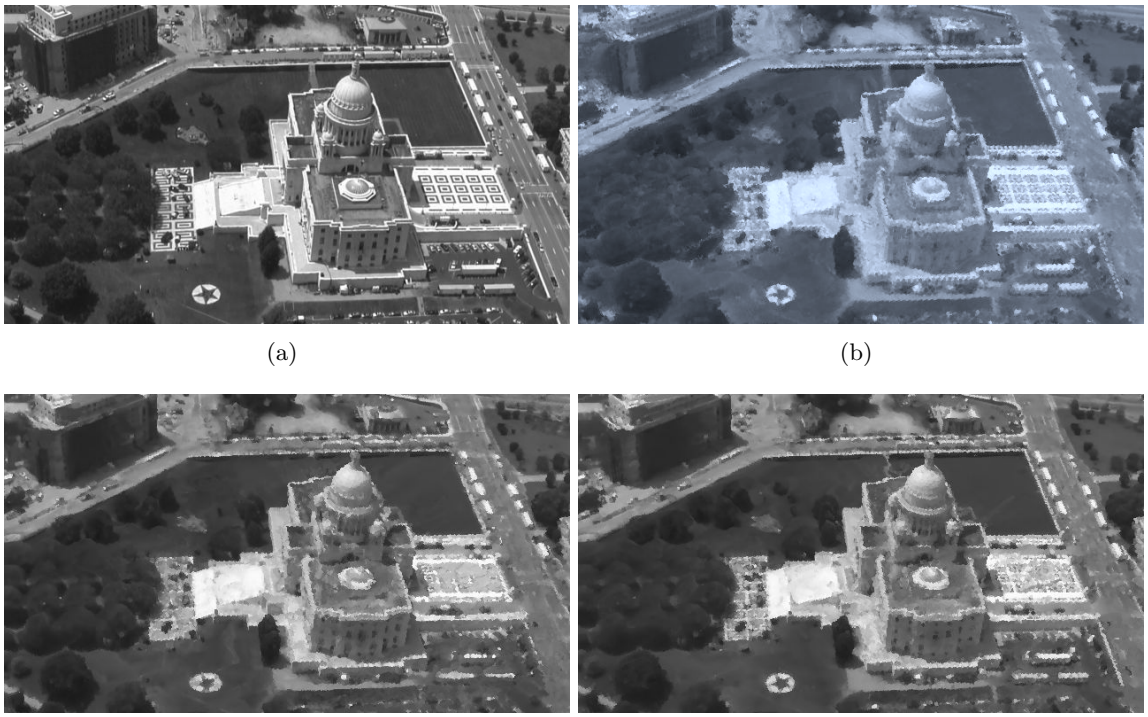


Figure 7.37: capitol40: visualization of the camera locations of the 40 training images (in blue cross) and 9 test images (in red circle). The ground is approximately on the xy plane with z = 0.

views of the 40 training images; Fig. 7.37 shows the camera locations of the 40 training images (in blue) and the camera locations of the 9 training images (in red).

Because both VWM and IRAY reconstruct a volumetric model directly and can render synthetic images at a novel view point, these two approaches' performance on novel view synthesis are compared by computing the synthetic images's difference with the ground-truth images. In this experiment, these two approaches use the same bounding box and volume resolution (432x432x96). One difference between IRAY and VWM is the existence of the regularization (aka, prior) terms. To better compare these two approaches, the results are given for both the regularized version of IRAY and the unregularized version. The regularized version of IRAY uses the same parameters as the previous experiements. The unregularized version of IRAY sets the weights of the prior terms to be zero, to better compare with VWM. Fig. 7.38 shows the synthetic image comparison for one of the test cameras. In the figure, the ground-truth test image, VWM's rendered image, IRAY's rendered image, and the rendered image from IRAY with the prior terms disabled are shown. It can be seen that IRAY's rendered image is sharper than VWM's result, especially on object boundaries, such as the building boundary, window, bush, etc. (Also see the zoomed-in version in Fig. 7.40.) This is further verified by the quantitative comparison result as shown in Fig. 7.39. Here the mean absolute error between the synthetic image and the ground



(a) (b)

(c) (d)

Figure 7.38: capitol40: visual comparison between (a) test image (ground truth), (b) rendered image from the voxel world model, (c) rendered image from IRAY, and (d) rendered image from IRAY with prior terms (pairwise smoothing terms and baloon terms) disabled.

truth image is computed. It can be seen that IRAY's synethetic image has lower errors than VWM's result for all 9 test views, and on average 3–3.5 graylevel better. Equivalently, the mean absolute error is reduced by 19–22% compared with VWM. To elleviate the influence by the fact that the two approaches handles the bounding box boundary slightly differently, another comparison is performed on the central cropped region of the image, as shown in Fig. 7.40 and 7.41. It can be seens that the performance is consistent with the previous result.

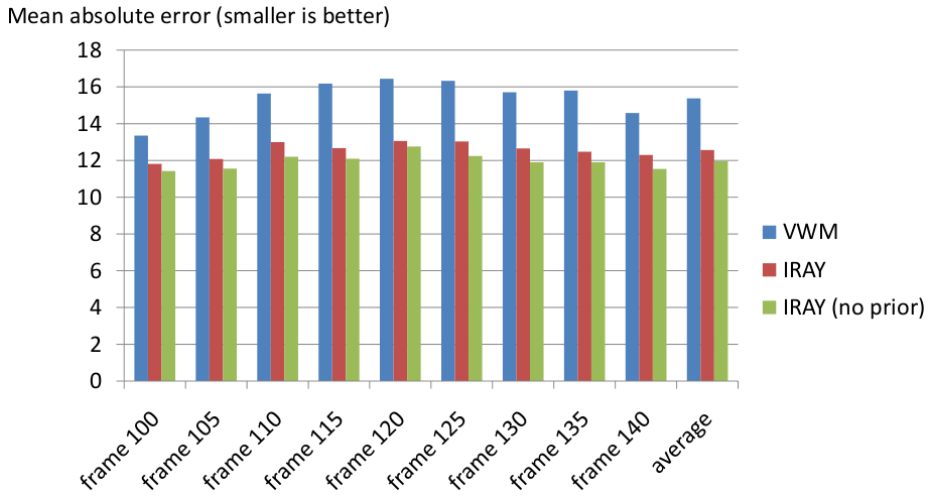


Figure 7.39: captiol40: mean absolute error of the synthetic images for both VWM, IRAY and IRAY without prior

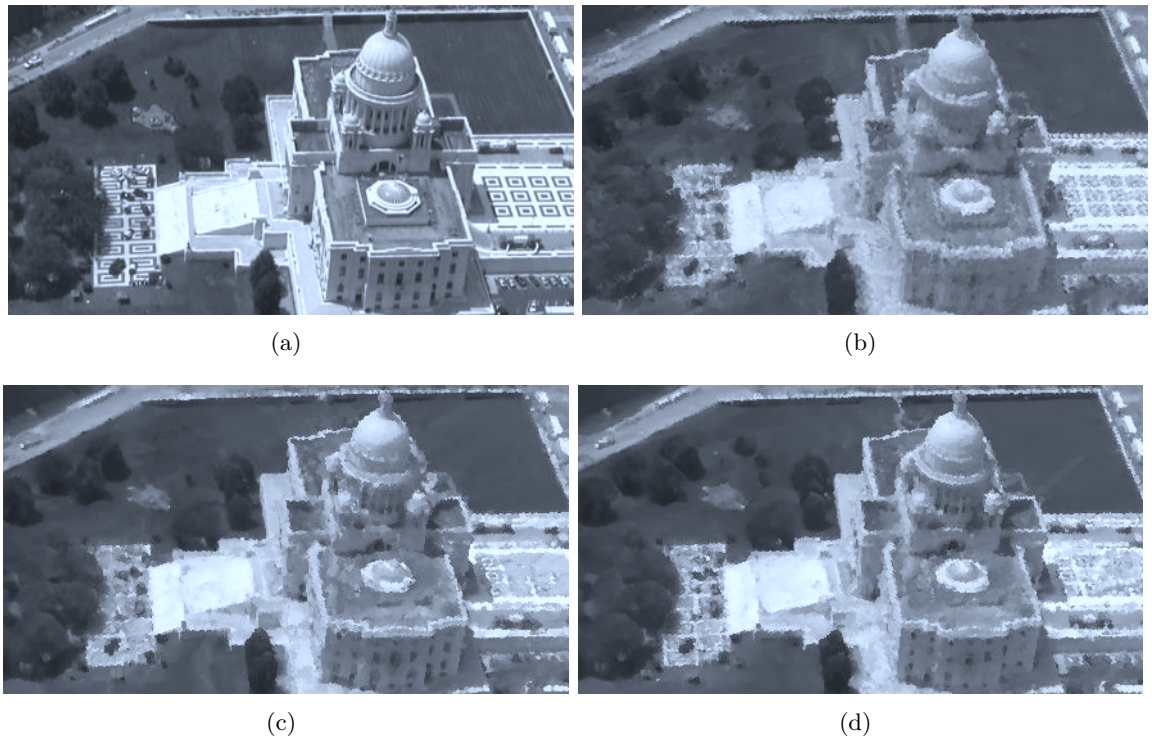


Figure 7.40: capitol40: sub-region visual comparison between (a) test image (ground truth), (b) rendered image from the voxel world model, (c) rendered image from IRAY, and (d) rendered image from IRAY with prior terms disabled.

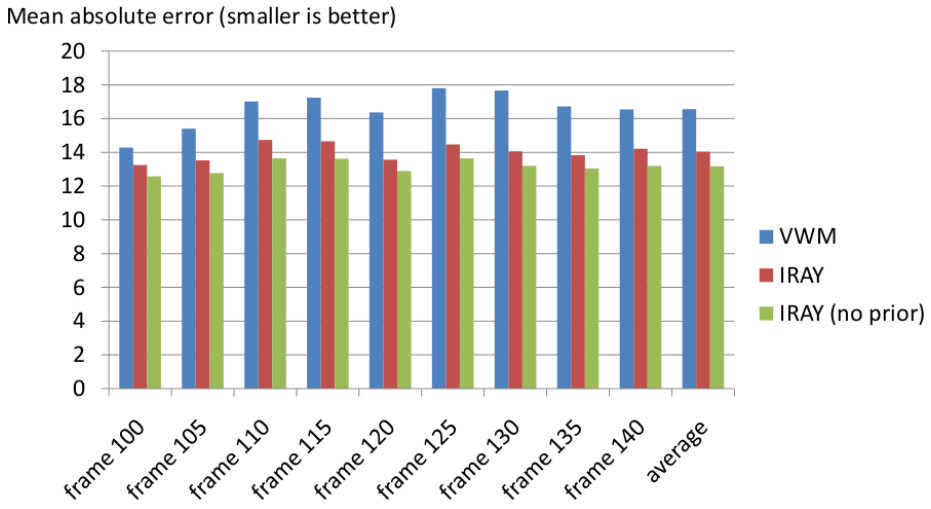


Figure 7.41: captiol40: mean absolute error of the synthetic images over the cropped center region for both VWM, IRAY and IRAY without prior

Chapter 8

Deep Belief Propagation For Efficient Higher-Order MRF Inference

Structures are the weapons of the mathematician.
— Nicolas Bourbaki (1939-1980)

In Chapter 5, the deep belief propagation algorithm is developed for the semi-global ray clique, by exploring its recursive chain structure. In this chapter, it is generalized to a more general class of higher-order MRFs. Recent successes in applying MRF to computer vision problems have also revealed its deficiency in capturing long-range variable interactions. This has motivated people to study prior cliques with a large number of variables, i.e., higher-order prior cliques. Another motivation for studying higher-order data cliques comes from the long-range sensing, where sensed signals record the interaction between large number of random variables, such as the pixel observation of a set of voxels. These higher-order MRFs have posed challenges to the corresponding inference problem. In this chapter, it is shown that although these higher-order cliques cannot be further factorized as smaller cliques, there exists another kind of "deep" factorization structure of these cliques, and these structures can be handled by dynamic programming. The developed dynamic programming algorithm can be further implemented using message passing, resulting in the deep belief propagation algorithm for higher-order MRF inference.

In the last decade, enormous empirical success stories have been reported in applying pair-wise MRF in low-level vision problems, including stereo, optical flow, image de-noising, image in-painting, texture analysis and synthesis, etc. At the same time, the pair-wise

MRF's incapacity to handle higher-order image regularities was exposed. This observation motivates the work on higher-order Markov random fields [149, 112]. To handle the challenging inference problem in higher-order MRF, different methods have been proposed. In [149], Zhu, Wu and Mumford computed the estimates from samples generated with the Markov Chain Monte Carlo (MCMC) algorithm. However, the convergence of the sampling-based approach is very slow. In the Fields-of-Experts model on image de-noising, Roth and Black [112] did the estimation through gradient descent based optimization. In that work, due to the nature of the problem, the noisy image can serve as a good initial guess, so the local solution is reasonably good. But the good initial guess is not always available for other applications. Lan et al. [76] proposed an approximation method for BP to make efficient inference possible in higher order MRFs. However, their results indicate that the approximation they made to BP only provides non-obviously better results to the naive gradient descent method. In [4], Ali et al. showed that a higher order polynomial can be efficiently transformed into a quadratic function. Then energy minimization tools for the pair-wise MRF models can be easily applied to the higher order counterparts. Kohli et al. [68, 69] introduced a class of higher order clique potentials and showed that the expansion and swap moves for any energy function composed of these potentials can be found by minimizing a sub-modular function. They also showed that for a subset of these potentials, referred to as the $\mathcal{P}^n$ Potts model, the optimal move can be found by solving a min-cut problem. In [70], Komodakis and Paragios proposed a powerful master-slave based framework for high-order MRF optimization. It allows decomposing (in a principled manner) a difficult high-order MRF problem into a set of easy-to-handle optimization subtasks for which efficient customized inference techniques can be easily exploited. In [59], Ishikawa introduced another technique to convert higher-order MRF to a first-order one, and showed a working example of a third-order MRF. In summary, several special groups of higher-order MRFs have been identified, for which efficient inference algorithms have been proposed. This thesis presents a solution based on the belief propagation for another group of higher-order MRFs.

8.1 Deep Belief Propagation for Higher-Order Cliques

In the min-sum version of belief propagation, a "small" optimization problem (Eq. (5.16)) needs to be solved at each iteration. The size of the optimization problem is proportional to the number of random variables in each clique, i.e., $|X_f|$, in Markov Random Fields. In traditional MRFs, $|X_f|$ is not only much smaller than the original problem size $|X|$, but also small in the absolute sense (usually $|X_f| = 2$, and there has been recent work on

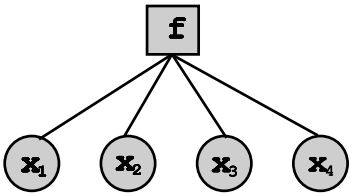


Figure 8.1: Factor graph representation of the clique $f(x_1, x_2, x_3, x_4)$

cliques of dozens of variables). But for higher-order cliques, e.g., the Ray MRF discussed in the Chapter 4, the size could be very large (100 — 1000, and even higher). Then the "small" optimization problem is no longer small for higher-order cliques. In this chapter, this problem is solved by exploring the *deep factorization* structure of the higher-order clique energy function. It is also shown that the structure can be utilized by dynamic programming to make the computation more efficient; and furthermore, the dynamic programming can be implemented with a message passing algorithm, similar to the traditional belief propagation.

Where do the higher-order cliques come from? There are at least two origins of these higher-order cliques. First the lower-order cliques (e.g., the pair-wise cliques) in the traditional MRFs are often insufficient to capture the higher-order prior statistics of the problem. *Higher-order prior cliques* in image processing can capture long-range pixel interactions, e.g., straight lines, flat surfaces, long-range curves, etc, which could improve the quality of de-noising, in-painting, stereo, shape from shading and also multi-view stereo. Secondly, many quantities are not directly observable; each observation is a record of the (complex) interactions between many hidden variables. These complex observation processes define *higher-order data cliques*, for example, the ray clique discussed in the last chapters. However, most of the algorithms developed for MRFs are just for the pair-wise cliques. In general, higher-order cliques are hard to deal with and have usually been studied case by case [101, 149, 112].

As shown in section 5.1, belief propagation is essentially a message passing implementation of the dynamic programming algorithm for computing the marginal integral or MAP estimation. Following the same spirit, in this chapter it will be shown that an efficient belief propagation algorithm, named as deep belief propagation, can be developed by using the dynamic programming algorithm to exploit the deep factorization structure of the higher-order clique, and by devising a message-passing implementation for the dynamic programming algorithm.

8.1.1 Dynamic Programming for Higher-Order Cliques

Take the following simple 4th-order clique energy,

$$f(x_1, x_2, x_3, x_4) \tag{8.1}$$

as an example. Its factor graph is shown in Fig. 8.1. This clique energy function cannot be further factorized as a summation of smaller clique energy functions.

Let's take a look a the message sent from the clique to the 2nd variable: $M_{f\to x_2}(x_2)$.

$$\begin{aligned} & M_{f\to x_2}(x_2) \\ &= \min_{x_1,x_3,x_4} \{f(x_1, x_2, x_3, x_4) + M_{x_1\to f}(x_1) + M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\}. \end{aligned} \tag{8.2}$$

Suppose random variables, x_1, x_2, x_3, x_4, all take K states. The minimization operation in Eq. (8.2) takes K^3 operations. Through simple induction, it can be seen that for a clique with N variables, each message from the clique to one of its participating variables takes K^{N-1} operations.

Although the clique energy $f(x_1, x_2, x_3, x_4)$ cannot be further factorized into a summation of smaller energy functions, what if it can be written as a function of two intermediate random variables z_1 and z_2, which depend on a subset of the original random variables x_1, x_2, x_3, x_4? That is,

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= h(z_1, z_2) \\ z_1 &= \phi_1(x_1, x_2) \\ z_2 &= \phi_2(x_3, x_4), \end{aligned} \tag{8.3}$$

Can this structure help us reduce the computational cost of message passing? Take a look at the the following equations.

$$\begin{aligned} & M_{f\to x_2}(x_2) \\ &= \min_{x_1,x_3,x_4} \{f(x_1, x_2, x_3, x_4) + M_{x_1\to f}(x_1) + M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\} && (8.4) \\ &= \min_{x_1,x_3,x_4} \{h(\phi_1(x_1, x_2), \phi_2(x_3, x_4)) + M_{x_1\to f}(x_1) + M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\} && (8.5) \\ &= \min_{x_1,z_2} \left\{ h(\phi_1(x_1, x_2), z_2) + M_{x_1\to h}(x_1) + \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to h}(x_3) + M_{x_4\to h}(x_4)\} \right\} && (8.6) \\ &= \min_{x_1} \left\{ M_{x_1\to h}(x_1) + \min_{z_2} \left\{ h(\phi_1(x_1, x_2), z_2) + \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to h}(x_3) + M_{x_4\to h}(x_4)\} \right\} \right\} && (8.7) \end{aligned}$$

In Eq. (8.5), the original clique energy function $f(x_1, x_2, x_3, x_4)$ is rewritten as a hierarchical function $h(\phi_1(x_1, x_2), \phi_2(x_3, x_4))$. In Eq. (8.6), through change of variables, the optimization variables are changed from $\{x_1, x_3, x_4\}$ to $\{x_1, z_2\}$. Then in Eq. (8.7) the order of optimization is distributed to different terms to reduce computation. Because the number of states the intermediate variable z_2 can take is at most K^2, the equivalent new optimization in Eq. (8.7) takes at most $(K^2 + K^2 + K = 2K^2 + K)$ operations. In many applications, the intermediate variables take the same number of states as the original random variable, i.e., K. In the above example, the computational cost will be $K^2 + 2K$.

As the number of variables increases and the "factorization" gets deeper, the reduction of computational costs will be more significant. The exact computational complexity depends on the number of states the intermediate random variables take. If the intermediate random variables take K states, then the computational complexity can be reduced from the original exponential complexity K^N to polynomial K^{w+1}, where w is the tree-width of the deep factor graph. Also the components in Eq. (8.7) can be re-used when computing the message from f to other random variables. The above observation is similar to what has been discussed in Section. 5.1.2: dynamic programming is used again!

From the above example, it can be seen that although the higher-order cliques cannot be further factorized, there are still other kind of structures that can be exploited to reduce the computation, and in some cases, the computational reduction can be huge, as will be demonstrated later. The structure exploited here is that some clique energies can be written as hierarchical functions of their participating random variable through some intermediate variables, and these intermediate variables can have small numbers of states (comparable to the number of states that the original variables have). This is the idea of "deep" factorization.

8.1.2 Deep Factor Graph

Similar to the factor graph representation, a *deep factor graph* representation of the deep factorization of the clique is introduced to explicitly represent the graphical structure of the deep factorization. The deep factor graph of the above example is shown in Fig. 8.2. Compared with the original factor graph of the example, Fig. 8.1, the following two differences can be observed: 1) in deep factor graph, in addition to the variable nodes and the clique nodes, there is another other type of nodes: intermediate variable nodes represented by triangles; 2) the undirected links between factor nodes and variable nodes in the traditional factor graph are replaced by directed links between factor nodes, intermediate

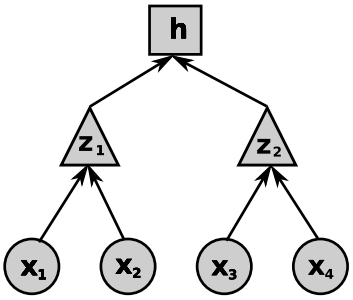


Figure 8.2: Deep factor graph of a clique node

variable nodes, and original variable nodes. The arrow is to indicate the decomposition relationship clearly (especially in the case of cyclic-graph decomposition), where in the traditional factor graph representation the arrow is omitted because the decomposition is clear (the arrow always goes from the random variable nodes to the factor nodes). Here, the nodes on an arrow are having the parent-child relationship. For example, in Fig. 8.2, z_2 is a parent node of x_3 and x_4, and also a child node of h.

8.1.3 Deep Belief Propagation

This section presents a message passing implementation of the dynamic programming algorithm, which has been illustrated with the above example, on the deep factor graph, by augmenting the original belief propagation message passing rules. It is called the *deep belief propagation (DBP)*, see Algorithm 3.

For the above example, let's use the deep belief propagation message passing rules to compute the message from f to x_2, $M_{f\to x_2}$, which is equivalent to $M_{z_1\to x_2}$, by definition. Firstly, the message from x_3 and x_4 to z_2 is computed as

$$M_{x_3\to z_2}(x_3) = M_{x_3\to f}(x_3), \tag{8.10}$$

$$M_{x_4\to z_2}(x_4) = M_{x_4\to f}(x_4). \tag{8.11}$$

Secondly, the message from z_2 to h is computed as

$$M_{z_2\to h}(z_2) = \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to z_2}(x_3) + M_{x_4\to z_2}(x_4)\} \tag{8.12}$$

$$= \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\} \tag{8.13}$$

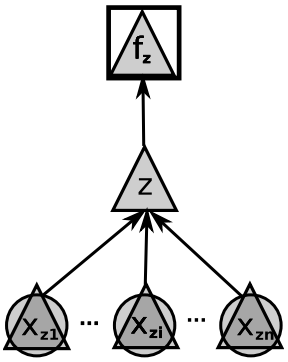


Figure 8.3: A branch of the deep factor graph centered around an intermediate variable node z. Here the parent node f_z of z could be a factor/clique node or an intermediate variable node, so it is represented by an overlapped shape of square and triangle; each child node could be an original variable node or an intermediate variable node, so it is represented by an overlapped shape of circle and triangle.

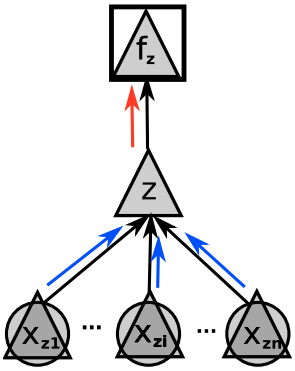


Figure 8.4: Message flow from an intermediate variable node z to its parent node f_z

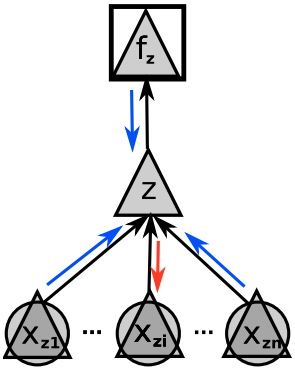


Figure 8.5: Message flow from an intermediate variable node z to its child node x_{zi}

Algorithm 3 Deep Belief Propagation Algorithm

1. For the messages sent from and to the factor nodes and the original variable nodes, the belief propagation algorithm in Sec. 5.1.3 applies.

2. For an intermediate variable z, denote its parent node as f_z, denote its n children nodes as $X_z = \{x_{z1}, x_{z2}, \cdots, x_{zn}\}$, with the relationship $z = \phi(X_z) = \phi(x_{z1}, x_{z2}, \cdots, x_{zn})$ (see its deep factor graph shown in Fig. 8.3). Then the message from z to its parent node f_z (see Fig. 8.4) is computed as

$$M_{z \to f_z}(z) = \min_{\phi(x_{z1}, x_{z2}, \cdots, x_{zn}) = z} \left(\sum_{i=1}^{n} M_{x_{zi} \to z}(x_{z_i}) \right). \tag{8.8}$$

   The message from z to its child node x_{zi} (see Fig. 8.5) is computed as

$$M_{z \to x_{zi}}(x_{zi}) = \min_{x_{zj} \in X_z \setminus x_{zi}} \left(M_{f_z \to z}(z) + \sum_{j \neq i} M_{x_{zj} \to z}(x_{zj}) \right). \tag{8.9}$$

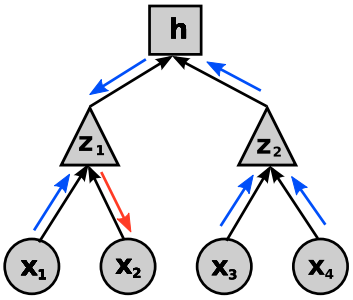


Figure 8.6: Message passing on deep factor graph

Then, the message from h to z_1 is

$$M_{h\to z_1}(z_1) = \min_{z_2}\{h(z_1,z_2) + M_{z_2\to h}(z_2)\} \tag{8.14}$$

$$= \min_{z_2}\left\{ h(z_1,z_2) + \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\} \right\} \tag{8.15}$$

Finally, the message from z_1 to x_2 is

$$M_{z_1\to x_2}(x_2) = \min_{x_1}\{M_{x_1\to z_1}(x_1) + M_{h\to z_1}(\phi_1(x_1,x_2))\} \tag{8.16}$$

$$= \min_{x_1}\left\{ M_{x_1\to z_1}(x_1) + \min_{z_2}\left\{ h(z_1,z_2) + \min_{\substack{x_3,\,x_4\,:\\ \phi_2(x_3,\,x_4)\,=\,z_2}} \{M_{x_3\to f}(x_3) + M_{x_4\to f}(x_4)\} \right\} \right\} \tag{8.17}$$

The above value is exactly the message, Eq.(8.7), computed with dynamic programming previously in Section 8.1.1

In the above, tree-structured deep factor graph has been studied. What's the solution to a general graph structure? Actually, the cycles in the graph can be clustered into single nodes, which converts the loopy graph into a tree. The exact technique for performing this surgery to the graph is the junction-tree algorithm. The created tree structure is called junction tree. The technique has been used in extending belief propagation to non-tree graphical models to compute the exact marginal and maximal distributions [78, 102]. If the variable nodes are discrete with K states, then junction-tree algorithm takes $O(NK^{w+1})$, where N is the number of nodes, and w is the tree-width of the graph. Similarly, the junction-tree algorithm can be applied to convert the loopy deep factor graph into a tree-structured graph. And the deep belief propagation can be extended to handle the general deep factor graph. The computational complexity depends on the sparsity of the deep factor graph, measured by the tree-width of the graph.

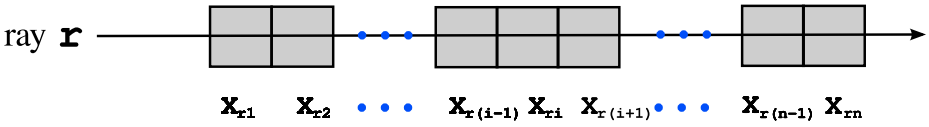


Figure 8.7: A ray passes through n voxels

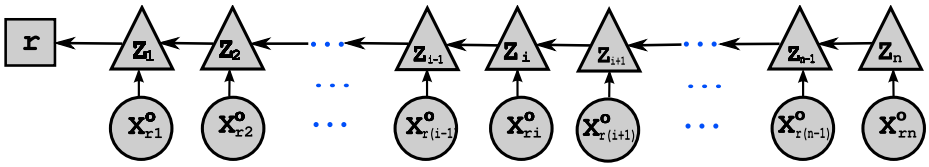


Figure 8.8: Deep factorization of ray clique, i.e., the deep factor graph of ray clique

The above simple example cannot fully demonstrate the computational cost saving of the deep belief propagation for higher-order cliques. Next its application in the Ray MRF inference is analyzed and the dramatic computational saving it can achieve is shown.

8.2 Deep Factorization of the Ray Clique

In Chapter 5, the deep belief propagation for the ray message passing was derived through induction: detecting and re-using common terms. Here it is shown that the same result can be derived more easily with the deep belief propagation idea developed above.

First let's see how to deeply factorize the ray clique energy $E_r(X_r) = (I_r - \Phi_r(X_r))^2$. Define z_1 the index of the visible voxel on the ray r. Then

$$\Phi_r(X_r) = x^c_{rz_1}. \tag{8.18}$$

Consider the first voxel on the ray x_{r1}: if it is solid, then it is the visible voxel, i.e., $z_1 = 1$; if the first voxel is empty, then the remaining voxels will be rendered in a similar way. Denote z_2 the index of the visible voxel on the ray starting from the 2nd voxel x_{r2}. Then if the first voxel is empty, then $z_1 = z_2$. In summary,

$$z_1 = \phi(x^o_{r1}, z_2) = \begin{cases} 1, & x^o_{r1} = 1 \\ z_2, & x^o_{r1} = 0. \end{cases} \tag{8.19}$$

The above procedure can be repeated recursively, i.e.,

$$z_i = \phi(x^o_{ri}, z_{i+1}) = \begin{cases} i, & x^o_{ri} = 1 \\ z_{i+1}, & x^o_{ri} = 0 \end{cases} \tag{8.20}$$

By defintion, z_i takes values from i to n, i.e.,

$$z_i \in \{i, i+1, \cdots, n\} \tag{8.21}$$

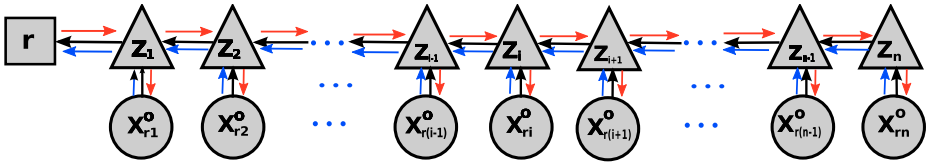


Figure 8.9: Message passing on ray clique deep factor graph

until the last voxel, where

$$z_{rn} = n. \tag{8.22}$$

The above recursive formulas show the hierarchical chain structure of the ray clique energy. The corresponding deep factor graph of the ray clique energy is shown in Fig. 8.8. It can be seen that the ray clique has a recursive chain structure in the deep factor graph and each intermediate variable z_i has at most n states. So by the deep belief propagation, an $O(n)$ complexity message passing algorithm can be derived for sending messages from the ray clique to each voxel.

The deep belief propagation algorithm for the ray clique can be written down by following the rules in Algorithm 3. Start with the message passing backwards. Message from z_{i+1} to z_i is computed as:

$$\begin{aligned}
& M_{z_{i+1}\to z_i}(z_{i+1}) \\
= \;& \min_{\phi_{i+1}(x^o_{r(i+1)},z_{i+2})=z_{i+1}}(M_{x^o_{r(i+1)}\to z_{i+1}}(x^o_{r(i+1)}) + M_{z_{i+2}\to z_{i+1}}(z_{i+2})) \\
= \;& \min\left\{M_{x^o_{r(i+1)}\to z_{i+1}}(0) + M_{z_{i+2}\to z_{i+1}}(z_{i+1}), M_{x^o_{r(i+1)}\to z_{i+1}}(1) + \min_{z_{i+2}} M_{z_{i+2}\to z_{i+1}}(z_{i+2})\right\} \\
= \;& \min\left\{M_{x^o_{r(i+1)}\to r}(0) + M_{z_{i+2}\to z_{i+1}}(z_{i+1}), M_{x^o_{r(i+1)}\to r}(1) + \min_{z_{i+2}} M_{z_{i+2}\to z_{i+1}}(z_{i+2})\right\}.
\end{aligned} \tag{8.23}$$

Message from $M_{z_{i-1}}$ to z_i is computed as:

$$\begin{aligned}
& M_{z_{i-1}\to z_i}(z_i) \\
= \;& \min_{x^o_{r(i-1)}}(M_{x^o_{r(i-1)}\to z_{i-1}}(x^o_{r(i-1)}) + M_{z_{i-2}\to z_{i-1}}(z_{i-1})) \\
= \;& \min\left\{M_{x^o_{r(i-1)}\to z_{i-1}}(0) + M_{z_{i-2}\to z_{i-1}}(z_i), M_{x^o_{r(i-1)}\to z_{i-1}}(1) + M_{z_{i-2}\to z_{i-1}}(i-1)\right\} \\
= \;& \min\left\{M_{x^o_{r(i-1)}\to r}(0) + M_{z_{i-2}\to z_{i-1}}(z_i), M_{x^o_{r(i-1)}\to r}(1) + M_{z_{i-2}\to z_{i-1}}(i-1)\right\}.
\end{aligned} \tag{8.24}$$

Then the message from to $M_{z_{ri}\to x_{ri}}(x_{ri})$, which is exactly the message to compute, $M_{r\to x_{ri}}(x_{ri})$, by definition, can be computed as:

$$\begin{aligned}
& M_{r\to x^o_{\mathrm{ri}}}(x^o_{\mathrm{ri}}) \\
= \;& M_{z_i\to x^o_{\mathrm{ri}}}(x^o_{\mathrm{ri}}) \\
= \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(z_i) + M_{z_{i+1}\to z_i}(z_{i+1})) \\
= \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(\phi_i(x^o_{\mathrm{ri}}, z_{i+1})) + M_{z_{i+1}\to z_i}(z_{i+1})).
\end{aligned} \tag{8.25}$$

That is,

$$\begin{aligned} & M_{r\to x^o_{\mathrm{ri}}}(0) \\ = \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(\phi_i(0,z_{i+1})) + M_{z_{i+1}\to z_i}(z_{i+1})) \\ = \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(z_{i+1}) + M_{z_{i+1}\to z_i}(z_{i+1})) \\ = \;& \min_{z_{i+1}} M_{z_{i-1}\to z_i}(z_{i+1}) + \min_{z_{i+1}} M_{z_{i+1}\to z_i}(z_{i+1})). \end{aligned} \tag{8.26}$$

$$\begin{aligned} & M_{r\to x^o_{\mathrm{ri}}}(1) \\ = \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(\phi_i(1,z_{i+1})) + M_{z_{i+1}\to z_i}(z_{i+1})) \\ = \;& \min_{z_{i+1}}(M_{z_{i-1}\to z_i}(i) + M_{z_{i+1}\to z_i}(z_{i+1})) \\ = \;& M_{z_{i-1}\to z_i}(i) + \min_{z_{i+1}} M_{z_{i+1}\to z_i}(z_{i+1}). \end{aligned} \tag{8.27}$$

Chapter 9

Conclusion and Discussion

This thesis presents a one-step optimization formulation of the image-based 3-d modeling problem, and develops an efficient algorithm to solve the corresponding large-scale optimization problem. The formulation models the ray-tracing based image generation process directly and can handle occlusion accurately. The proposed algorithm has also been generalized to solve the inference problem in higher-order Markov random fields. The framework and algorithm is shown to be capable of handling general and complex scenes, verified by extensive experiments on standard and locally collected challenging datasets. Compared with point cloud based approaches, the proposed method can reconstruct more complete surfaces and handle sparse cameras. Compared with surface based approaches and graph-cuts based volumetric approaches, this approach does not require a good initial surface estimate. Compared with other volumetric approaches, it provides a better global-optimization-based solution, and generates more accurate geometric models.

Here is a summary of the properties of the proposed approach.

Accuracy The accuracy is comparable to the state-of-the-art algorithms with the current proof-of-concept implementation. The main limitation of the current implementation is the high memory requirement due to the large number of voxels. Better implementation would permit higher resolution and would handle the larger number of voxels by using sparse representation of the volume, e.g., octree representation of the volume.

Photo-Reality The reconstruction is photo-realistic due to the nature of the approach: it tries to reproduce each image with a 3-d model. So each pixel has an explanation in the model.

Robustness The proposed algorithm can handle objects of complex topology, arbitrary

camera configurations (small or large number of cameras, small base-line or large base-line), and can handle background and transient clutter automatically.

Minimum User Interaction The current implementation only requires the user to specify a bounding box, which can be relaxed easily. When the implementation can handle large resolution, it is natural to pick up a region based on the intersection of the cameras' viewing frustum.

Certainly there are many aspects to be improved. The following is a list of promising directions to explore in the future.

Sparse Representation and Large Scale Reconstruction In the current implementation, an evenly divided volume serves as the main 3-d representation of the scene. But most of the voxels are either totally empty or solid, the number of voxels surrounding the surface is much smaller than the total number of voxels. So there is a lot of waste in both memory and computation. One solution is to use a sparse representation of the volume, e.g., octree. Octree data structure can be used for a hierarchical sparse representation of the volume. The benefit of using octree is two-fold: one is to improve the scalability of the system; another is to get a compact 3-d model. The first benefit is obvious: the spatial and computational requirement can be dramatically reduced. Let's take a look at the second one. The mesh generated using the marching-cube algorithm for the volume is very large, with millions of vertices and edges. But it can be reduced by orders of magnitude without noticeable difference in representation accuracy through mesh simplification operation. With the octree based representation, a very compact mesh representation of the object can extracted with the iso-surface extraction methods for octree volume [65]. And the marching cubes algorithm has also been extended to handle an octree representation to create compact mesh representations [142, 121, 140]. There have been a numbewr of works on octree-based ray tracing [79], which results can be used in this work. And recently, highly efficient algorithms have been proposed to process giga-voxels in real-time using GPU acceleration. These demonstrate the expression power of the octree-based volume representation and its potential for high performance visualization [75]. The main challenges include 1) multi-resolution MRF modeling of the problem, and 2) dynamically updating the octree in each iteration based on the probabilities computed for each voxel. In [30], Crispell and Mundy have developed algorithms to handle similar problems.

More Realistic Appearance Model In this work, the Lambert shading model is assumed. The estimation of the appearance is the coupled color of an object's material and lighting. To get a realistic reconstruction that could be re-lighted under different illumination, lighting and an object's material need to be separated using a more general shading model. This problem, known as the full inverse rendering or full inverse ray tracing, has been discussed at the end of Chapter 3.

Taking Advantage of GPU Modern GPUs have evolved to be flexible enough to accelerate ray tracing to render complex scenes in real time. As the reverse operation of ray tracing, statistical reverse ray tracing should also be able to take advantage of the GPU hardware. The improvement on speed could be 10x–200x, extrapolated from the reported performance speedup in other similar image processing problems. For example, the GPU accelerated version of the voxel world model algorithm has been reported to be 200x faster than the CPU version [94].

Bibliography

[1] flickr, http://www.flickr.com.

[2] Kinect brings games and entertainment to life in extraordinary new ways without using a controller, http://www.xbox.com/en-us/kinect.

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proceedings of International Conference on Computer Vision*, pages 72–79, 2009.

[4] A. M. Ali, A. A. Farag, and G. L. Gimelfarb. Optimizing binary MRFs with higher order cliques. In *Proceedings of European Conference on Computer Vision*, pages 98–111, 2008.

[5] A. Appel. On calculating the illusion of reality. *IFIP Congress*, 2:945–950, 1968.

[6] M. B. Averintsev. On a method of describing complete parameter fields. *Problemy Peredaci Informatssi*, 6:100–109, 1970.

[7] S. Avidan and A. Shamir. Seam carving for content-aware image retargeting. In *ACM SIGGRAPH*, 2007.

[8] R. J. Baxter. *Exactly solved models in statistical mechanics*. Academic Press, 1992.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[10] F. Bernardini, I. Martin, J. Mittleman, H. Rushmeier, and G. Taubin. Building a digital model of Michelangelo's Florentine Pieta. *IEEE Computer Graphics & Applications*, 22:59–67, 2002.

[11] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[12] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing Ltd, 1998.

[13] J. E. Besag. Nearest-neighbour systems and the auto-logistic model for binary data. *J. Roy. Stat. Soc. (B)*, 34:75–83, 1972.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning*, chapter Graphical Models, pages 359–422. Springer, 2006.

[15] J.-Y. Bouguet. Camera calibration toolbox for matlab. available at `http://www.vision.caltech.edu/bouguetj/calib_doc`.

[16] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of International Conference on Computer Vision*, pages 26–33, 2003.

[17] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[18] A. Broadhurst. *A Probabilistic Framework for Space Carving*. PhD thesis, University of Cambridge, 2001.

[19] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *Proceedings of International Conference on Computer Vision*, pages 388–393, 2001.

[20] N. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Proceedings of European Conference on Computer Vision*, pages 766–779, 2008.

[21] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1261–1268, 2010.

[22] R. Chellappa and A. Jain, editors. *Markov Random Fields: Theory and Application*. Boston: Academic Press, 1993.

[23] P. Cignoni, G. Ranzuglia, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, N. Pietroni, M. Tarini, and et al. Meshlab: an open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes, http://meshlab.sourceforge.net/.

[24] F. S. Cohen and D. B. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):195–219, 1987.

[25] F. S. Cohen and D. B. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields. (2002). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:195–219, 1987.

[26] F. S. Cohen, D. B. Cooper, J. F. Silverman, and E. B. Hinkle. Simple parallel hierarchical and relaxation algorithms for segmenting textured images based on noncausal markovian random field models. In *Proceedings of International Conference on Pattern Recognition*, pages 1104–1107, 1984.

[27] D. B. Cooper and Z. Lei. On representation and invariant recognition of complex objects based on patches and parts. In *International NSF-ARPA Workshop on Object Representation in Computer Vision*, 1994.

[28] D. B. Cooper, J. Subrahmonia, Y. P. Hung, and B. Cernuschi. *Markov Random Fields: Theory and Application*, chapter The use of Markov random fields in estimating and recognizing objects in 3D space. Academic Press, 1993.

[29] D. B. Cooper and F. P. Sung. Multiple-window parallel adaptive boundary finding in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983.

[30] D. E. Crispell. *A Continuous Probabilistic Scene Model for Aerial Imagery*. PhD thesis, Brown University, 2010.

[31] A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *Proceedings of British Machine Vision Conference*, 2008.

[32] Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3D and 2D data using PDE-based deformable surfaces. In *Proceedings of European Conference on Computer Vision*, pages 238–251, 2004.

[33] I. Eden and D. B. Cooper. Using 3D line segments for robust and efficient change detection from multiple noisy images. In *Proceedings of European Conference on Computer Vision*, pages 172–185, 2008.

[34] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[35] R. Fabbri and B. B. Kimia. 3D curve sketch: Flexible curve-based stereo reconstruction and calibration. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1538–1545, 2010.

[36] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDEs, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7:336–344, 1998.

[37] O. D. Faugeras, Q.-T. Luong, and S. Maybank. Camera self-calibration: Theory and experiments. In *Proceedings of European Conference on Computer Vision*, pages 321–334, 1992.

[38] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70:41–54, 2006.

[39] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345–345, 1962.

[40] D. Fofi, T. Sliwa, and Y. Voisin. A comparative survey on invisible structured light. In *Proc. SPIE Electronic Imaging - Machine Vision Applications in Industrial Inspection XII*, 2004.

[41] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[42] C. Fruh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60(1):5–24, October 2004.

[43] Y. Furukawa. *High-Fidelity Image-Based Modeling*. PhD thesis, University of Illinois at Urbana-Champaign, 2008.

[44] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010.

[45] Y. Furukawa and J. Ponce. Patch-based multi-view stereo software (pmvs - version 2), http://grail.cs.washington.edu/software/pmvs/.

[46] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *Proceedings of European Conference on Computer Vision*, pages 564–577, 2006.

[47] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[48] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[49] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[50] S. Geman and K. Kochanek. Dynamic programming and the graphical representation of error-correcting codes. *IEEE Transactions on Information Theory*, 47:549–568, 2001.

[51] S. Geman, K. Manbeck, and D. E. McClure. *Markov Random Fields: Theory and Applications*, chapter A comprehensive statistical model for single photon emission tomography, pages 93–130. Academic Press, 1993.

[52] S. Geman and D. E. McClure. Bayesian image analysis: An application to single photon emission tomography. In *1985 Proceedings of the American Statistical Association. Statistical Computing Section*, pages 12–18, 1985.

[53] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2402–2409, 2006.

[54] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *Proceedings of International Conference on Computer Vision*, pages 1–8, 2007.

[55] U. Grenander and M. Miller. *Pattern Theory: From Representation to Inference*. Oxford University Press, 2007.

[56] T. Haber, C. Fuchs, P. bekaert, H.-P. Seidel, M. Goesele, and H. P. A. Lensch. Relighting objects from image collections. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 627–634, 2009.

[57] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. (unpublished), 1971.

[58] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1430–1437, 2009.

[59] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2993–3000, 2009.

[60] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo beyond Lambert. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 171–178, 2003.

[61] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, 2005.

[62] J. T. Kajiya. The rendering equation. In *ACM SIGGRAPH*, pages 143–149, 1986.

[63] T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, 1965.

[64] M. Kazhdan and M. Bolitho. Poisson surface reconstruction (version 2), http://www.cs.jhu.edu/ misha/code/poissonrecon/.

[65] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symposium Geometry Processing (SGP)*, pages 61–70, 2006.

[66] J. H. Kim and J. Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of International Joint Conferences on Artificial Intelligence*, pages 190–193, 1983.

[67] D. E. Knuth. *The Art of Computer Programming Volume 2: Seminumerical Algorithms*, pages 501–501. Addison-Wesley Professional, 1997.

[68] P. Kohli, M. P. Kumar, and P. H. Torr. P3 & beyond: Solving energies with higher order cliques. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[69] P. Kohli, M. P. Kumar, and P. H. Torr. P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1645–1656, 2009.

[70] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2985–2992, 2009.

[71] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[72] A. Kushal and J. Ponce. Modeling 3D objects from stereo views and recognizing them in photographs. In *Proceedings of European Conference on Computer Vision*, pages 563–574, 2006.

[73] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 2000.

[74] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *Proceedings of International Conference on Computer Vision*, pages 1–8, 2007.

[75] S. Laine and T. Karras. Efficient sparse voxel octrees. In *ACM SIGGRAPH*, 2010.

[76] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *Proceedings of European Conference on Computer Vision*, pages 269–282, 2006.

[77] D. Lanman and G. Taubin. Build your own 3D scanner: 3D photograhy for beginners. In *ACM SIGGRAPH 2009 courses*, pages 1–87, New Orleans, LA USA, 2009.

[78] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50:157–224, 1988.

[79] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9:245–261, 1990.

[80] M. Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, pages 33–40, 1990.

[81] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang. Bundled depth-map merging for multi-view stereo. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2769–2776, 2010.

[82] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[83] S. Liu and D. B. Cooper. Ray Markov random fields for image-based 3D modeling: model and efficient inference. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1530–1537, San Francisco, CA, 2010.

[84] S. Liu and D. B. Cooper. A complete statistical inverse ray tracing approach to multi-view stereo. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, 2011.

[85] S. Liu, K. Kang, J.-P. Tarel, and D. B. Cooper. Free-form object reconstruction from silhouettes, occluding edges and texture edges: A unified and robust operator based on duality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:131–146, 2008.

[86] S. Liu, K. Kang, J.-P. Tarel, and D. B. Cooper. Distributed volumetric scene geometry reconstruction with a network of distributed smart cameras. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2334–2341, Miami, FL, 2009.

[87] Y. Liu, X. Cao, Q. Dai, and W. Xu. Continuous depth estimation for multi-view stereo. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2121–2128, 2009.

[88] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH*, pages 163–169, 1987.

[89] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[90] D. J. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[91] S. Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, 1998.

[92] J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of British Machine Vision Conference*, pages 384–393, 2002.

[93] S. Mchugh. Panoramic image projections. `http://www.cambridgeincolour.com/tutorials/image-projections.htm`.

[94] A. Miller, V. Jain, and J. L. Mundy. Real-time rendering and dynamic updating of 3-d volumetric data. In *Fourth Workshop on General Purpose Processing on Graphics Processing Units*, Newport Beach, CA, March 5 2011.

[95] K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief-propagation for approximate inference: An empirical study. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[96] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[97] F. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4:767–775, 1965.

[98] D. Nister and H. Stewenius. Linear time maximally stable extremal regions. In *Proceedings of European Conference on Computer Vision*, pages 183–196, 2008.

[99] A. S. Ogale and Y. Aloimonos. Shape and the stereo correspondence problem. *International Journal of Computer Vision*, 65(3):147–162, 2005.

[100] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[101] R. Paget and I. D. Longstaff. Texture synthesis via a noncausal nonparametric multiscale markov random field. *IEEE Transactions on Image Processing*, 7:925–931, 1998.

[102] M. A. Paskin. A short course on graphical models : 3. the junction tree algorithms. available at `http://ai.stanford.edu/~paskin/gm-short-course/lec3.pdf`.

[103] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 133–136, Pittsburgh, Pennsylvania, 1982.

[104] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 2 edition, 1988.

[105] T. B. Pollard. *Comprehensive 3-d Change Detection Using Volumetric Appearance Modeling*. PhD thesis, Brown University, 2009.

[106] T. B. Pollard and J. L. Mundy. Change detection in a 3D world. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007.

[107] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, Katholieke Universiteit Leuven, 1999.

[108] M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of International Conference on Computer Vision*, pages 90–95, 1998.

[109] J.-P. Pons and J.-D. Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay trangulation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[110] B. Potetz. Efficient belief propagation for vision using linear constraint nodes. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2005.

[111] S. F. Ray. *Applied Photographic Optics*. Focal Press, 3rd edition, 2002.

[112] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009.

[113] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 272–280, 2003.

[114] M. Rousson and N. Paragios. Shape priors for level set representations. In *Proceedings of European Conference on Computer Vision*, pages 78–92, 2002.

[115] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.

[116] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. Middlebury multi-view benchmark (http://vision.middlebury.edu/mview).

[117] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–526, 2006.

[118] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1067–1073, 1997.

[119] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[120] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[121] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *Proceedings of the 7th conference on Visualization*, pages 335–342, 1996.

[122] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proceedings of International Conference on Computer Vision*, pages 349–356, 2005.

[123] N. Snavely. Bundler – structure from motion for unordered image collections, http://phototour.cs.washington.edu/bundler.

[124] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. In *ACM SIGGRAPH*, 2006.

[125] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2000.

[126] C. Strecha, T. Tuytelaars, and L. V. Gool. Dense matching of multiple wide-baseline views. In *Proceedings of International Conference on Computer Vision*, pages 1194–1201, 2003.

[127] E. B. Sudderth and W. T. Freeman. Signal and image processing with belief propagation. *IEEE Signal Processing Magazine*, 25(2):114–120, 2008.

[128] E. B. Sudderth, A. Ihler, W. T. Freeman, and A. Willsky. Nonparametric belief propagation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 95–103, 2003.

[129] E. B. Sudderth, M. Mandel, W. T. Freeman, and A. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1369–1376, 2004.

[130] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1068–1080, 2008.

[131] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *Proceedings of International Conference on Computer Vision*, pages 900–906, 2003.

[132] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2005.

[133] S. Tran and L. Davis. 3D surface reconstruction using graph cuts with surface constraints. In *Proceedings of European Conference on Computer Vision*, pages 219–231, 2006.

[134] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal Robotics and Automation*, 3(4):323–344, 1987.

[135] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[136] G. Vogiatzis, C. H. Esteban, P. H. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.

[137] G. Vogiatzis, P. H. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 391–398, 2005.

[138] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[139] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques:Theory and Practice*. Addison-Wesley, Reading, Massachusetts, 1992.

[140] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, 15:100–111, 1999.

[141] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23:343–349, 1980.

[142] J. Wilhelms and A. V. Gelder. Octrees for faster iso-surface generation. *ACM Transactions on Graphics*, 11:201–227, 1992.

[143] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Proceedings of Advances in Neural Information Processing Systems*, pages 689–695, 2000.

[144] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[145] A. J. Yezzi and S. Soatto. Stereoscopic segmentation. In *Proceedings of International Conference on Computer Vision*, pages 59–66, 2001.

[146] T. Yu, H. Wang, N. Ahuja, and W.-C. Chen. Sparse lumigraph relight by illumination and reflectance estimation from multi-view images. In *Eurographics Symposium on Rendering (EGSR)*, 2006.

[147] S. Zhang and P. Huang. *High-resolution, Real-time 3-D Shape Measurement*. PhD thesis, Stony Brook University, 2005.

[148] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[149] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 2004.