

Abstract of “Smooth Signed Distance (SSD) Surface Reconstruction Framework”

by Fatih Çalaklı, Ph.D., Brown University, February 2021.

This thesis presents a global surface reconstruction framework that uses outputs of popular 3D shape acquisition methodologies (ranging from time-of-flight cameras, 3D laser scanners, structured light systems to multi-view stereo algorithms) as input, and generates detailed polygon mesh surfaces. As part of the framework, novel variational formulations are introduced targeted to reconstruct watertight surfaces from various 3D data representations such as oriented points, colored oriented points, calibrated depth maps, cross-sectional curves (parallel or non-parallel), and volumetric occupancy grids. The variational formulations proposed in different chapters are based on minimizing energy functionals composed of simple geometric constraints to estimate a surface implicitly defined as the zero level set of the Euclidean signed distance function to the surface, *i.e.*, the implicit function is forced to be a smooth approximation of the signed distance function to the surface. In contrast, the Poisson surface reconstruction approach forces the implicit function to approximate the indicator function of the volume bounded by the implicit surface. Since an indicator function is discontinuous, its gradient does not exist exactly where it needs to be compared with the normal vector data. The smooth signed distance has approximate unit slope in the neighborhood of the data points. As a result, the normal vector data can be incorporated directly into the energy function without implicit function smoothing. In detail, each energy functional consists of two data terms, and one regularization term to constrain the space of signed distance functions. The first data term forces the implicit function to be close to zero at the data points. The second data term forces the gradient of the function to be close to the corresponding normal vectors at the data points. The regularization term forces the gradient of the function to be constant away from the data points. The formulations allow for a number of different discretizations, reduce to scalable least squares problems for all linearly parameterized families of functions, and do not require boundary conditions. The resulting algorithms are significantly simpler and easier to implement, and efficient space-adaptive (octree-based) implementations are shown to produce high-quality adaptive manifold polygon meshes comparable with state-of-the-art algorithms.

Smooth Signed Distance (SSD) Surface Reconstruction Framework

by

Fatih Çalıklı

Sc. B., Boğaziçi University, 2007

Sc. M., Applied Mathematics, Brown University, 2012

Sc. M., Computer Science, Brown University, 2012

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

February 2021

© Copyright 2021 by Fatih Çalaklı

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

Date _____

Gabriel Taubin, Director

Recommended to the Graduate Council

Date _____

Daniel Ritchie, Reader
(Computer Science)

Date _____

Matthew T. Harrison, Reader
(Applied Mathematics)

Approved by the Graduate Council

Date _____

Andrew G. Campbell
Dean of the Graduate School

Vita

Fatih Calakli was born in Isparta, Turkey. He attended Boğaziçi University in Istanbul, Turkey, where he received his Bachelor of Science degree with honors in Electrical and Electronic Engineering in June 2007 with a senior thesis “A New Approach to Modeling Geometric Distortions and Its Use in Image Watermarking” advised by Prof. Kivanc Mihcak. He joined the Ph.D. program in Computer Engineering at Brown University in Providence, RI in Fall 2007. During the first years of the program, Fatih completed required coursework, and was awarded a Master of Science degree in Applied Mathematics. He later earned a Master of Science degree in Computer Science in May 2012 with a master thesis “High Resolution Colored Surface Reconstruction from Oriented Points” advised by Prof. Gabriel Taubin. Fatih spent the summer of 2012 as a research intern at Ricoh Innovations Inc., Cupertino, CA where he worked on a real-time 3D scanner project based on structured light, and contributed to a patent application with his manager John Barrus. Fatih was a member of the Computer Vision Research Group (LEMS: Laboratory for Engineering Man/Machine Systems) at Brown University from June 2009 to September 2013, contributed to 7 research publications and 2 patent applications in collaboration with Ali Osman Ulusoy, Daniel Moreno, Maria Isabel Restrepo, Fernanda Andalo, Siome Goldenstein, Joseph Mundy, and Gabriel Taubin. During the last three years of his studies, he worked on developing a general variational framework for Surface Reconstruction from various 3D data representations under the supervision of his doctoral adviser Prof. Gabriel Taubin. The framework presented in this thesis is the capstone of this effort.

Acknowledgements

I am deeply indebted to my doctoral adviser Prof. Gabriel Taubin. We met in January 2009 when I took his 3D Photography course at Brown University. Since then he always helped me to find my own ways, and become a complete researcher. I can not thank enough to my friends at Brown University: Ali Osman Ulusoy, Firat Kalaycilar, Daniel Moreno, Isabel Restrepo, Anil Usumezbas, Sarthak Khanal, Yahya Qasem, Alexandra Papoutsaki, Octavian Biris, Eduardo Almeida, Ozge Can Ozcanli, Ibrahim Eden, Alptekin Kupcu, Cagatay Demiralp, and many; whose company made my life fun and memorable in Providence. In particular, I am grateful to my friend Mehmet Kiral from the Math department. Our conversations were always motivational, and helped me not to quit my studies when I was in Turkey helping my mother to survive cancer. I thank to John Barrus who was my manager, and to Sapna Shroff who was my mentor at Ricoh. Their guidance helped me to finish my internship project on time with fruitful results. I am also thankful to my college friends Oytun Arslan, Ulas Ayaz, Yalcin Balcioglu, and Huseyin Ozkan, for always being available for a quick insightful conversation although we all live in different parts of the world. I would be ungrateful if I forgot to thank my former girlfriends: Gonca, Ivana, Nicole, Ceyda, and Sophie; for entering my life for a certain period of time, and making sure that my life was not entirely occupied with research. I am forever grateful to my parents Zubeyde & Sevkett, and my sister Dilek whose constant love helped me to overcome several challenges during this journey. This thesis is dedicated to *them*.

The framework presented in this thesis was mainly supported by the National Science Foundation under Grant No. IIS-0808718, CCF-0729126, and CCF-0915661. The author thanks the members of the Digital Michelangelo project, in particular, Szymon Rusinkiewicz and Benedict Brown, for providing non-rigid body aligned David head data; Donald H. Sanders for providing the images of the cuneiform brick; and Williams College Museum of Art, Williamstown, MA, for granting permission to use the images.

Contents

1	Introduction	1
1.1	Thesis statement	1
1.2	Motivation	1
1.3	Background	2
1.3.1	3D Data Acquisition	3
1.3.2	Registration of Multiple 3D Scans	5
1.3.3	3D Surface Reconstruction	7
1.3.4	Implicit Functions	10
1.4	Overview of Contributions	12
1.5	Impact	16
2	Smooth Signed Distance Surface Reconstruction (SSD)	19
2.1	Introduction	19
2.2	Continuous Formulation	21
2.3	Linearly Parameterized Families	24
2.4	Smooth Basis Functions	25
2.5	Sparsity	27
2.6	Discretization with Discontinuous Gradient	27
2.7	Octree-Based Implementation	30
2.8	Matrix-free Multigrid Solver	31
2.9	Isosurface Extraction	33
2.10	Results	33
2.11	Relation to Poisson Surface Reconstruction	38

2.12	Conclusion	40
3	Smooth Color Map Estimation (SSD-C)	41
3.1	Introduction	41
3.2	Continuous Formulation	43
3.3	Linearly Parameterized Families	44
3.4	Discretization with Discontinuous Function	45
3.5	Octree-Based Implementation	46
3.6	Results	48
3.7	Conclusion	53
4	Generalization of SSD to Volumetric Data (GSSD)	54
4.1	Introduction	54
4.2	Generalized Energy Formulation	57
4.3	Data preprocessing	59
4.4	Implementation	61
4.5	Results	62
4.6	Conclusion	66
5	SSD from Incomplete Surface Normal Data (SSD-Inc)	68
5.1	Introduction	68
5.2	Constrained Energy formulation	71
5.3	Majorize-Minimize (MM) technique	71
5.4	Surrogate Objective	72
5.5	Solution and Algorithm	73
5.6	Implementation	75
5.7	Results	76
5.8	Conclusion	78
6	SSD for Big Data - an Implementation on Octree Duals (SSD-Dual)	79
6.1	Introduction	79
6.2	Dual Formulation	81
6.3	Sufficient Statistics	83

6.4	Analysis of Sufficient Statistics	86
6.5	Implementation	86
6.6	Results	88
6.7	Online Implementation Strategy	90
6.8	Multigrid Method Using Only Sufficient Statistics	92
6.8.1	Expansion of Sufficient Statistics from Leaves to Internal Nodes	93
6.8.2	Proportion of Internal Nodes Among All Octree Nodes	94
6.9	Further Reduction in Memory Footprint of Statistics	94
6.10	Conclusion	96
7	Conclusions and Future Directions	97
	Appendix	100
A.1	Analytical Gradient of Trilinear Interpolation Formula	101
A.2	Continuous Probabilistic Volumetric Modeling (CPVM)	104
A.3	A Point Weighting Scheme to Reveal Sharp Features	107
A.4	Registration of Points to Signed Distance Function	109
	Bibliography	126

★ Parts of this dissertation appeared in proceedings of conferences or in journals. In particular, Chapters 2, 3, and 4 are extended versions of [1], [2], and [3] respectively. Chapter 5 is an extended version of the patent application [4].

List of Tables

2.1	The running time, the peak memory usage, and the number of triangles in the reconstructed surface of the David's head generated with different methods.	35
2.2	Hausdorff distance between real surfaces and reconstructed surfaces. Each row is normalized by the maximum error to provide relative comparison (lower is better).	38
4.1	Quantitative evaluation taken from the Middlebury benchmark [114]. The first value corresponds to accuracy (mm) and the second to completeness (%). All meshes were obtained using GSSD parameters $\lambda_1 = 1$ and $\lambda_2 = 1$ (see (4.4))	66

List of Figures

1.1	Distance visualization framework proposed by from Dick <i>et al.</i> allows surgeons to plan an implant operation ahead of time by interactively showing distances between the 3D model of a patient's bone and the 3D model of the implant in three different modes. Images taken from [5].	2
1.2	Optical 3D data acquisition systems classified based on whether active light sources are exploited or only ambient light of the environment is used by the system to measure the surface of an object or a scene.	3
1.3	An example of multi-view stereo data acquisition pipeline to reconstruct Rome's Colosseum. Bundler software [37] developed by Snavely takes a set of images, image features, and image matches as input, and produces a 3D reconstruction of camera and sparse scene geometry as output. PMVS software [38] developed by Furukawa takes the output of Bundler as input, and produces a dense set of <i>unorganized</i> 3D patches as output. Images taken from [37, 38].	5
1.4	Steps of the global alignment framework proposed by Brown and Rusinkiewicz. From left to right: pairwise correspondences are computed; features are associated with global positions; global feature positions are optimized; each scan is aligned. Images taken from [41].	6
1.5	Surface reconstruction methods are expected to handle different forms of point cloud artifacts. In the figure, these artifacts are depicted on a curve in 2D for the sake of clarity. Images taken from the surface reconstruction survey of Berger <i>et al.</i> [53]. . .	7

1.6	Surfaces reconstructed by various implicit methods from an outlier-ridden point cloud generated by a multi-view stereo algorithm; applied to the reconstruction of a car from outdoor photography. FSSR [64], a <i>local</i> implicit method, shows strong artifacts, whereas <i>global</i> methods (SSD [1], SPR [65], MSF [54]) generate smoother surfaces. Images taken from [54].	8
1.7	An isosurface obtained by running Marching Cubes on the primal octree cells displays cracks (Left). The same isosurface obtained by running Marching Cubes on the dual octree cells is crack-free (Right).	9
1.8	Common types of implicit functions (top-row) used in 3D Vision and Graphics. The functions are illustrated in 2D for clarity. The 1D plots in the bottom row show the behavior of each function near the surface, and away from the surface for the cross-sections shown in red, overlaid in the top-row. True location of the surface is traced in blue in each representation, which corresponds to (from left to right) peaks of the surface density; 0.5 level set of the indicator function; and 0 level set of the signed distance field.	10
1.9	Surface reconstructions of a sphere from 1,000 random samples by global surface reconstruction methods. Poisson (PR) [72] and Screened Poisson (SPR) [65] methods are based on estimating an indicator function of the shape. Smooth Signed Distance (SSD) [1] surface reconstruction method estimates an approximation of signed distance field to the surface. An exception is the recent surface reconstruction method (GR) proposed by Lu <i>et al.</i> [76] which estimates a hybrid of both functions, <i>i.e.</i> , the function behaves like a signed distance function near the surface, and like an indicator function away from the surface. In the figures, the color blue illustrates small RMS (relative to the bounding box diagonal) error, whereas the color red shows areas with large error. Images taken from [76].	11

1.10	Surface reconstructions of King Neptune model by Poisson surface reconstruction (PR) method proposed by Kazhdan <i>et al.</i> [72]; by Smooth Signed Distance (SSD) surface reconstruction method proposed by Calakli and Taubin [1]; and by Screened Poisson (SPR) surface reconstruction method proposed by Kazhdan and Hoppe [65]. The Neptune model is a real (laser) scanner data from the Aim@Shape repository consisting of 2.4M points; half of which were used for reconstruction, and the other half were used for validation by the authors in [65]. The 2D plots in the bottom row show cross-section curves from the reconstructions of the mustache, overlaid with adjacent validation points. Images taken from [65].	16
1.11	Surface reconstructions of an urban scene in Breisach, Germany by the <i>local</i> Floating Scale Surface Reconstruction (FSSR) method proposed by Fuhrmann and Gesele [64]; and by the <i>global</i> Multiscale Fusion Surface Reconstruction (MSF) method proposed by Ummenhofer and Brox [54]. The Breisach dataset consist of 1.5 billion points generated by a multi-view stereo algorithm; applied to the large scale reconstruction of architecture from outdoor photography. Images taken from [54].	17
2.1	A surface reconstructed by the proposed algorithm on an octree of depth 9 (right) from the input point cloud (left).	20
2.2	A 2D oriented point cloud as a sample of a 2D curve represented as a level set of an indicator function (gray=1, white=0). The graph of the smooth signed distance function estimated by our algorithm from the same point cloud.	22
2.3	Vertices of an hexahedral cell	28
2.4	The primal graph of an octree.	30
2.5	An illustration of dual edges of an octree.	31
2.6	Surface reconstructions of David's head with 1 million samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of uniformly distributed points and accurate normals.	34
2.7	Surface reconstructions of David's eye with 187,526 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of uniformly distributed points and <i>inaccurate</i> normals.	35

2.8	Surface reconstructions of a Horse model with 100,000 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of <i>non-uniformly</i> distributed points and accurate normals.	36
2.9	Surface reconstructions of a Chiquita model with 705,375 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of <i>non-uniformly</i> distributed points and <i>inaccurate</i> normals.	37
3.1	Reconstruction of the side of a castle model: The input point cloud (top-left), Surface reconstructed by the proposed algorithm(top-right), Two views from the surface and color map reconstructed by the proposed algorithm (bottom).	42
3.2	Reconstruction of a human frontal view. Top row: the input point cloud (top-left), surface geometry (top-middle), and color map (top-right) reconstructed by the proposed algorithm. Bottom row: 6 images that are used for shape acquisition.	48
3.3	Reconstruction of a building facade. Top row: The input point cloud, and surface geometry reconstructed by the proposed algorithm. Middle row: Two views from the surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 14 images that are used for shape acquisition.	49
3.4	Reconstruction of a brick with cuneiform, Mesopotamian, 859-840 BCE, clay, Overall 35 x 35 x 11 cm, Williams College Museum of Art, Williamstown, MA, Gift of Professor Edgar J. Banks and Dr. John Henry Haynes, Class of 1876, (20.1.33.A). Top row: the input point cloud (left), and surface geometry (right) reconstructed by the proposed algorithm. Middle row: Two views from surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 21 images that are used for shape acquisition.	51
3.5	Reconstruction of a shoe print. Top row: the input point cloud (left), and surface geometry (right) reconstructed by the proposed algorithm. Middle row: Two views from surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 11 images that are used for shape acquisition.	52

4.1	Surface Reconstruction of an urban scene from volumetric likelihood of surface occupancy μ using the proposed <i>generalization</i> of SSD, namely GSSD. The likelihood of surface occupancy μ in the volume is learned by using a multi-view stereo method (CPVM [80, 81]) from aerial imagery captured by flying a helicopter around the city of Providence, RI.	55
4.2	Surface Reconstructions of Temple and Dino models from volumetric likelihood of surface occupancy. Two models on the left are obtained by directly applying Marching Cubes algorithm to extract manually selected isolevels of the surface occupancy as described in [124]. Two models on the right are obtained by the proposed method. The surface occupancy was learned by CPVM [80, 81] from multiple photographs of the models provided with the Middlebury benchmark [114].	56
4.3	Sample frames and camera locations: (a) Middlebury dataset [114], “Full” set of cameras in blue, “Ring” in red; (b) “Downtown” site; (c) “SciLi” site.	62
4.4	Point cloud visualization of “Downtown” (from left to right column wise): point cloud using the top (4.7) 70% of CPVM samples; point cloud using the top (4.7) 10% of CPVM samples; PMVS point cloud.	63
4.5	Reconstructed surfaces from the “Downtown” (top two rows) and “SciLi” (bottom two rows) sites, using PMVS+Poisson (leftmost column) and CPVM+GSSD (mid column). Sample images for the corresponding scenes (rightmost column).	64
4.6	Renderings of 3D models (colored mesh) of the “SciLi” (left) and the “Downtown” (right) sites reconstructed by the framework CPVM+GSSD. Note the high resolution texture on buildings.	65
5.1	Left: Irregularly sampled input point cloud with extremely inaccurate normal vectors. The point cloud is obtained by adding synthetical noise to the directions of the normals of the point set originally taken from [86]. Center: Surface reconstructed with SPR [65], trusting the given normal vectors Right: Surface reconstructed with the proposed method, which simultaneously estimates surface normals.	69
5.2	Curve estimation from synthetically generated points using proposed method. Red dots: Input point samples. Green needles: Input lines of sight. Blue needles: Recovered curve normals.	76

5.3	Reconstruction of “Downtown” site with zoom-in cross-sectional views. Left Column: Input point cloud rendered as points and line of sight vectors. Right column: Reconstruction of the site using the proposed method, rendered as points and recovered normals.	77
6.1	Quadtree based discretizations. Left: Primal graph. Right: Dual graph.	80
6.2	Surface reconstruction of David’s head using the proposed dual discretization (right), and using the original primal discretization (left). Left eye of the model is zoomed in to reveal visual similarities and differences between the results of two implementations. Similar behavior with minor visual differences is expected.	81
6.3	Surface reconstructions of “Downtown” site: using the original SSD-Primal (left), and the proposed SSD-Dual (right). Note similar quality with minor visual differences.	88
6.4	Surface reconstructions of “Brown University” site: using the original SSD-Primal (top), and the proposed SSD-Dual (bottom). Note similar quality with minor visual differences.	89
A.1	A function evaluated at 8 vertices of a regular hexahedral grid. The value at a point (x, y, z) is approximated with trilinear interpolation formula (A.1), and its gradient with (A.3)	101
A.2	(a) Plots of occlusion density ρ and vis as a ray travels through volume. ρ peaks (and visibility drops) as the ray pierces the two walls of the (almost) empty cube. (b) A parametrization of the ray into intervals according to octree cell intersections.	104
A.3	Color-map visualization of a <i>weighted</i> oriented point cloud. The weights are calculated using the proposed scheme (A.14). Color blue shows areas with the lowest weight, whereas the red shows areas with the highest weight. The green shows areas with values in between.	107

Chapter 1

Introduction

1.1 Thesis statement

This thesis provides a general and flexible surface reconstruction framework that uses outputs of popular 3D shape acquisition methodologies (ranging from time-of-flight cameras, 3D laser scanners, structured light based algorithms to multi-view stereo algorithms) as input, and generates detailed polygon mesh surfaces. As part of this thesis, new variational formulations for the problem of reconstructing a watertight surface defined by an implicit equation, from various 3D data representations (oriented points, cross-sectional curves, or volumetric occupancy grids) are introduced. The formulations allow for a number of different discretizations, reduce to scalable least squares problems, and do not require boundary conditions. The resulting algorithms are significantly simpler and easier to implement, and efficient space-adaptive (octree-based) implementations are shown to produce high-quality adaptive manifold polygon meshes comparable with state-of-the-art algorithms. The source code for part of the framework is made publicly available at <http://mesh.brown.edu/ssd>.

1.2 Motivation

Automated estimation of a scene’s geometry and appearance from images is an important research problem with applications ranging from realistic modeling for the film, mapping and gaming industry, to quantitative measures for urban planning, autonomous navigation and various surveillance tasks. The problem of image-based 3D modeling or multi-view stereo has been widely studied in the

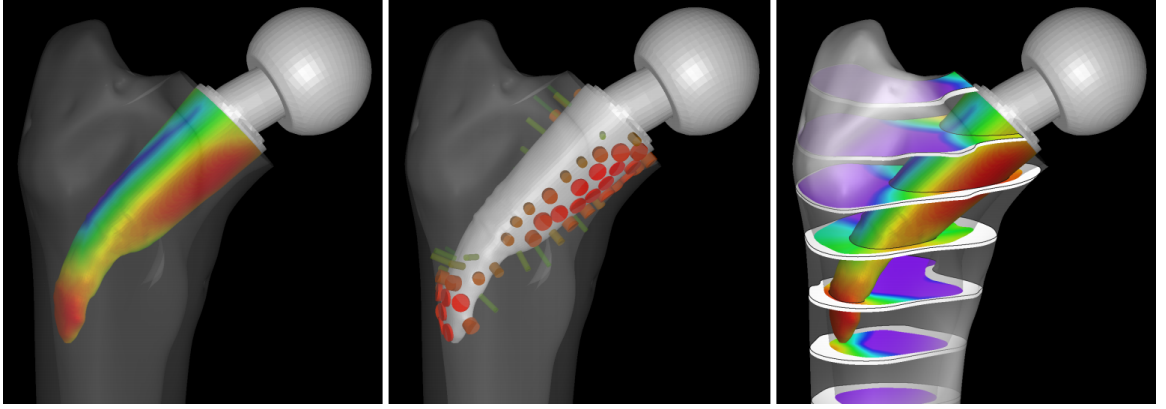


Figure 1.1: Distance visualization framework proposed by Dick *et al.* allows surgeons to plan an implant operation ahead of time by interactively showing distances between the 3D model of a patient’s bone and the 3D model of the implant in three different modes. Images taken from [5].

field of computer vision, computer graphics and computational photography. Most scalable 3D reconstruction techniques have demonstrated results that are sufficient for visualization purposes, but do not necessarily offer a solution to the problem of accurate surface reconstruction.

1.3 Background

A digital 3D model is a numerical representation of a real object. 3D models can be grouped into two broad categories: solid models and surface models. A surface model such as a polygon mesh model provides a continuous 2D description for the surface of the object so that surface attributes (such as coordinates, normals, curvatures) and visual attributes (such as colors, textures, reflectances) can be *explicitly* associated, whereas these attributes are typically associated *implicitly* across the volume in a solid model such as in an indicator function, signed distance field, or density of surface occupancy (see Figure 1.8 for 2D illustrations of these models). The attributes of the surface are then deduced from either by ray-tracing [6–8] the volume from different viewpoints or by contouring [9–12] isosurfaces of the volume at different levels.

The increase in availability and power of graphical processing units (GPUs) has enabled 3D applications to progress. Most of these applications allow users to interact with digital 3D models, *i.e.*, visualize, modify and display attributes of the model in real-time of the interaction which provides a great advantage to digital models over physical ones. 3D digital models are nowadays used in a long list of applications such as in archeology [13–17], fashion design [18], marketing [19], quality

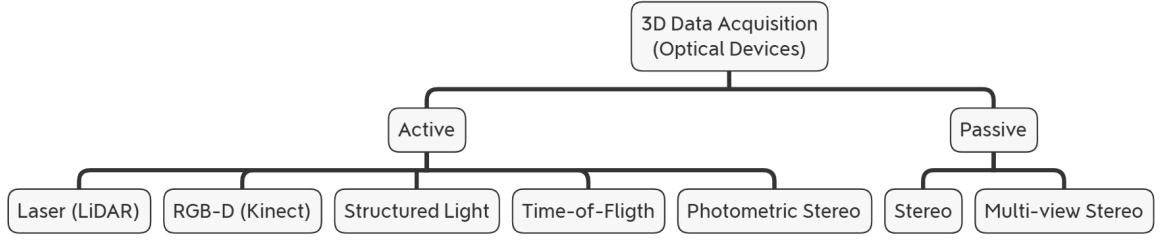


Figure 1.2: Optical 3D data acquisition systems classified based on whether active light sources are exploited or only ambient light of the environment is used by the system to measure the surface of an object or a scene.

control [20], person identification [21], medical education, planning for surgeries [5, 22](Figure 1.1), architectural design, industrial design, reverse engineering [23], entertainment industry (film making, animation [24], gaming), machine manufacturing. 3D digital models are traditionally designed by 3D artists either by drawing its surface using a Computer Aided Design (CAD) software, or by applying boolean operations to simple solid shapes using Constructive Solid Geometry (CSG). Most often, these activities require training, experience and skill. In contrast, 3D digital data can be conveniently obtained by directly measuring the surface of the physical object through 3D scanning or simply by using a photogrammetry software from photographs of the object taken from different view points. More recently, novel 3D shapes and scenes can also be synthesized by machines trained by a large collection of 3D shapes and scenes [25]. With the advancements in artificial intelligence, and the availability of easy-to-use 3D data acquisition pipelines such as inexpensive structure light systems [26, 27] or freely available automated image-based modeling softwares [28, 29], 3D artists are becoming less essential for 3D modeling as the demand for 3D shapes and scenes keeps increasing.

1.3.1 3D Data Acquisition

3D data acquisition involves measuring the geometric and color information of the physical object. The device, the technique, and the scanning procedure vary greatly depending on the size (micro or macro), surface properties (opaque or transparent), or the dynamic nature (still or moving) of the object to be scanned. Since each application comes with a set of different requirements, one should also take into account several other parameters such as accuracy, resolution, speed, flexibility, and cost of the scanning system while choosing one scanner over another.

3D scanning systems can be grouped into two broad categories based on the interaction between

the sensor and the object surface to be scanned. Contact scanners such as coordinate-measuring machines (CMM) or articulated jointed arms physically touch the object surface to get a very accurate sample, albeit rather slowly. On the other hand, non-contact scanners calculate the coordinate of a sample from the interaction of the object with some kind of radiation, as a result, they provide more measurements than contact scanners do for the same time period, albeit with less accuracy. Industrial computed tomography (Industrial-CT), optical sensors, and sensors based on sonar or radar fall into this category. With the increasing availability of digital cameras and handheld devices, optical sensors have gained great popularity in the last decade. Depending on whether active light sources are exploited or only ambient light of the environment is used, these optical devices can also be divided into two sub-categories: active and passive (Figure 1.2). Active methods [30] consists of laser scanners, RGB-D cameras, structured light systems, time-of-flight cameras, and scanners based on photometric stereo method [31]. Computational vision algorithms designed to capture the shape and the texture of an object or a scene in the form of dense point clouds or volumetric likelihood of surface occupancies from calibrated two images (stereo) or multiple images (multi-view stereo) constitute examples of passive methods. Passive stereo devices basically consist of two video cameras operating in sync to capture two video streams of a dynamic scene or a moving object of interest. They are accompanied by an efficient stereo matching algorithm [32] to find corresponding pixels in two views, and to output a depth map stream in real-time of video capture. RGB-D cameras replace one of the cameras with a still projector to actively illuminate the scene with a known static pattern based on non-periodic sequences [33, 34] or pseudo-randomly generated speckles [35, 36] with the aim to simplify the matching process between the views of the projector and the camera, and as a result, to achieve more reliable depth measurements than passive stereo systems.

A multi-view stereo reconstruction system is not necessarily a device but a software that accepts a collection of photographs of an object or a scene captured from different viewpoints to typically output dense point measurements of the object or the scene. A standard multi-view stereo reconstruction software consists of a series of algorithms applied in order by using the outputs of the previous algorithms as input. For example, a publicly available multi-view stereo reconstruction software, namely VisualSFM [28], first uses [39] to extract salient features in each image, and to match them across images, then uses [37] to estimate parameters of each camera such as focal length, position, orientation together with a sparse set of 3D locations of the features, and finally uses [38] to extract dense features in each image, and to match them across images to produce a dense set of

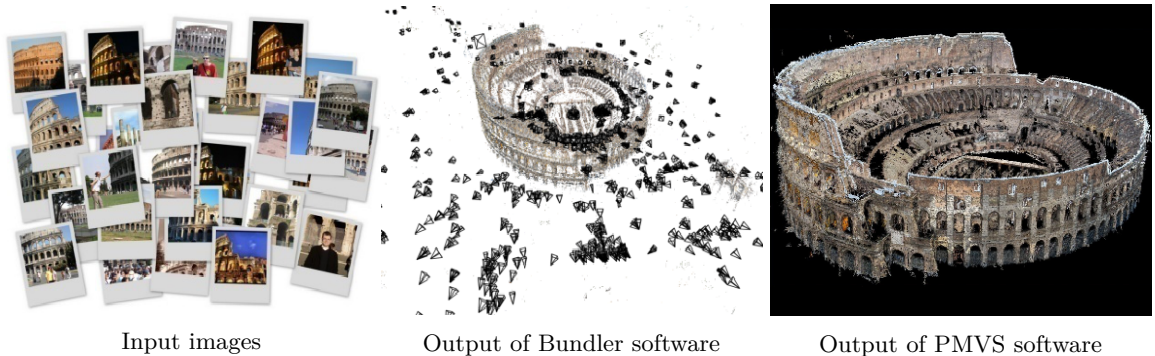


Figure 1.3: An example of multi-view stereo data acquisition pipeline to reconstruct Rome’s Colosseum. Bundler software [37] developed by Snavely takes a set of images, image features, and image matches as input, and produces a 3D reconstruction of camera and sparse scene geometry as output. PMVS software [38] developed by Furukawa takes the output of Bundler as input, and produces a dense set of *unorganized* 3D patches as output. Images taken from [37, 38].

unorganized 3D patches as output. Each patch is represented by a point (x, y, z) for the location, an oriented unit normal (n_x, n_y, n_z) for the orientation, a scale σ for the size, and an (r, g, b) value for the color of the patch. Figure 1.3 summarizes the aforementioned steps for the 3D reconstruction of the Colosseum in Rome, Italy. A literature review focused on multi-view stereo reconstruction algorithms is provided in Section 4.1. For more general reviews of other scanning techniques mentioned in this section, the reader is referred to [30, 40].

1.3.2 Registration of Multiple 3D Scans

Creating 3D models of large and complex objects usually require combining several 3D scans. This is due to sensor scope and resolution, but mainly due to visibility constraints. For example, a laser based range scanner sees the object to be scanned only from a single viewpoint at a time. If the object is small, one can obtain calibrated motion of the scanner or the object by placing the object in a controlled environment such as on motorized turntable so that the relative alignments between range scans are known. On the other hand, if the object is too large to move or to fit in the scanners working volume, scanning has to be performed from different viewpoints and potentially in multiple sessions. As a result, the range scans must be registered into a global coordinate frame after acquisition. For example, it is reported in [14] that more than 600 individual scans were registered to create the model of Michelangelo’s Florentine Pieta, and in another project [13] that more than 1000 individual scans were registered to create only the head model Michelangelo’s David [13].

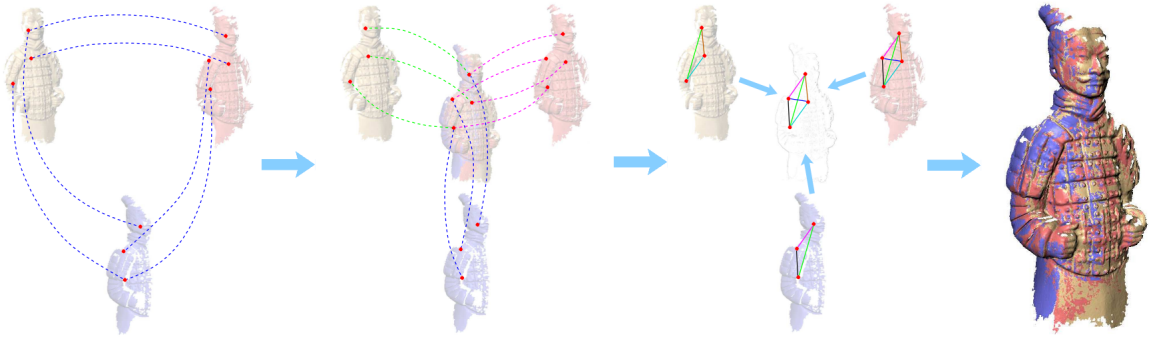


Figure 1.4: Steps of the global alignment framework proposed by Brown and Rusinkiewicz. From left to right: pairwise correspondences are computed; features are associated with global positions; global feature positions are optimized; each scan is aligned. Images taken from [41].

The traditional tool to solve the relative rigid-body alignment problem between two partial scans is Iterative Closest Point (ICP) algorithm [42]. An initial estimate for the relative position and orientation is produced by the user interactively establishing three or more correspondences, or automatically by feature extraction and matching [43]. Then the initial estimate is refined using the ICP algorithm. Each iteration of the algorithm performs a search for matching points in between scans which results in a set of corresponding pairs, followed by a least squares minimization to determine the relative alignment between the pairs in closed form [44]. The algorithm eventually converges to a fixed point. The choice of correspondences both initially and during the iterations of the algorithm is important to prevent the routine from getting stuck at a poor local minimum [45]. In order to constrain the space of potential solutions, several variants of ICP have been proposed over the years replacing the *point-to-point* objective of the original ICP [42] with *point-to-plane* [46], *quadratic* [47], *two-plane* [48, 49], and *symmetric* [50] objectives which significantly improve both the speed of converge and the quality of the final estimate of the relative alignment [50, 51].

Equipped with an ICP variant to align a pair of close scans, one can obtain the final model by recursively pairing and aligning the scans until one big scan is left. Depending on the application, scans can also be aligned incrementally by appending a new scan to the previously aligned scans. However, due to calibration and measurement errors, the final model may not be consistent everywhere. Therefore, ICP is typically employed as part of a global optimization framework [41, 52] to jointly refine alignments for all scans with the aim to distribute overall alignment error evenly among partial scans. Figure 1.4 demonstrates steps of the global alignment framework proposed by Brown and Rusinkiewicz [41].

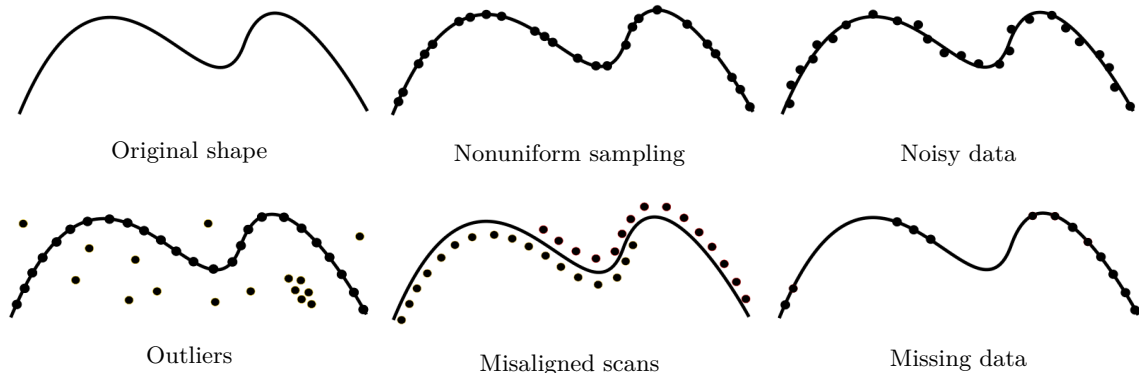


Figure 1.5: Surface reconstruction methods are expected to handle different forms of point cloud artifacts. In the figure, these artifacts are depicted on a curve in 2D for the sake of clarity. Images taken from the surface reconstruction survey of Berger *et al.* [53].

1.3.3 3D Surface Reconstruction

3D acquisition techniques mentioned in Section 1.3.1 generate geometrical data in a discrete format such as in the form of unorganized point samples. A continuous surface representation for the captured data is usually required by most of the applications described in Section 1.3 mainly for compression, storage, transfer, measurement, visualization, modification, animation, and printing purposes. Surface reconstruction methods are expected to handle various types of imperfections present in gathered point cloud data (Figure 1.5) such as non-uniform sampled data due to concave surfaces, missing data due to inaccessible regions, noisy data due to sensor incapacibilities, and distorted data due to misalignment of partial scans. In addition, the size of a point cloud can grow rapidly as new measurements are registered to previous scans. For example, the raw dataset of Michelangelo’s David contains 2 billion points in the Digital Michelangelo Project of Levoy *et al.* [13]. Multi-view stereo reconstructions of large scale urban scenes also result in several hundreds of millions of points even up to a few billions. For example, Citywall dataset provided with the MVE tool [29] contains 256 million points; whereas Breisach dataset [54] contains 1.5 billion points. Thus, numerous surface reconstruction algorithms have been proposed for more than three decades in the graphics and vision community to address the aforementioned challenges. We discuss some of the existing methods focused mainly on implicit methods (based on fitting to the points a 3D scalar function, and then extracting an isosurface) since they are relevant to the variational methods presented in this dissertation. For a more general review, the reader is referred to the surface reconstruction survey of Berger *et al.* [53].

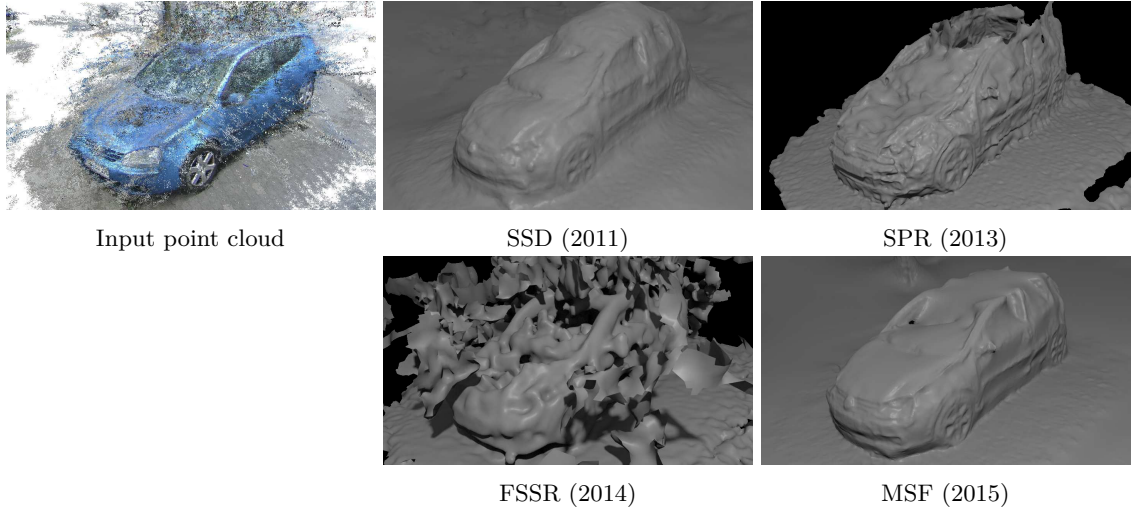


Figure 1.6: Surfaces reconstructed by various implicit methods from an outlier-ridden point cloud generated by a multi-view stereo algorithm; applied to the reconstruction of a car from outdoor photography. FSSR [64], a *local* implicit method, shows strong artifacts, whereas *global* methods (SSD [1], SPR [65], MSF [54]) generate smoother surfaces. Images taken from [54].

Surface reconstruction techniques can be reviewed in two broad categories: explicit methods (based on computational geometry techniques), and implicit methods. **Explicit methods** produce interpolating polygon meshes where all or some of the input points become vertices of the polygons. Crust [55], Ball-pivoting [56], Power Crust [57], Cocone [58], Robust Cocone [59], Wrap [60], and flow complex [61] fall into this category. Most of these methods are based on constructing Voronoi diagrams and Delaunay triangulations of the data points, and are combinatorial in nature. Even though these methods in general come with elegant theoretical guarantees [62], they can be sensitive to noise and imperfections of the input data. A global segmentation approach is proposed by Kolluri *et al.* [63] to handle outliers, and as a result, to improve the reconstruction quality.

On the other hand, **implicit methods** have proven efficient and robust to noise [53]. They estimate a 3D implicit function from the input points, and then extract an isosurface to produce polygon mesh surfaces. Implicit methods can be grouped into two sub-categories: local and global methods. **Local methods** fit data points to local implicit functions (functions with compact support). Then, they blend the local implicit functions to obtain a single function. These methods typically fit a local signed distance function to the data points based on the points' line-of-sight [66], or based on the points' normals [64, 67], or based on local polynomials [68–70]. Even though most of these methods are faster compared to global methods because of using only local information, they

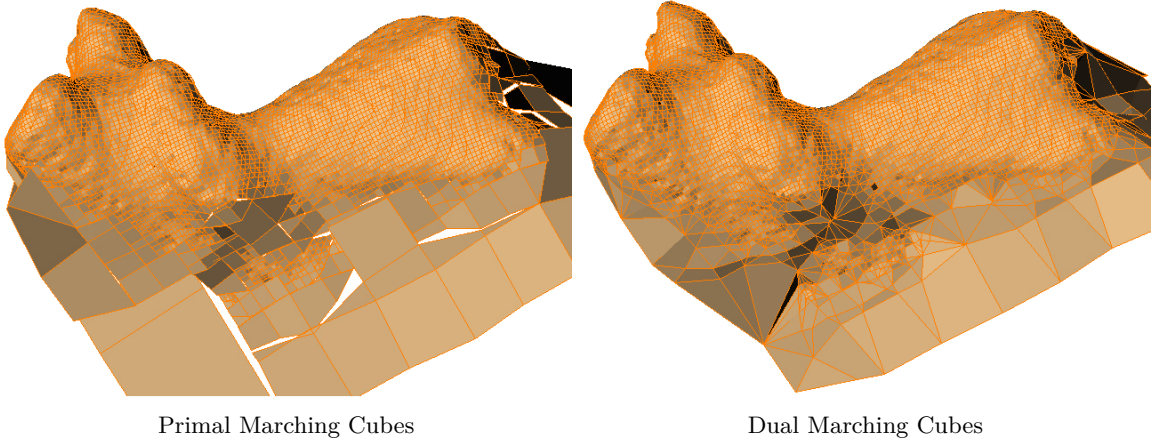


Figure 1.7: An isosurface obtained by running Marching Cubes on the primal octree cells displays cracks (Left). The same isosurface obtained by running Marching Cubes on the dual octree cells is crack-free (Right).

tend to be sensitive to noise and imperfections of the data. On the other hand, **global methods** show more resilience to variance in data (see Figure 1.6) because they consider all data points to estimate the value of the implicit function at a point in the volume. This is typically performed by estimating the implicit function as a solution of a global linear system with respect to a set of basis functions [1, 54, 65, 71–74], or by pointwise evaluation of functions with global support [75, 76]. These methods typically employ a multigrid [77] approach based on a hierarchy of space-adaptive discretization and an iterative solver [78] to obtain the solution efficiently.

Isosurface Extraction: Once the implicit function is obtained, a polygonal approximation of its isolevel 0 for signed distance functions or 0.5 for binary indicator functions (see Figure 2.2 for a 2D illustration of these functions) is typically extracted by a variant of marching cubes algorithms. The original Marching Cubes [9] can be implemented in such a way that it can be applied to any volumetric grid composed of hexahedral cells, and in particular, to regular voxel grids and to octree grids. When it is applied to a scalar field defined on the primal vertices of an octree, a mesh with so-called “cracks” is typically obtained as shown in Figure 1.7(left) due to depth disparity in the octree, *i.e.*, different decisions are made on different sides of a face shared by octree cells at different depths. The primal marching cubes method proposed by Kazhdan *et al.* [12] solves the problem by means of constructing edge trees where the positions of the zero crossings are explicitly defined. Alternatively, since the dual grid of an octree hexahedral grid is composed of cells which can be regarded as

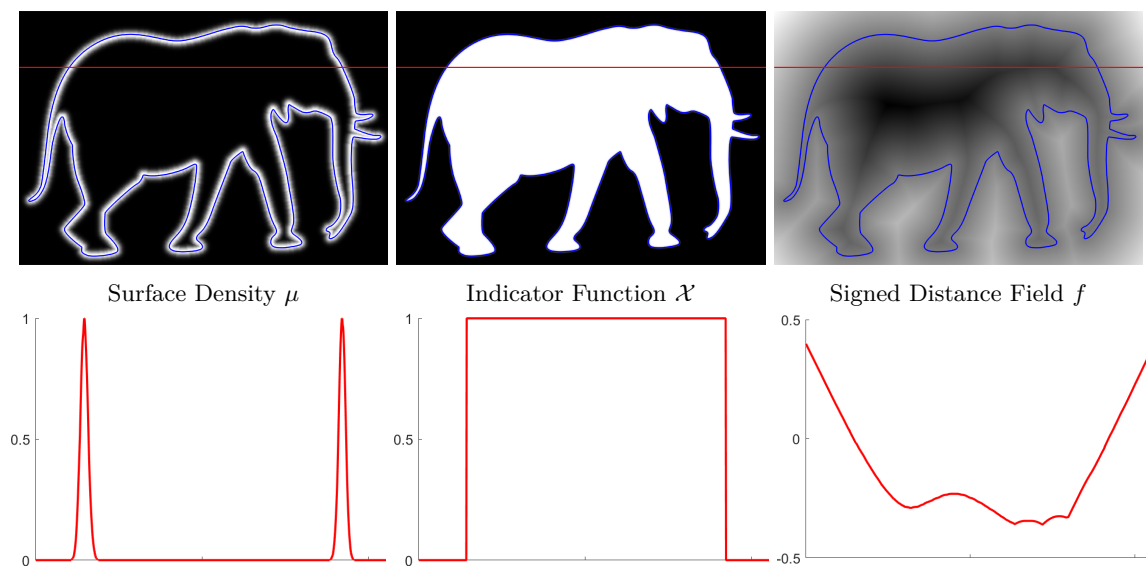


Figure 1.8: Common types of implicit functions (top-row) used in 3D Vision and Graphics. The functions are illustrated in 2D for clarity. The 1D plots in the bottom row show the behavior of each function near the surface, and away from the surface for the cross-sections shown in red, overlaid in the top-row. True location of the surface is traced in blue in each representation, which corresponds to (from left to right) peaks of the surface density; 0.5 level set of the indicator function; and 0 level set of the signed distance field.

hexahedral cells with collapsed edges, the same general implementation of Marching Cubes can be used to instantiate the Dual Marching Cubes (DMC) algorithm [11] to extract watertight meshes as shown in Figure 1.7(right). It is shown in [11] that if vertices of the dual grid are first modified to align with the features of the implicit surface, DMC approximates thin features of the surface more accurately. The precursor of DMC is the Dual Contouring algorithm [10] which allowed for the use of additional information such as the gradient of the implicit function for better vertex placement inside a given octree cell by minimizing a quadratic error function.

1.3.4 Implicit Functions

Signed distance fields (SDFs) and binary indicator functions (BIFs) are widely used in Computer Graphics to implicitly represent 3D shapes and surfaces, and provide volumetric alternatives to polygon mesh surfaces. Volumetric surface densities, on the other hand, are more common in the Computer Vision community to explicitly model uncertain geometry and appearance of a 3D scene which are typically learned from multiple images in a probabilistic framework such as by [79–81]. The differences between these implicit representations are illustrated in Figure 1.8. Smooth signed

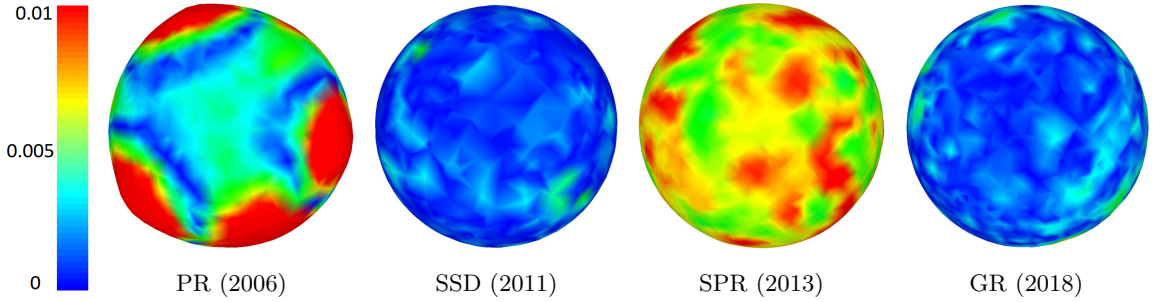


Figure 1.9: Surface reconstructions of a sphere from 1,000 random samples by global surface reconstruction methods. Poisson (PR) [72] and Screened Poisson (SPR) [65] methods are based on estimating an indicator function of the shape. Smooth Signed Distance (SSD) [1] surface reconstruction method estimates an approximation of signed distance field to the surface. An exception is the recent surface reconstruction method (GR) proposed by Lu *et al.* [76] which estimates a hybrid of both functions, *i.e.*, the function behaves like a signed distance function near the surface, and like an indicator function away from the surface. In the figures, the color blue illustrates small RMS (relative to the bounding box diagonal) error, whereas the color red shows areas with large error. Images taken from [76].

distance surface reconstruction methods [1–3, 54] deal with estimations of a smoothed SDF whose zero level set represents the surface to be reconstructed, whereas Poisson surface reconstruction methods [65, 72] deal with estimations of a smoothed BIF whose 0.5 level set defines the surface. An SDF encodes the oriented distance to the surface, whereas an indicator function takes value 1 inside the surface, and 0 outside the surface.

In surface reconstruction contexts, methods designed to estimate an SDF tend to be more accurate than the ones based on estimating a BIF (see Figure 1.9 taken from [76] and Table 1 in [54]), albeit slower (see Table 1 in [65] and Table 1 in [74]). These are expected results as an SDF is a richer function than a BIF, and its gradients are non-zero almost everywhere in the working volume except at the ridges (along the medial axis) of the SDF. The availability of non-zero gradients can be exploited in Simultaneous localization and mapping (SLAM) applications to robustly track handheld RGB-D cameras in cluttered environments by aligning a new depth map to the current prediction of a global SDF, *i.e.*, by using a point-to-SDF objective in ICP (Appendix A.4) instead of the SDF ray-casting approach used in the popular KinectFusion algorithm [82, 83]. The aligned scan can then be fused into the SDF as usual by [66] or by updating the sufficient statistics (Section 6.3) of the SDF, followed by refining the SDF. The variational formulations presented in this dissertation exploit this feature of SDFs (the availability of non-zero gradients) as well, to define data terms which allow for *direct* comparison of the value of the function and its gradient with point samples

and normal samples, respectively, and to define a biharmonic *regularization* term based on finite differences of gradients in a single global optimization framework. Therefore, the resulting methods generate surfaces that faithfully approximate the data while staying more resilient to variance in the data than the methods based on estimating BIFs (see Figure 1.6: SSD [1] and MSF [54] are based on estimating an SDF function to the surface, whereas SPR [65] estimates a BIF function of the surface). On the other hand, it is more straightforward to impose boundary constraints in methods designed to estimate BIFs than the methods for SDFs, *i.e.*, if the surface must stay in a certain part or envelope of the volume, by hardly constraining the value of the function to be *zero* outside of the given envelope [84]. The value of the SDF outside the envelope is not known a-priori before the surface is estimated. Fortunately, its value can still be encouraged to be *non-negative* [85] outside the given envelope for the same purpose.

1.4 Overview of Contributions

Chapter 2 introduces the backbone of the framework (referred to as Smooth Signed Distance Surface Reconstruction method or SSD in short for the rest of the dissertation) which takes an oriented point cloud as input, fits to the points a 3D scalar field (implicit signed distance function) and outputs a polygon mesh approximation of the surface by contouring isolevel zero of the implicit function. The implicit function is formulated as an argument of a novel variational optimization problem based on simple geometric cues and then estimated by solving the problem in an octree-based hybrid finite element-finite difference discretization scheme. As in the Poisson Surface Reconstruction approach, discretizations of the continuous formulation reduce to the solution of sparse linear systems of equations. But rather than forcing the implicit function to approximate the indicator function of the volume bounded by the implicit surface, in our formulation the implicit function is forced to be a smooth approximation of the signed distance function to the surface. Since an indicator function is discontinuous, its gradient does not exist exactly where it needs to be compared with the normal vector data. The smooth signed distance has approximate unit slope in the neighborhood of the data points. As a result, the normal vector data can be incorporated directly into the energy function without implicit function smoothing. In addition, rather than first extending the oriented points to a vector field within the bounding volume, and then approximating the vector field by a gradient field in the least squares sense; here the vector field is constrained to

be the gradient of the implicit function, and a single variational problem is solved directly in one step.

Chapter 3 extends the capability of the framework to estimate pleasing color maps for implicitly defined surfaces from *colored* oriented point clouds which are nowadays obtained using a variety of 3D measurement techniques, in particular, multi-view stereo reconstruction algorithms from multiple 2D images. In the chapter, a novel variational formulation for the problem of reconstructing the color map of a 3D surface defined by an implicit equation from a finite set of *colored* oriented points is proposed. Octree-based finite difference discretization of the formulation reduces the problem to solving a sparse system of linear equations. An implicit function that determines the surface geometry is first estimated from the point positions and orientations by using an implementation of SSD method introduced in Chapter 2, then the proposed method smoothly extrapolates the color information from the points to the surface in the volume using an iterative algorithm based on Gauss-Seidel method. It is shown in the results section that the method is capable of generating pleasing color maps with minimal blurring effects. The proposed method provides an efficient (and scalable) alternative to texture mapping algorithms that depend on image data. The proposed method only uses *color* attributes of points, and avoids most of the difficulties that texture mapping algorithms experience such as photometric discontinuities, seams between face boundaries, and ghosting effects.

Chapter 4 extends the capability of the framework to reconstruct accurate surfaces from probabilistic volumes where the underlying data is not a finite set of points anymore but a volumetric likelihood of surface occupancy. In the chapter, this type of data is acquired by running an implementation of a probabilistic volumetric modeling algorithm from multiple 2D images. However, volumetric data consisting of occupancies can also be obtained in medical imaging applications. A continuous probabilistic volumetric model allows for explicit representation of ambiguities caused by moving objects, reflective surfaces, areas of constant appearance, and self-occlusions; and learns the geometry and appearance of a scene from a calibrated image sequence. The main contribution of this chapter is the *generalization* of the SSD introduced in Chapter 2 to allow reconstruction of surfaces from continuous probabilistic volumetric models using the entire information and ambiguities present in the volume. This is achieved by extending the formulation of SSD to incorporate non-negative continuous surface densities, and also singular measures defined over oriented points. With singular measures, the original formulation of SSD becomes a special case of Generalized-SSD

(GSSD). As in SSD, finite-element/finite-difference discretization of GSSD formulation reduces to solving a sparse system of linear equations. The resulting algorithms are efficient and scalable to large data sets. In the results section of the chapter, a framework that combines probabilistic volumetric modeling with the proposed GSSD is presented to reconstruct surfaces of large scale urban scenes from multi-view aerial imagery. Extensive evaluation shows that the framework equipped with the proposed GSSD produces highly detailed and accurate surfaces from probabilistic volumes.

Chapter 5 extends the capability of the framework to reconstruct surfaces from points clouds where accurate surface normals are not available mainly due to highly irregular sampling of the surface, yet the measuring sensor still provides some form of inside/outside information associated with each point in the dataset. Nearly all optical measuring systems implicitly or explicitly provide the coordinate of the sensor location through calibration information. The vector from a measured point to the sensor location is called the “line-of-sight” associated with the point. A useful property of line of sight vectors is that they are consistently oriented toward the outside (or inside) of the object being scanned similar to the oriented surface normals used by most variational surface reconstruction methods. In general, normals associated with points are estimated from the points using local principal component analysis (PCA) followed by orienting them consistently toward the tensor direction. In regions with missing or irregularly sampled data, or near thin features, normals estimated by PCA get highly unreliable; and as a result, they often lead to erroneous surface reconstruction results. A novel constrained energy formulation is proposed to solve the surface reconstruction problem from a finite set of points and *incomplete* normals associated for each point. We use the term *incomplete* as these vectors are not necessarily surface normals, but instead can be any set of vectors as long as they are consistently oriented towards the outside of the object being scanned, including line-of-sight vectors. We propose an efficient iterative algorithm to minimize the constrained objective based on Majorize-Minimize (MM) technique where in the first step a “feasible” set of normals are estimated from the previous estimate of the implicit function and the *incomplete* normals, and where in the second the step surface estimate is refined using points and “feasible” normals. These two steps are repeated until convergence which is guaranteed by MM. Fortunately, both alternating steps are quite efficient, and the second step reduces to solving SSD introduced in Chapter 2. A dataset that consists of points and *incomplete* normals is referred to as *partially* oriented points in the chapter. A *partially* oriented point set can be as general as any raw 3D data such as those obtained from

3D laser scanners, RGB-D cameras, and inexpensive structured light systems. Since no strong assumption is made on the type of *incomplete* normals, the proposed method, with slight modification, can also be used to reconstruct surfaces from sets of parallel or non-parallel 2D curves which appear in 3D modeling applications based on 2D drawings or are acquired by processing the outputs of medical cross-sectional imaging techniques such as CT, MRI, PET, SPECT, and ultrasonography.

Chapter 6 introduces an alternative discretization for the SSD objective introduced in Chapter 2 to allow for reconstructions of very large scenes consisting of hundreds of millions of oriented points. Similar to SSD, the new discretization is space adaptive and octree based. On the other hand, a dual-graph (which has the octree leaf centroids as vertices) based discretization of the implicit function is preferred over the primal-graph (which has the octree leaf corners as vertices) used in SSD. A primal grid provides a natural way to interpolate a function evaluated at octree corners inside an octree leaf through trilinear interpolation which is exploited to parametrize the implicit function in SSD. However, an octree data structure with implicitly or explicitly defined dual grid is more standard in Computer Science communities. Most libraries come with efficient routines for traversing octree leaves (unfortunately not octree leaf corners) and pairs of leaves that share a common face. Without the availability of trilinear coordinates, the implicit function is defined as a piecewise linear function with the introduction of an auxiliary vector field which ideally serves as the gradient of the implicit function. In order to avoid a constrained optimization problem, a coupling term for these two vector fields is added to the original objective of the SSD. As in SSD, dual discretizations of the continuous formulation reduce to the solution of sparse linear systems of equations. The main contributions of this chapter are the derivation of sufficient (and minimal) statistics that are adequate to solve the linear system with less memory requirements than solving it directly with the point cloud data, and the introduction of an online implementation strategy to accumulate these statistics from point cloud data, *i.e.*, points are never fully loaded or stored in memory. To operate online, an octree is built in a bottom-up fashion rather than the recursive top-down strategy used in SSD. Sufficient statistics are first accumulated at the deepest leaves possible allowed by the octree or desired by the user. Then, the sufficient statistics are expanded from the leaves to the internal nodes, and recursively up to the root. Finally, the linear system is solved efficiently from the top-down using a multigrid approach similar to the one used in SSD.

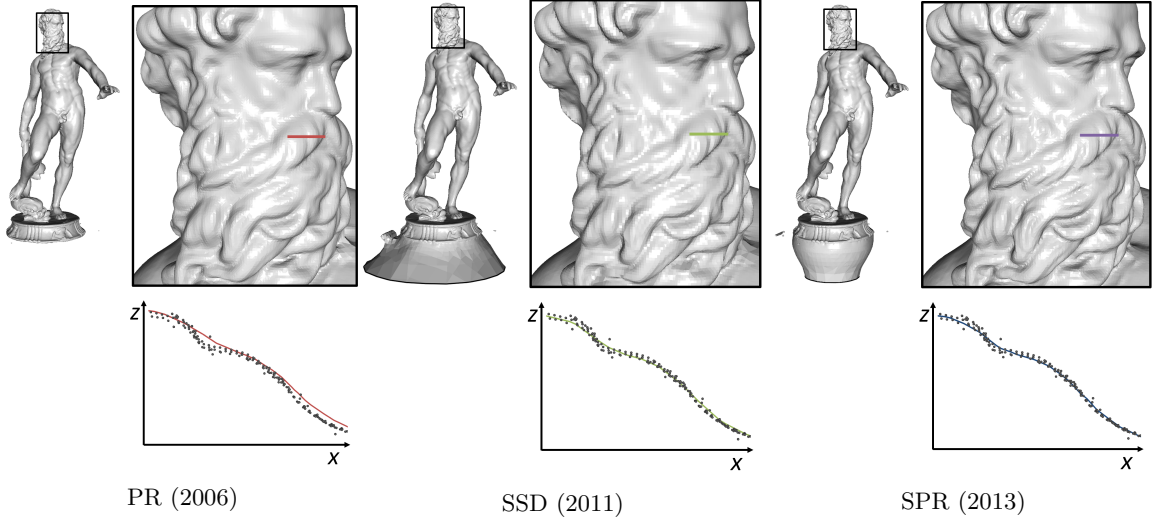


Figure 1.10: Surface reconstructions of King Neptune model by Poisson surface reconstruction (PR) method proposed by Kazhdan *et al.* [72]; by Smooth Signed Distance (SSD) surface reconstruction method proposed by Calakli and Taubin [1]; and by Screened Poisson (SPR) surface reconstruction method proposed by Kazhdan and Hoppe [65]. The Neptune model is a real (laser) scanner data from the Aim@Shape repository consisting of 2.4M points; half of which were used for reconstruction, and the other half were used for validation by the authors in [65]. The 2D plots in the bottom row show cross-section curves from the reconstructions of the mustache, overlaid with adjacent validation points. Images taken from [65].

1.5 Impact

This section provides a brief discussion about the impact of ideas and formulations presented in this dissertation. The variational formulations proposed in different chapters are based on minimizing energy functionals composed of simple geometric constraints to estimate a surface *implicitly* defined as the zero level set of a volumetric signed distance function to the surface. Chapter 2 introduces the energy functional (2.3) which consists of two data terms (2.1), and one regularization term (2.2) to constrain the space of signed distance functions. In detail, the first data term constrains the zero level set of the signed distance function pass through input point locations, while the second data term constrains the gradients of the signed distance function align to input normals at the point locations. The regularization term ensures that the signed distance function is locally smooth everywhere in the volume. All these constraints are satisfied in the least squares sense by solving a single sparse linear system.

Kazhdan and Hoppe [65] incorporated the first data term in (2.1) into the formulation of the

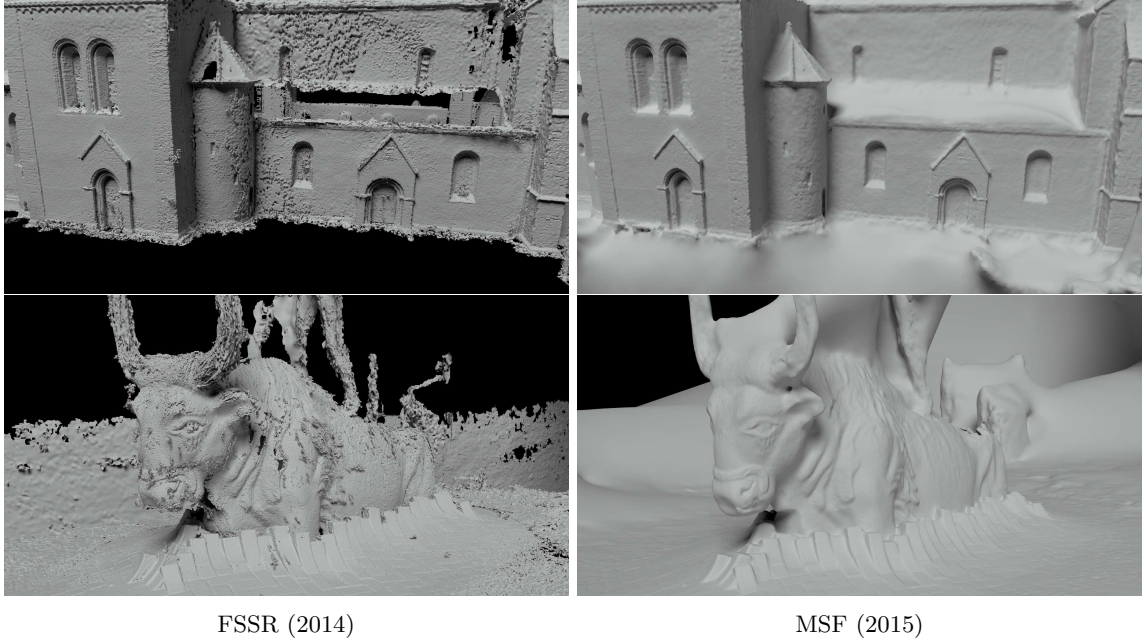


Figure 1.11: Surface reconstructions of an urban scene in Breisach, Germany by the *local* Floating Scale Surface Reconstruction (FSSR) method proposed by Fuhrmann and Goesele [64]; and by the *global* Multiscale Fusion Surface Reconstruction (MSF) method proposed by Ummenhofer and Brox [54]. The Breisach dataset consist of 1.5 billion points generated by a multi-view stereo algorithm; applied to the large scale reconstruction of architecture from outdoor photography. Images taken from [54].

Poisson Surface Reconstruction [72] method as a screening term to enable pointwise positional constraints. The authors showed the extended method, namely *Screened* Poisson Surface Reconstruction [65] method, interpolates point cloud data more accurately (Figure 1.10) than their original Poisson Surface Reconstruction [72] method. Similar to the color map estimation proposed in Chapter 3, the authors also added a *color* option to their software [86] to extrapolate the color values of the input points to the vertices of the reconstructed mesh. The authors later introduced a *general* Adaptive Multi-Grid Solver [74] capable of efficiently solving sparse linear systems that arise from octree-based discretization of energy functionals with pointwise or integrated constraints; which, in surface reconstruction context, typically appear in the formulations of variational methods including the ones presented in this dissertation.

Ummenhofer and Brox [54] replaced the squared norms (ℓ_2 -norms) in the energy functional (2.3) with more robust norms (a smoothed variant of ℓ_1 -norm). Even though the problem does not reduce to solving a single linear system, it is possible to obtain the *global* solution by solving a series of linear systems based on iteratively reweighted least squares approach because the resulting objective is still

convex with the use of ℓ_1 -norms. Similar to the dual graph discretization proposed in Chapter 6, the authors used an octree-based dual grid to discretize the robustified version of the functional. Instead of directly working with the point cloud data, a set of kernel density estimates (KDEs) [87] with less memory footprint than oriented points were first aggregated from data. Then, the minimization is performed only from the KDEs. This allowed the authors to gracefully reconstruct very large points sets. For example, the Breisach dataset with 1.5 billion points was reconstructed in 4 days on a machine with 24 cores (Intel X7460 CPU @2.66GHz, 2008) with a peak memory consumption of 151.8 GB. Figure 1.11 compares the quality of the resulting mesh with the mesh obtained by a local method, namely FSSR [64] which took 3 days to reconstruct the same scene with a memory peak of 164.5 GB on the same machine. A similar approach is proposed in Chapter 6 where the minimization is performed only from *sufficient* statistics which were first accumulated from point cloud data.

Chapter 2

Smooth Signed Distance Surface Reconstruction (SSD)

2.1 Introduction

The problem of reconstructing watertight surfaces from oriented point clouds has received an immense amount of attention since the mid 80's [88]. Oriented point clouds are nowadays usually obtained using optical measuring devices such as laser scanners, inexpensive structured lighting systems, by other computational means such as multi-view stereo reconstruction, and also result from large scale simulations. Dense oriented point clouds have become a pervasive surface representation in Computer Graphics [89].

The main challenges in this problem domain (as illustrated in Figure 1.5) are: how to extrapolate to areas where the sampling is uneven, how to handle missing and noisy data, how to fill holes, and how to develop simple and efficient algorithms which gracefully scale up to very large data sets. As we show in section 2.10 the formulation introduced in this chapter performs particularly well on unevenly sampled data sets. The primary contribution of this chapter is a simple variational formulation of the problem developed from elementary geometric concepts. Various discretizations are proposed based on popular surface representations. The particular hybrid FE/FD discretization described in Section 2.6 results in a simple algorithm which competes in quality and speed with the state-of-the-art methods.

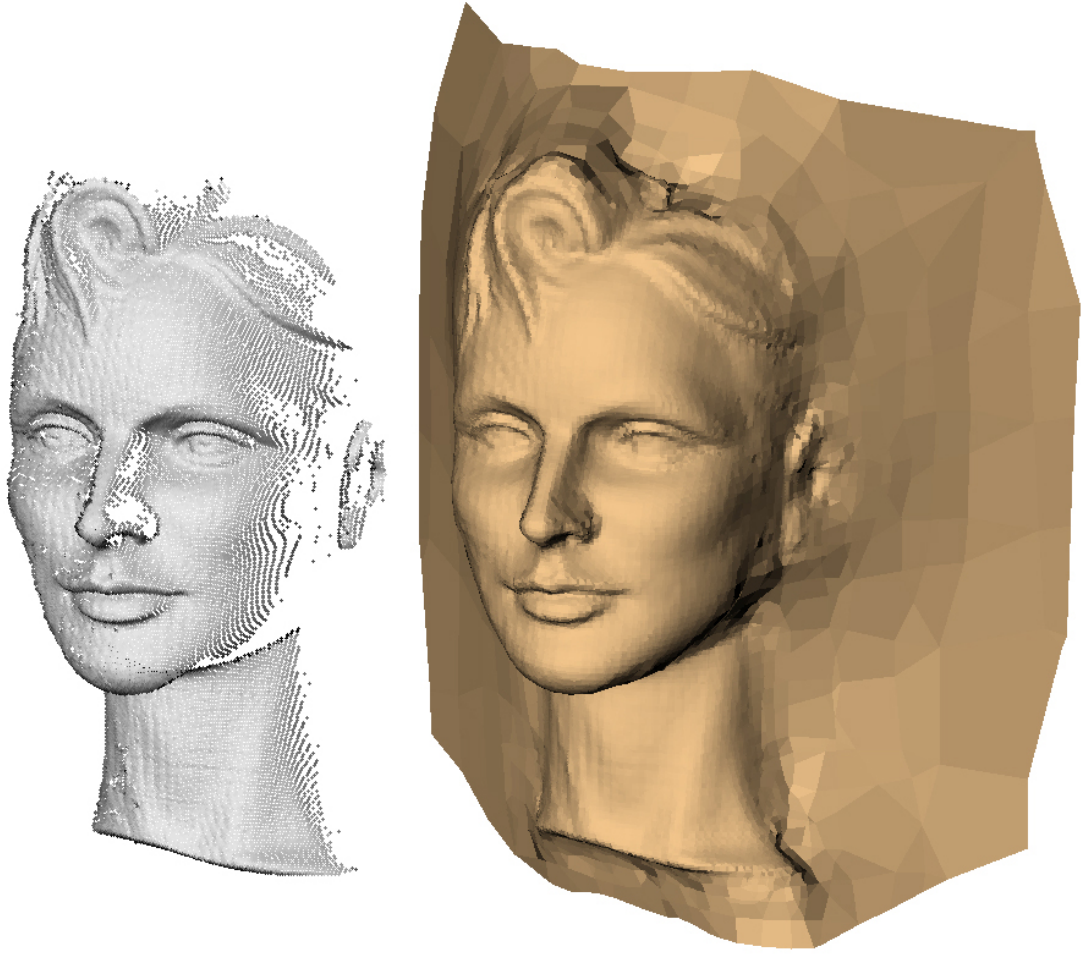


Figure 2.1: A surface reconstructed by the proposed algorithm on an octree of depth 9 (right) from the input point cloud (left).

The prior art in surface reconstruction methods is extensive. Discussions on some of the existing methods are provided in this section and in Section 1.3.3, and we refer the reader to [53] for a survey on recent developments. Despite the long history, the area is still very active. One family of algorithms produces interpolating polygon meshes where all or some of the input points become vertices of the polygons [56]. Most of these algorithms are combinatorial in nature, are based on constructing Voronoi diagrams and Delaunay triangulations of the data points, and many come with guaranteed reconstruction quality [62].

It is often more desirable to produce approximating surfaces rather than meshes supported on the input points. Various representations for the reconstructed surface have been considered. Since the implicit representation guarantees watertight surfaces, most surface reconstruction methods in

this category produce implicit surfaces. Early work in this group, going back to the late 80's, include methods to fit algebraic surfaces to unoriented point clouds [90–94], and PDE-based techniques [95].

More recent methods include [72, 73, 96, 97] which reconstruct a binary indicator function; [66, 67, 98] which estimate a signed distance function; and [69–71, 73, 99–101] based on local function fitting and blending, or moving least squares. Most of these methods reduce the surface reconstruction problem to the solution of a large scale numerical optimization problem, and very often to the solution of a sparse linear system.

Even though the implicit functions produced by some of these methods have explicit analytic forms [69, 71, 73, 99], since polygon meshes are usually needed for visualization or further computation purposes, the estimated implicit function is in the end evaluated on a regular grid of sufficient resolution, and an isosurface is extracted using an algorithm such as Marching Cubes [9], resulting in a polygonal mesh approximation of the desired level set. To reduce the computational and storage complexity, as well as the running time of the isosurface extraction algorithm, many of the proposed methods use octree-based schemes [72, 73, 102, 103], and so do we in some of the algorithms resulting from discretizing our continuous formulation.

Our formulation is most closely related to the Poisson Reconstruction approach [72], which has become a leading contender in this field lately, among other reasons because of the high quality implementations made available by the authors. Streaming [102], parallel [104], and even GPU-based implementations [103] are now available. We devote Section 2.11 of this chapter to establish similarities and differences between the two formulations. As a proof of concept, in Section 2.7 we present an in-core CPU-based implementation. Figure 2.1 shows one example of a surface reconstructed with the proposed method, along with the input point cloud. Note the good behavior near boundaries, holes, and uneven sampling.

Even though we are interested in point clouds and surfaces embedded in three dimensional space, our formulation is valid in spaces of arbitrary dimension. As a result, we use planar examples to illustrate various concepts which extrapolate to higher dimensions.

2.2 Continuous Formulation

We are concerned with the problem of reconstructing a surface S defined by an implicit equation $S = \{x : f(x) = 0\}$, approximating a finite set of oriented points $\mathcal{D} = \{(p_1, n_1), \dots, (p_N, n_N)\}$,

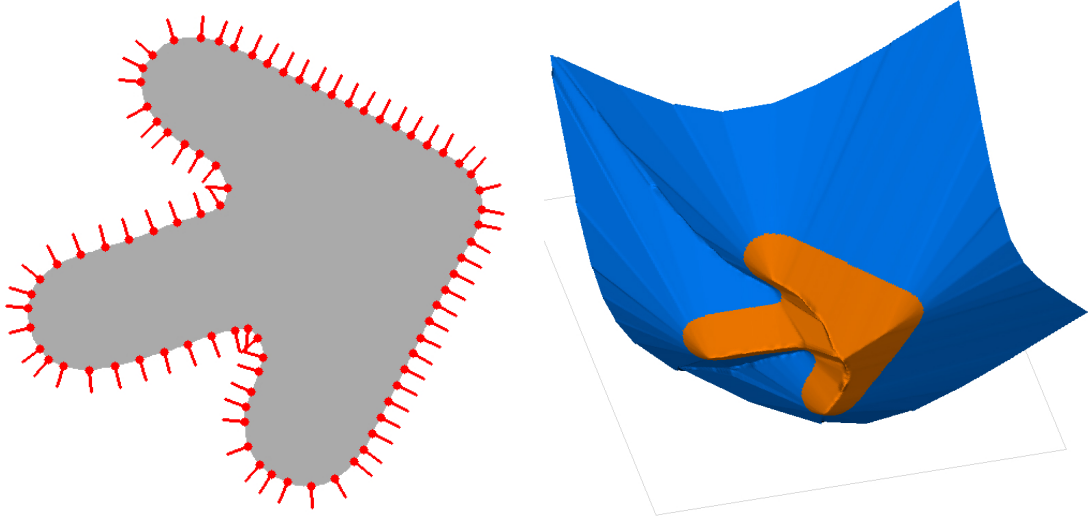


Figure 2.2: A 2D oriented point cloud as a sample of a 2D curve represented as a level set of an indicator function (gray=1, white=0). The graph of the smooth signed distance function estimated by our algorithm from the same point cloud.

where p_i is a surface location sample, and n_i is the corresponding surface normal sample oriented towards the outside of the object. If the function is continuous, then this surface is watertight (closed). Without loss of generality we will assume that $f(x) < 0$ inside and $f(x) > 0$ outside of the object. As a reference, in both FFT Surface Reconstruction [96] and Poisson Reconstruction [72] the implicit function is forced to be the indicator function of the volume bounded by the surface S . This function is identically equal to zero outside, to one inside, and discontinuous exactly on S , as shown in a 2D example in Figure 2.2.

Usually we are only interested in reconstructing the surface S within a bounded volume V , such as a cube containing all the data points, which may result in an open surface at the boundaries of the volume V , particularly when the points only sample one side of the object, there are holes in the data, or more generally if the points are not uniformly distributed along the surface of the solid object. Point clouds produced by 3D scanning systems are viewpoint dependent, and to cover the whole surface of the object several scans need to be registered and merged. We are not covering these topics here. Instead, we assume that multiple scans have already been registered, and we seek a method that works well on regularly sampled as well as unevenly sampled data.

The gradient $\nabla f(x)$ of a function $f(x)$ with first order continuous derivatives is a vector field normal to the level sets of the function, and in particular to the surface S . If the gradient $\nabla f(x)$

does not vanish on the surface S , then S is a manifold surface with no singular points. Without loss of generality, we will further assume that the gradient field on the surface S is unit length, which allows us to directly compare the gradient of the function with the point cloud normal vectors.

Since the points p_i are regarded as samples of the surface S , and the normal vectors as samples of the surface normal at the corresponding points, for an interpolatory scheme the implicit function should satisfy $f(p_i) = 0$ and $\nabla f(p_i) = n_i$ for all the points $i = 1, \dots, N$ in the data set. Interpolating schemes, which require parameterized families of functions with very large numbers of degrees of freedom, are not desirable in the presence of measurement noise. For an approximating scheme, which is what we are after, we require that these two interpolating conditions be satisfied in the least-squares sense. As a result, we consider the problem of minimizing the following data energy

$$\mathcal{E}_{\mathcal{D}}(f) = \lambda_0 \mathcal{E}_{\mathcal{D}_0}(f) + \lambda_1 \mathcal{E}_{\mathcal{D}_1}(f) \quad (2.1)$$

where

$$\mathcal{E}_{\mathcal{D}_0}(f) = \frac{1}{N} \sum_{i=1}^N f(p_i)^2 \quad \text{and} \quad \mathcal{E}_{\mathcal{D}_1}(f) = \frac{1}{N} \sum_{i=1}^N \|\nabla f(p_i) - n_i\|^2$$

and λ_0 and λ_1 are positive constants used to give more or less weight to each one of the two energy terms. The normalization by the number of points is introduced to make these two parameters independent of the total number of points. At this point this normalization has no effect, but plays a role after we add a third regularization term.

Depending on which family $\mathcal{F}$ of functions $f(x)$ is considered as the space of admissible solutions, the problem of minimizing the energy function $\mathcal{E}_{\mathcal{D}}(f)$ of equation (2.1) may or may not be well conditioned. In particular, this energy function does not specify how the function should behave away from the data points. Many parameterized families of functions have parameters highly localized in space, and the energy function of equation (2.1) may not constrain all the parameters. If the unconstrained parameters are arbitrarily set to zero, the estimated surface S may end up containing spurious components located far away from the data points, as we can observe in figures 2.6, 2.7, and 2.9. To address this problem we add the following regularization term to the energy function

$$\mathcal{E}_{\mathcal{R}}(f) = \frac{1}{|V|} \int_V \|Hf(x)\|^2 dx \quad (2.2)$$

where $Hf(x)$ is the Hessian matrix of f , the 3×3 matrix of second order partial derivatives of $f(x)$,

and the norm of the matrix is the Frobenius matrix norm, *i.e.*, the sum of the squares of the nine elements of the matrix. The integral is over the volume V where we are interested in reconstructing the surface, $|V| = \int_V dx$ is the measure of this volume.

The total energy function in the proposed formulation is a weighted average of the data and regularization terms

$$\mathcal{E}(f) = \lambda_0 \mathcal{E}_{\mathcal{D}_0}(f) + \lambda_1 \mathcal{E}_{\mathcal{D}_1}(f) + \lambda_2 \mathcal{E}_{\mathcal{R}}(f) , \quad (2.3)$$

and λ_2 is another positive parameter. Increasing λ_2 with respect to λ_0 and λ_1 produces a smoother result. Note that the three columns of the Hessian matrix $Hf(x)$ are the partial derivatives of the gradient $\nabla f(x)$ with respect to the three cartesian variables:

$$Hf(x) = \begin{bmatrix} \frac{\partial \nabla f(x)}{\partial x_1} & \frac{\partial \nabla f(x)}{\partial x_2} & \frac{\partial \nabla f(x)}{\partial x_3} \end{bmatrix} .$$

As a result, by forcing the square norm of the Hessian matrix to be close to zero, the regularization term makes the gradient of the function almost constant away from the data points. In the vicinity of the data points the data energy terms dominate the total energy, and make the function to approximate the signed distance function. Away from the data points the regularization energy term dominates the total energy and tends to make the gradient vector field $\nabla f(x)$ constant.

Section 2.6 describes the actual discretization that we propose. Sections 2.3, 2.4, and 2.5 are included primarily to motivate the approach.

2.3 Linearly Parameterized Families

In this section we are concerned with the existence and uniqueness of solution for the variational problem defined by the energy function $\mathcal{E}(f)$ of equation (2.3)

$$\mathcal{E}(f) = \frac{\lambda_0}{N} \sum_{i=1}^N f(p_i)^2 + \frac{\lambda_1}{N} \sum_{i=1}^N \|\nabla f(p_i) - n_i\|^2 + \frac{\lambda_2}{|V|} \int_V \|Hf(x)\|^2 dx \quad (2.4)$$

We restrict the analysis here to families of functions linearly parameterized by a finite number of parameters, because in most of these cases the problem has a unique solution, which can be computed by solving a system of linear equations.

By a linearly parameterized family of functions $\mathcal{F}$ we mean a finite dimensional vector space of

functions. Explicitly, a member of this family can be written as a linear combination of certain basis functions

$$f(x) = \sum_{\alpha \in \Lambda} f_{\alpha} \phi_{\alpha}(x) \quad (2.5)$$

where α denotes an index which belongs to a finite set Λ , say with K elements, $\phi_{\alpha}(x)$ is a basis function, and f_{α} is the corresponding coefficient. The K basis functions are chosen in advance, and are not regarded as variables in the problem. After selecting a particular order for the K indices, the function can also be written as an inner product of two K -dimensional column vectors

$$f(x) = \Phi(x)^{\top} F \quad (2.6)$$

where $F = (f_{\alpha} : \alpha \in \Lambda)$ and $\Phi(x) = (\phi_{\alpha}(x) : \alpha \in \Lambda)$ are K -dimensional column vectors.

2.4 Smooth Basis Functions

Even though this is not what we advocate in this chapter, for the sake of developing the ideas, let's first consider the case of basis functions with continuous derivatives up to second order, so that the gradient exists and the second order derivatives are integrable in V . Radial basis functions [71], compactly supported basis functions [69, 99], polynomials [90, 91, 94, 105] trigonometric polynomials [96], wavelets [73], B-splines [72], and many other popular families of functions are included in this category.

Restricting the energy function $\mathcal{E}(f)$ of equation (2.3) to one of these families results in a positive-semidefinite, non-homogeneous, quadratic function $F^{\top} A F - 2 b^{\top} F + c$ in the K -dimensional parameter vector F . Except for very singular configurations, the matrix A is usually positive definite, and the resulting finite dimensional minimization problem has a unique minimum. The global minimum is determined by solving the system of linear equations $A F = b$. To complete the process we need to compute the coefficients $A_{\alpha\beta}$ and b_{α} of the matrix A and vector b , and then select an appropriate algorithm to solve the linear equations.

Note that the three terms of the energy function $\mathcal{E}(f)$ contribute to the matrix A , but since the first data term and regularization terms are homogeneous quadratic functions of F , only the second data term contributes to the vector b .

Keeping in mind the parameterization described by equation (2.6), the contribution of the first

data term $\mathcal{E}_{\mathcal{D}_0}(f)$ to the matrix A is

$$A_{\alpha\beta}^0 = \frac{1}{N} \sum_{i=1}^N \phi_{\alpha}(p_i) \phi_{\beta}(p_i) \quad (2.7)$$

Since

$$\|\nabla f(p_i) - n_i\|^2 = \|\nabla f(p_i)\|^2 - 2n_i^{\top} \nabla f(p_i) + \|n_i\|^2$$

the contribution of the second data term $\mathcal{E}_{\mathcal{D}_1}(f)$ to the matrix A is

$$A_{\alpha\beta}^1 = \frac{1}{N} \sum_{i=1}^N \nabla \phi_{\alpha}(p_i)^{\top} \nabla \phi_{\beta}(p_i) \quad (2.8)$$

and to the vector b is

$$b_{\alpha}^1 = \frac{1}{N} \sum_{i=1}^N n_i^{\top} \nabla \phi_{\alpha}(p_i) \quad (2.9)$$

The contribution of the second data term to the constant c is equal to λ_1 , because the normal vectors are assumed to be unit length, but this is irrelevant since an additive constant in the energy function does not play any role in the optimization.

The contribution of the regularization term $\mathcal{E}_{\mathcal{R}}(f)$ to the matrix A is

$$A_{\alpha\beta}^2 = \frac{1}{|V|} \int_V \langle H\phi_{\alpha}(x), H\phi_{\beta}(x) \rangle dx \quad (2.10)$$

where $\langle \cdot, \cdot \rangle$ is the inner product associated with the Frobenius norm

$$\langle H\phi_{\alpha}(x), H\phi_{\beta}(x) \rangle = \sum_{i=1}^3 \sum_{j=1}^3 \frac{\partial^2 \phi_{\alpha}(x)}{\partial x_i \partial x_j} \frac{\partial^2 \phi_{\beta}(x)}{\partial x_i \partial x_j}$$

Finally, we have $A = \lambda_0 A^0 + \lambda_1 A^1 + \lambda_2 A^2$ and $b = \lambda_1 b^1$. While accumulating the contributions A^0 , A^1 and b^1 of the two data terms requires the evaluation of the basis functions and their derivatives on all the data points, for fixed basis functions which are known in analytic form, the contribution A^2 of the regularization term can be precomputed and saved in a static table, which can be hardcoded in the implementation. If the basis functions are constructed as functions of the data set, then the integrals (2.10) have to be evaluated during the program execution. Numerical approximations of these integrals are usually acceptable, although for many of the linear families mentioned above, these integrals can be evaluated analytically in closed form.

2.5 Sparsity

If the number of parameters K is small, as in the case of polynomial basis functions, solving the linear system $AF = b$ is straightforward. One of many direct linear solvers for matrices represented as arrays can be used, including the QR method. However, in most surface reconstruction problems, a large number of degrees of freedom, typically $O(N)$, is required for acceptable quality results. For example, the radial basis functions used in [71] and trigonometric polynomials [96] are non-zero almost everywhere, and result in a dense matrix A . This is not only a problem in terms of storage, but also in terms of speed, since out-of-core storage of matrix and vector may be required along with simple and slow iterative schemes. On the other hand, the matrix A is sparse when the basis functions are compactly supported, so that the support of only a small number of basis functions overlaps at any given point in the volume V . This property also results in many of the integrals shown in equation (2.10) associated with the regularization term being zero: if the supports of ϕ_α and ϕ_β do not intersect, then the integral $A_{\alpha\beta}^2$ is equal to zero.

For example, the linear families used in the following methods [69, 72, 73, 99], which all use compactly supported basis functions, would be appropriate for this approach. However, since computing the integrals associated with the regularization term can result in significant implementation complexity, we are not advocating for any of these families. Instead, we propose a hybrid FE/FD discretization with discontinuous gradient, which results in a much simpler regularization term.

2.6 Discretization with Discontinuous Gradient

The approach described in the previous section can be extended to the case of basis functions with derivatives continuous only up to first order, and with second order derivatives which are integrable in V in the sense of generalized functions. As long as the discretized function $f(x)$, gradient $\nabla f(x)$, the Hessian $Hf(x)$ can be written as homogeneous linear functions of the same parameters F , the problem still reduces to the solution of linear equations $AF = b$. In fact, even independent discretizations for $f(x)$, $\nabla f(x)$ and $Hf(x)$ can be used, again, as long as all these expressions are homogeneous linear functions of the same parameters F . We propose in this section one such hybrid discretization, where a finite element discretization is used for the function $f(x)$, but finite differences discretizations are used for gradient $\nabla f(x)$ and the Hessian $Hf(x)$. This discretization turns out to be particularly simple to implement, is applicable to adaptive grids, and produces results of high

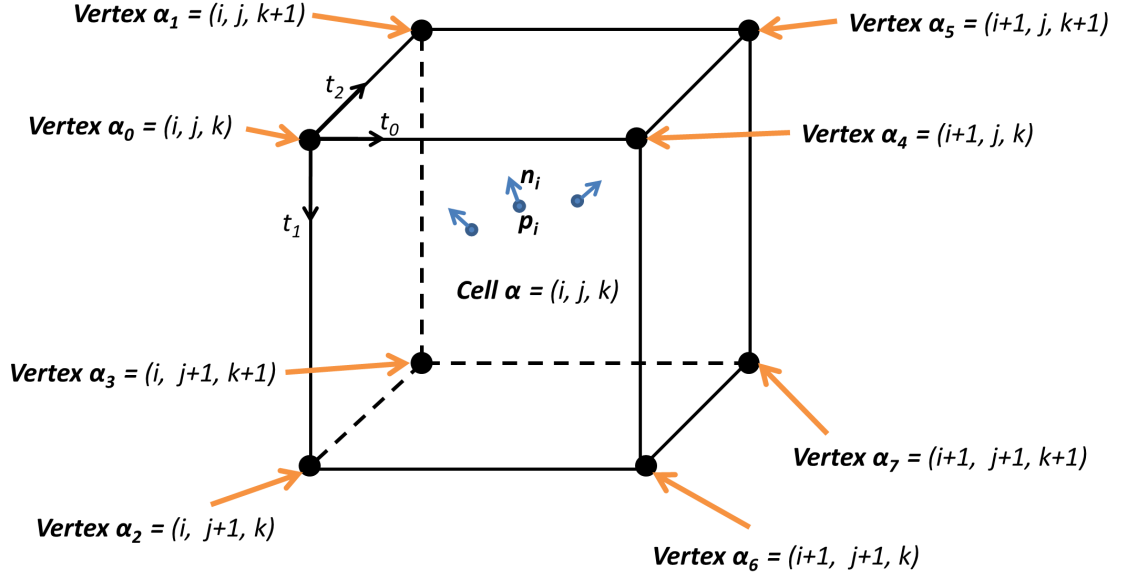


Figure 2.3: Vertices of an hexahedral cell

quality at high speed as presented in Section 2.10.

To simplify the notation, we first consider a regular hexahedral grid partition of space. We assume that the volume V is a unit cube $V = [0, 1] \times [0, 1] \times [0, 1]$, and that each of the three axes is split into M equally long segments, resulting in M^3 hexahedral cells and $(M + 1)^3$ vertices. In the general case a bounding box for the data points has to be computed first, perhaps expanded by a certain amount to prevent boundary-related artifacts, and all the coordinates need to be scaled. If the scaling is anisotropic in the three coordinates, then the normal vector coordinates need to be scaled as well and then renormalized to unit length.

In this discretization, the indices $\alpha \in \Lambda$ are the grid vertex multi-indices $\alpha = (i, j, k)$, for $0 \leq i, j, k \leq M$. The coefficients f_α are then the values of the function $f(x)$ at the grid vertices $p_\alpha = (i/M, j/M, k/M)$. Since the result of this computation is a scalar field defined on the vertices of a regular hexahedral grid, the construction of an output isosurface using Marching Cubes [9] does not require any further function evaluations. The grid cells C_α are also indexed by multi-indices $\alpha = (i, j, k)$, except that in this case $0 \leq i, j, k < M$. The indices of the eight vertices of a cell C_α are depicted in Figure 2.3.

We use finite differences discretizations for the gradient and Hessian. Within each cell C_α we use

trilinear interpolation as the discretization of the function $f(x)$. This is a finite element discretization. Determining which cell C_α each point p_i belongs to reduces to quantization. If $p_i \in C_\alpha$ then

$$f(p_i) \approx \sum_{h=0}^7 w_h f_{\alpha_h} \quad (2.11)$$

where $w_0, \dots, w_7$ are the trilinear coordinates of p_i . Within each cell C_α we use a constant average of finite differences discretization for the gradient $\nabla f(x)$

$$\nabla_\alpha f \triangleq \frac{1}{4\Delta_\alpha} \begin{pmatrix} f_{\alpha_4} - f_{\alpha_0} + f_{\alpha_5} - f_{\alpha_1} + f_{\alpha_6} - f_{\alpha_2} + f_{\alpha_7} - f_{\alpha_3} \\ f_{\alpha_2} - f_{\alpha_0} + f_{\alpha_3} - f_{\alpha_1} + f_{\alpha_6} - f_{\alpha_4} + f_{\alpha_7} - f_{\alpha_5} \\ f_{\alpha_1} - f_{\alpha_0} + f_{\alpha_3} - f_{\alpha_2} + f_{\alpha_5} - f_{\alpha_4} + f_{\alpha_7} - f_{\alpha_6} \end{pmatrix} \quad (2.12)$$

where Δ_α is the side length of cell C_α . If the dimensions of the bounding box are different along each of the three coordinate axes, then different lengths must be used to normalize each of the three coordinates of $\nabla_\alpha f$. If $p_i \in C_\alpha$ then

$$\nabla f(p_i) \approx \nabla_\alpha f \quad (2.13)$$

The piecewise constant gradient leads to zero second derivatives within each cell. In this case the square norm of the Hessian matrix $Hf(x)$ is a generalized function supported on the square faces shared by neighboring voxels (as Dirac's δ "function"), and the integral over the volume V reduces to a finite sum over the faces. Hence, we only need an approximation to $\|Hf(x)\|^2$ at the faces shared by cells. If C_α and C_β are two such cells that share a face, then we use finite differences of discrete gradients

$$H_{\alpha\beta}f \triangleq \frac{1}{\Delta_{\alpha\beta}} (\nabla_\alpha f - \nabla_\beta f) \quad (2.14)$$

where $\Delta_{\alpha\beta}$ is the Euclidean distance between the centers of the cells. As a result, the integral in the regularization term becomes a sum over all the neighboring cell pairs

$$\mathcal{E}_R(f) \approx \frac{1}{|V|} \sum_{(\alpha,\beta)} |V|_{\alpha\beta} \|H_{\alpha\beta}f\|^2 \quad (2.15)$$

where $|V|_{\alpha\beta}$ is the area of the face shared by cells C_α and C_β , and $|V| = \sum_{(\alpha,\beta)} |V|_{\alpha\beta}$.

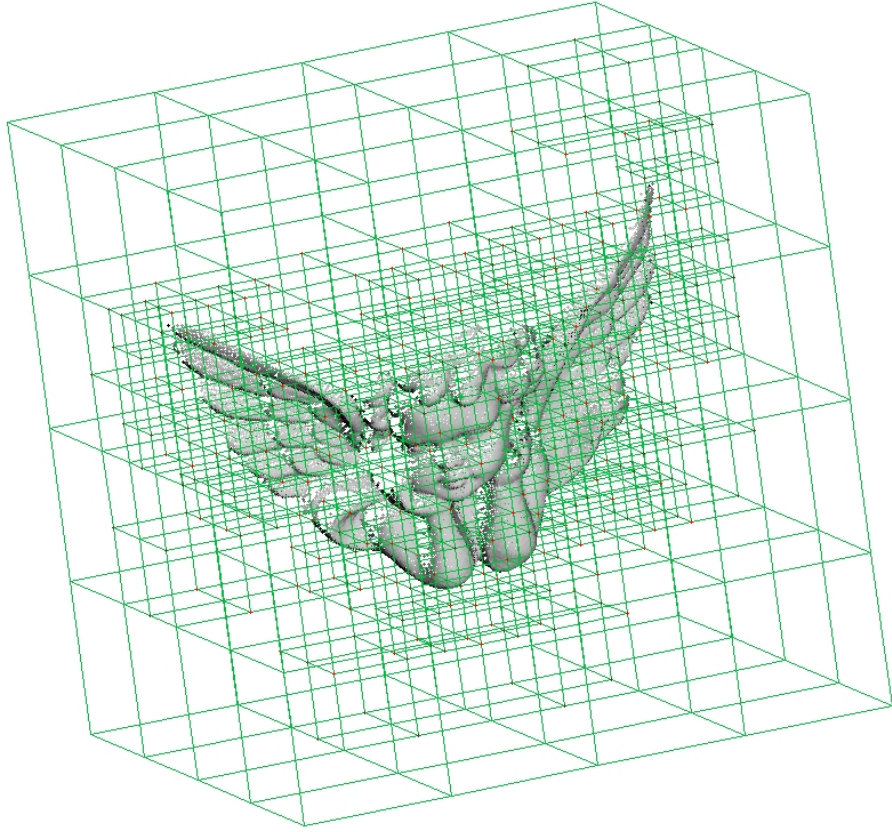


Figure 2.4: The primal graph of an octree.

2.7 Octree-Based Implementation

As in many of the prior works cited above, a regular grid formulation results in high storage requirements and slow algorithms. Also, as in many of the cited works, we adaptively construct an octree based on the location of the data points. In the recursive subdivision algorithm, a cell is split if the number of points contained is larger than a prespecified value, and the cell depth does not exceed another prespecified value. The data set is recursively partitioned into subsets of points $\mathcal{D}_\alpha$ associated with cells C_α . For octrees the multi-indices $\alpha = (L, i, j, k)$ must be augmented with a level (equivalently depth) value L , both for cells and vertices, where $0 \leq L \leq L_{\text{MAX}}$. The correspondence between indices α and cells is 1-1, but not for vertices, since (L, i, j, k) and $(L + 1, 2i, 2j, 2k)$ refer to the same vertex. However, we use these relations in our implementation, along with hashing algorithms [106], to efficiently traverse the primal and dual graphs of an octree. Figure 2.4 shows an oriented point cloud and the primal graph of an octree constructed in this manner.

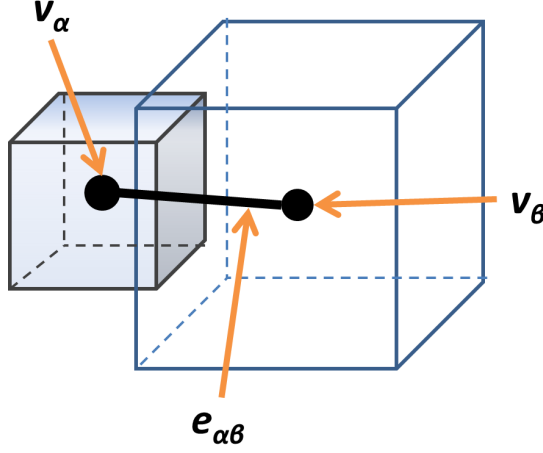


Figure 2.5: An illustration of dual edges of an octree.

The finite-differences discretization presented in the previous section for regular hexahedral grids extends in a natural way to octree grids, without any changes. There is one function value f_α associated with each primal vertex of the octree. As a result, within each hexahedral cell we use trilinear interpolation to discretize the function $f(x)$, and the constant gradient $\nabla_\alpha f$ of equation (2.12). Finally, we use the discretization of the Hessian $Hf(x)$ of equation (2.14), where the dual edge weight $|V|_{\alpha\beta}$ is now the area of the common face. Figure 2.5 illustrates this concept.

2.8 Matrix-free Multigrid Solver

The formulation described in previous section reduces to the solution of a linear system $AF = b$. The data set $\mathcal{D}$ is partitioned into disjoint sets $\mathcal{D}_\alpha$ during the recursive subdivision process used to construct the octree. The elements of the matrix A and vector b are accumulated by traversing the list of octree leaf cells. For each point in a non-empty cell C_α , new terms are added to $A_{\alpha_i\alpha_j}$ and to b_{α_i} , where α_i and α_j are indices of the eight vertices of the cell C_α . To accumulate the contribution of the regularization term the octree dual graph is traversed. For each dual edge connecting cells C_α and C_β , terms are added to the elements $A_{\alpha_i\beta_j}$ of the matrix A , where α_i is an index of a vertex of cell C_α , and β_j is an index of a vertex of cell C_β . After all the coefficients are accumulated, for reasonably sized problems one can use a direct solver.

For large scale problems, an iterative solver is typically preferred over direct solvers since the number of non-zero entries in sparse A can grow very rapidly. Most iterative methods designed

to solve linear systems do not require matrices built explicitly, *i.e.*, the result of the matrix vector product $A\theta^{(k)}$ is only needed for a given vector $\theta^{(k)}$ to perform an iteration k [78]. We accumulate the $K \times 1$ result vector of the product $AF^{(k)}$ and the $K \times 1$ right hand side vector b through following sums:

$$\begin{aligned}
AF^{(k)} &= \frac{\lambda_0}{N} \sum_{\alpha} \sum_{i \in \alpha^*} \Phi_{\alpha}(p_i) \left\{ \Phi_{\alpha}(p_i)^{\top} F^{(k)} \right\} \\
&+ \frac{\lambda_1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} \nabla \Phi_{\alpha}(p_i) \left\{ \nabla \Phi_{\alpha}(p_i)^{\top} F^{(k)} \right\} \\
&+ \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} \nabla \Phi_{\alpha}(x_{\alpha}) \left\{ \nabla \Phi_{\alpha}(x_{\alpha})^{\top} F^{(k)} - \nabla \Phi_{\beta}(x_{\beta})^{\top} F^{(k)} \right\} w_{\alpha\beta} \\
&- \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} \nabla \Phi_{\beta}(x_{\beta}) \left\{ \nabla \Phi_{\alpha}(x_{\alpha})^{\top} F^{(k)} - \nabla \Phi_{\beta}(x_{\beta})^{\top} F^{(k)} \right\} w_{\alpha\beta} \\
b &= \frac{\lambda_1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} \nabla \Phi_{\alpha}(p_i) n_i
\end{aligned} \tag{2.16}$$

where the first two terms are contributions from the data terms, and the last two are contributions from the regularization term after expanding the finite difference. The set α^* represents the indices of the points enclosed in octree leaf cell C_{α} , *i.e.*, $\alpha^* = \{i : p_i \in \mathcal{D}_{\alpha}\}$. $\Phi_{\alpha}(x)$ is a $K \times 1$ sparse vector with 8 non-zero entries (trilinear coordinates of x in C_{α}), and $\nabla \Phi_{\alpha}(x)$ is a $K \times 3$ sparse gradient matrix with 24 non-zero entries. x_{α} denotes the centroid of leaf cell C_{α} . Evaluating the analytical form $\nabla \Phi_{\alpha}(x)$ given in [A.1, equation (A.3)] at the centroid x_{α} gives the constant average $\nabla \Phi_{\alpha}(x_{\alpha}) = \nabla_{\alpha} \Phi$ of finite difference discretization as in equation (2.12). And the finite difference of constant average gradients gives an approximation to the hessian as in equation (2.14). The sums associated with the regularization term are over the pairs of octree leaves C_{α} and C_{β} that share a common face. The associated smoothing weight

$$w_{\alpha\beta} = \frac{A_{\alpha\beta}}{\Delta_{\alpha\beta}}$$

where $A_{\alpha\beta}$ is the area of the common face and $\Delta_{\alpha\beta}$ is the Euclidean distance between the two leaves. $W = \sum_{\alpha\beta} w_{\alpha\beta}$ is normalization factor so that $w_{\alpha\beta}$'s sum up to one.

Since octrees inherently have multiresolution structure, we follow an iterative cascading multigrid approach. We start by solving the problem on a much coarser level than desired using a conjugate gradient solver. Then use it to initialize the solution at the next level, which is then refined with the conjugate gradient solver.

2.9 Isosurface Extraction

Once the coefficients f_α are determined, we are faced with the final problem of constructing a polygonal approximation of the isolevel zero of the discretized implicit function. Marching Cubes [9] can be implemented in such a way that it can be applied to any volumetric grid composed of hexahedral cells, and in particular, to regular voxel grids and to octree grids. When it is applied to a scalar field defined on the primal vertices of an octree, a mesh with so-called “cracks” typically arises (Figure 1.7). Since the dual grid of an octree hexahedral grid is composed of cells which can be regarded as hexahedral cells with collapsed edges, we use the same general implementation of Marching Cubes to instantiate the Dual Marching Cubes algorithm [11]. Dual Marching Cubes requires values of the implicit function at the vertices of the dual graph, *i.e.*, at the centroids of the primal octree hexahedral cells. The results of the optimization algorithm for the proposed finite differences discretization is defined on the vertices of the primal graph. An implicit function value is computed for the centroid of each primal cell by averaging the values associated with the eight cell corners.

2.10 Results

We compare the proposed Smooth Signed Distance (SSD) surface reconstruction method with Multi-Level Partition of Unity Implicits (MPU) [69], Poisson Surface Reconstruction (Poisson) [72], and Streaming Surface Reconstruction Using Wavelets (D4 Wavelets) [73]. We present the reconstruction results in terms of speed, memory, efficiency, and accuracy. We show that, in terms of the quality of reconstruction, our method outperforms these other methods even on data sets associated with extreme complications: non-uniform sampling, sensor noise, and missing data. We apply our algorithm to point clouds retrieved either from a single partial scan, or from multiple scans that have been aligned with respect to a common reference frame. We assume that each point is associated with a surface normal vector. In the absence of this information, we efficiently estimate the normals from partially triangulated scans, and determine their orientations from the viewpoint direction to the scanner. If this process introduces significant errors, one can alternatively estimate a point sample’s normal vector from the positions of the local neighbors using local fitting [67].

Our initial test case is the head of Michelangelo’s David raw dataset of 1 million samples. This dataset is assembled from many range images acquired by a laser scanner. Since the points are

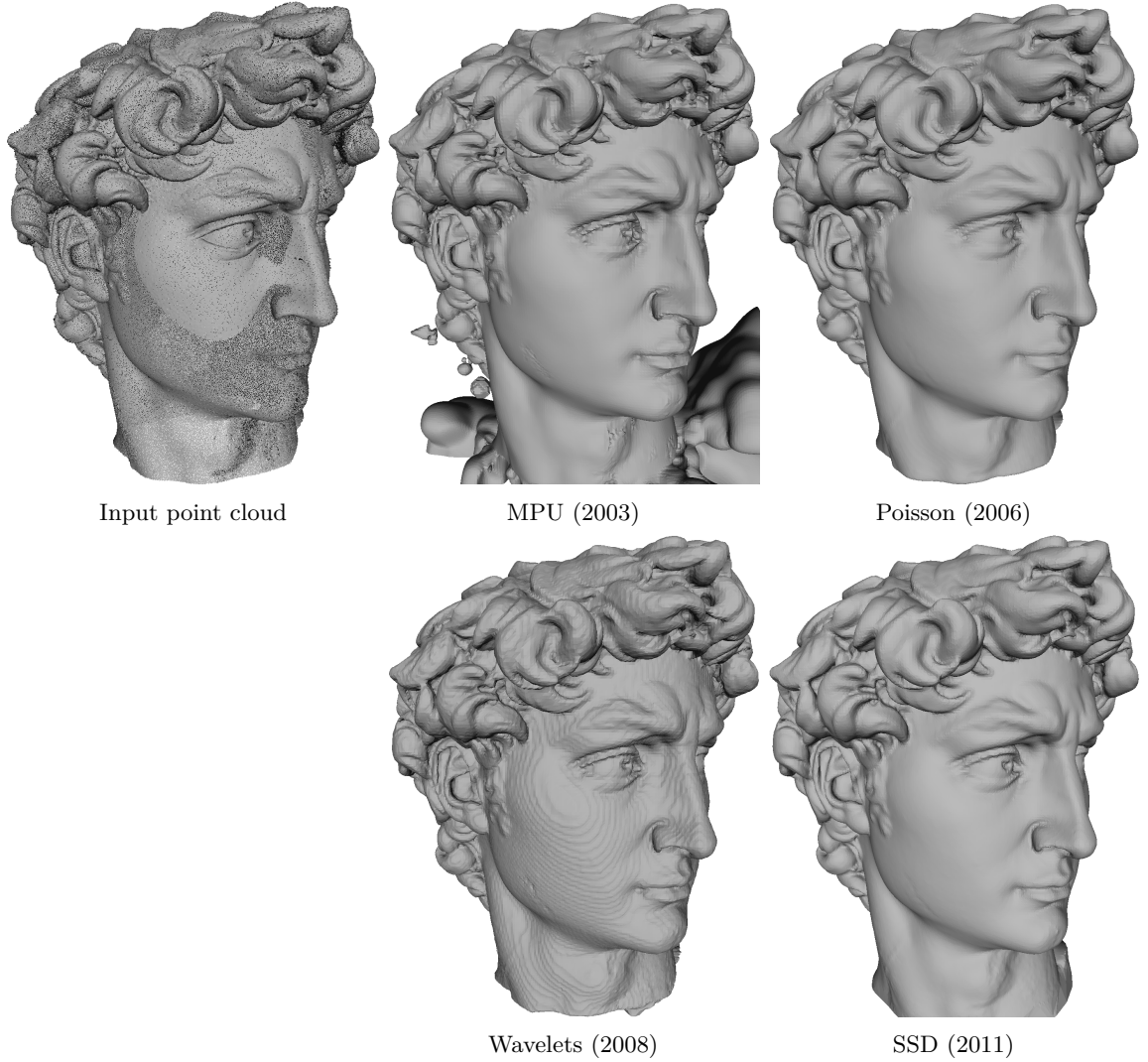


Figure 2.6: Surface reconstructions of David’s head with 1 million samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of uniformly distributed points and accurate normals.

almost uniformly sampled, we ignore the noisy triangles provided by the scanner, and use local surface fitting to estimate a sample’s normal. As a result, we have uniformly distributed points with accurate normals. Figure 2.6 compares different reconstructions. All methods operate on the same data set at a maximal octree depth of 8, and produce surfaces of comparable quality. Table 2.1 summarizes the performance characteristics of each algorithm.

Our second test case is the left eye of Michalengelo’s David raw dataset with 187,526 samples. This dataset is cropped from the head of Michalengelo’s David raw dataset. We estimated normals

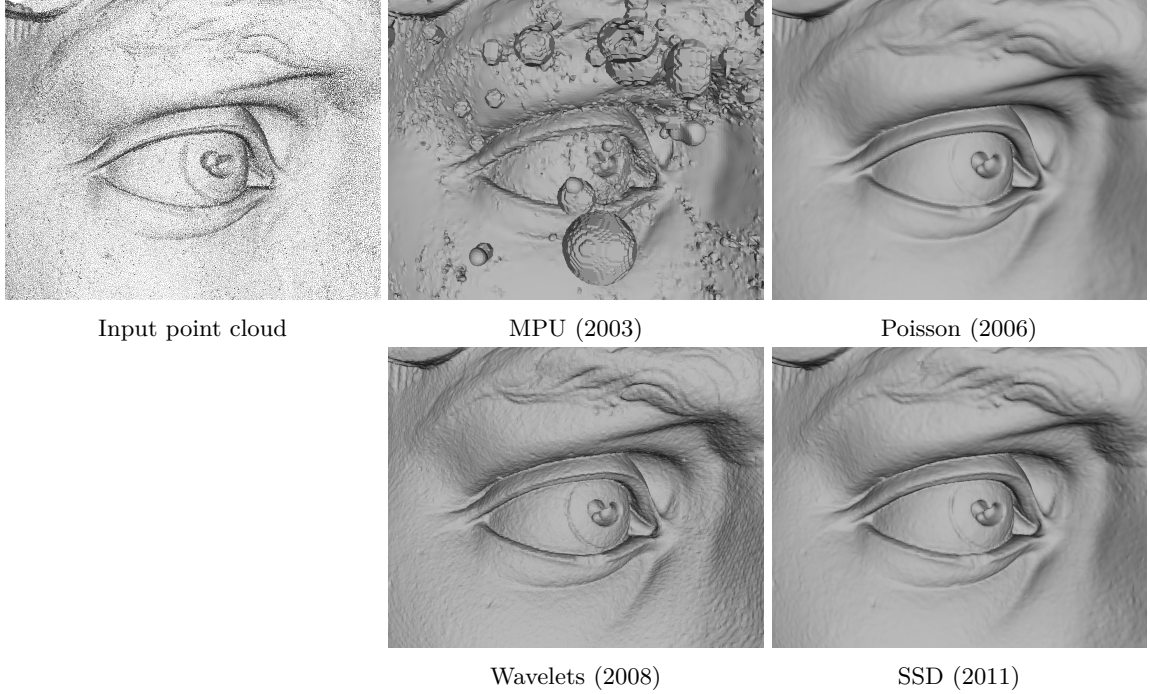


Figure 2.7: Surface reconstructions of David’s eye with 187,526 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of uniformly distributed points and *inaccurate* normals.

from the noisy triangles provided by the scanner. The resulting oriented point set has uniformly distributed points with many *inaccurate* normals. Figure 2.7 compares different reconstructions at a maximal octree depth of 10. Since the data is noisy, MPU results in reconstruction with spurious surface sheets. Poisson, D4 Wavelets, and our SSD Reconstruction all accurately reconstruct the surface, although the one reconstructed with Poisson is a bit smoother.

Our third test case is composed of 100,000 oriented point samples obtained by sampling a virtual horse model with a sampling density proportional to curvature, giving a set of *non-uniformly* distributed points with accurate normals. Figure 2.8 compares different reconstructions at a maximal

Table 2.1: The running time, the peak memory usage, and the number of triangles in the reconstructed surface of the David’s head generated with different methods.

Method	Time (Sec)	Memory (MB)	Polygons
MPU	27	148	378925
Poisson	43	283	319989
D4	17	63	365974
SSD	72	264	346652

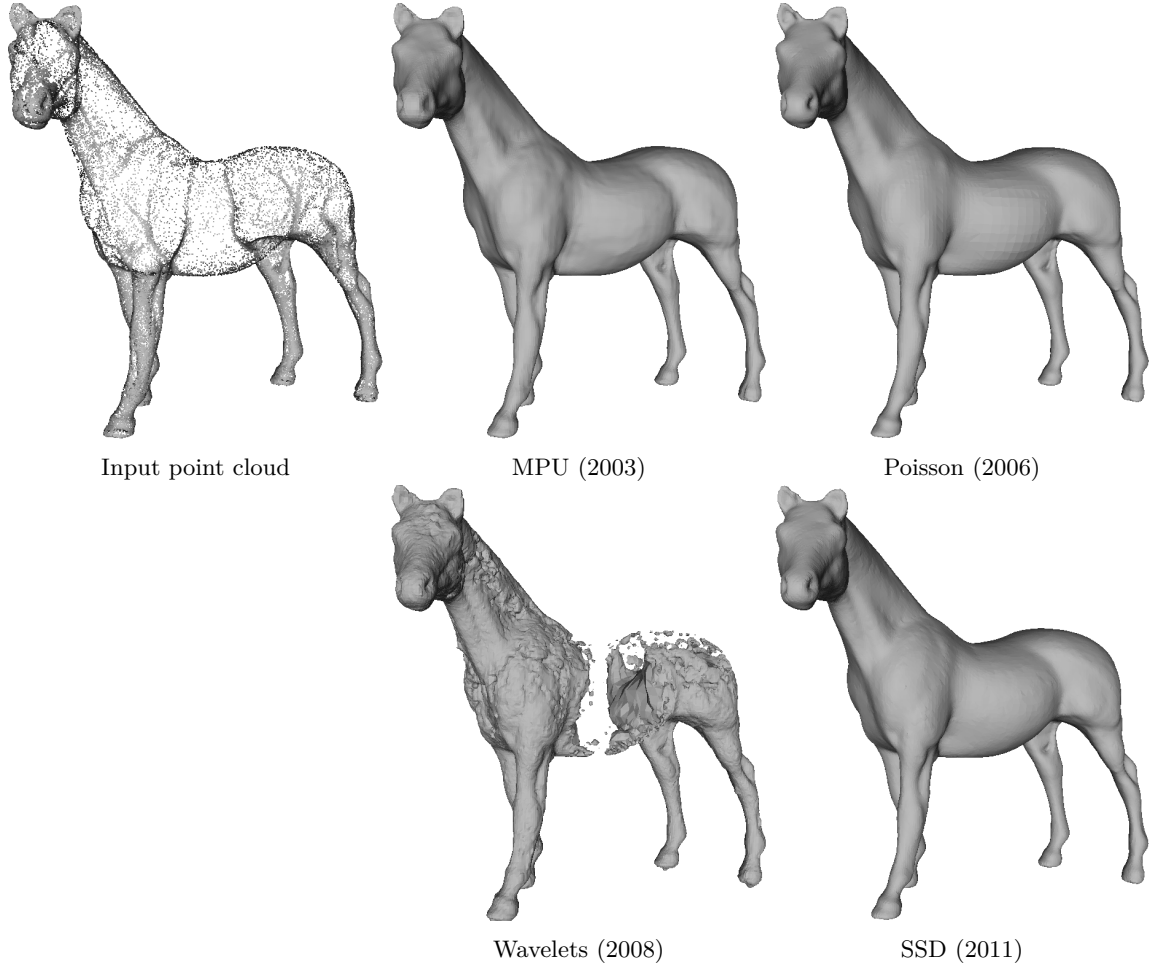


Figure 2.8: Surface reconstructions of a Horse model with 100,000 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of *non-uniformly* distributed points and accurate normals.

octree depth of 9. The D4 Wavelets Reconstruction fails to reconstruct the surface accurately. MPU, Poisson, and our SSD Reconstruction all accurately reconstruct the surface with subtle differences.

Our final test case is the Chiquita model raw dataset comprising 705,375 samples. This dataset was acquired by scanning the real world object with an inexpensive 3D structured lighting system [107]. It is an example of *non-uniformly* distributed points with *inaccurate* normals. Figure 2.9 compares different reconstructions at a maximal octree depth of 9. MPU produces spurious surface sheets. Although D4 Wavelets performs much better, it can not fill in gaps reasonably. Poisson, and our SSD Reconstruction both result in pleasing surfaces with subtle differences.

To evaluate the numerical accuracy of the reconstruction results we follow the same strategy as

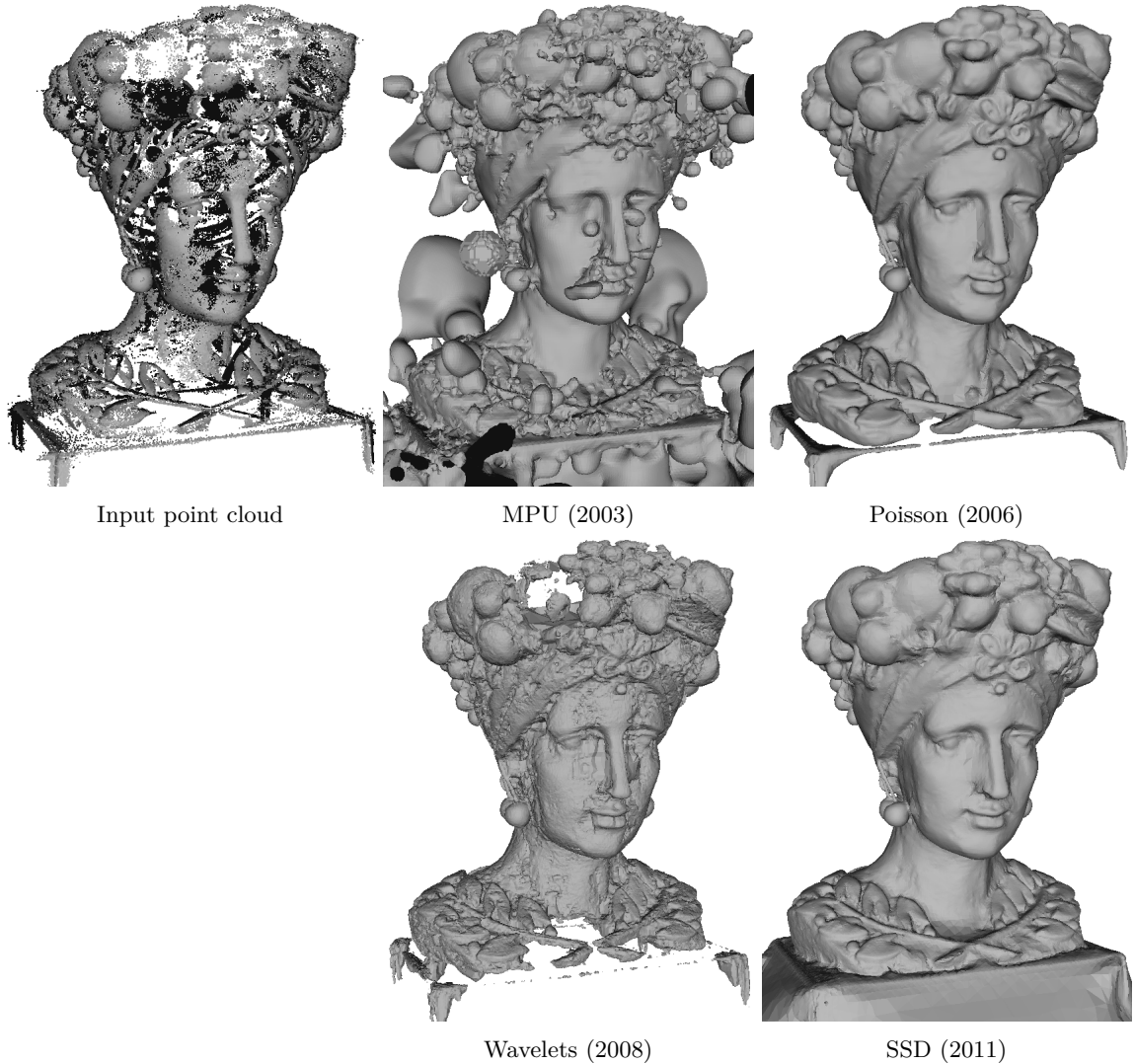


Figure 2.9: Surface reconstructions of a Chiquita model with 705,375 samples using MPU, Poisson, D4 Wavelets, and our SSD methods. The point cloud consists of *non-uniformly* distributed points and *inaccurate* normals.

in [73]. We first sample points from a known model, then reconstruct surfaces using each method with this point set. We then compute the Hausdorff distance between each reconstructed model and the known model using Metro tool [108]. Table 2.2 summarizes the result. Note that, in all cases, our SSD reconstruction recovers the surfaces with higher degree of accuracy than the Poisson reconstruction. And in two cases (Horse, and Igea) it also outperforms D4 Wavelets.

2.11 Relation to Poisson Surface Reconstruction

As in our formulation, the Poisson Surface Reconstruction approach [72] looks for an implicit function which should be zero at the data points, and such that its gradient evaluated at the data points equals the corresponding normal vectors. The set of oriented data points $\mathcal{D}$ is regarded as a sparse sampling of a continuous vector field defined on the bounding volume V . As a result, the vector field samples are first extended to a continuous three dimensional vector field $v(x)$ defined on the volume V . The implicit function $f(x)$ is subsequently recovered integrating the vector field, *i.e.*, solving the functional equation $\nabla f = v$. But since not every vector field is the gradient of a function the problem may or may not have solution. The necessary and sufficient condition for a smooth vector field to be the gradient of a function is that its curl be identically zero, or equivalently, that the integral of the vector field along any closed curve be equal to zero. Since these conditions are difficult to impose as constraints, an alternative least-squares solution is proposed, where the functional equation is approximately solved by minimizing the integral equation

$$\int_V \|\nabla f(x) - v(x)\|^2 dx \quad (2.17)$$

with respect to the function $f(x)$, while keeping the vector field constant. Solving this variational problem is equivalent to solving the classical Poisson equation $\Delta f = \nabla \cdot v$ in V . From the linear algebra point of view, the solution of this problem is the orthogonal projection of the vector field v onto the subspace of gradient vector fields, with the solution represented in parametric form. The solution is usually unique modulo an additive constant f_0 , which is then determined by minimizing the fitting error

$$\sum_{i=1}^N (f(p_i) - f_0)^2$$

Table 2.2: Hausdorff distance between real surfaces and reconstructed surfaces. Each row is normalized by the maximum error to provide relative comparison (lower is better).

Model	MPU	Poisson	D4	SSD
Armadillo	1.0000	0.4617	0.2383	0.3514
Dragon	0.8779	1.0000	0.5301	0.6810
Horse	0.0752	0.0827	1.0000	0.0551
Igea	1.0000	0.7761	0.5701	0.4018

with $f(x)$ being a minimizer of the integral equation (2.17). The solution to this last problem is the average of the function on the data points

$$f_0 = \frac{1}{N} \sum_{i=1}^N f(p_i) .$$

Note the the whole process involves three minimization problems rather than one, performed in a sequential manner.

If we introduce the vector field v as an additional variable in our formulation, and we remove the first data term from the total energy of equation (2.3) we obtain a vector field energy

$$\frac{\lambda_1}{N} \sum_{i=1}^N \|v(p_i) - n_i\|^2 + \frac{\lambda_2}{|V|} \int_V \|Dv(x)\|^2 dx \quad (2.18)$$

which can be minimized independently. In this equation Dv denotes the Jacobian matrix of the vector field v . Note that if $\nabla f = v$, then $Hf = Dv$. The result is a vector field defined on the volume V which extends the oriented points, as in the Poisson Surface Reconstruction approach. Then the same other two steps, of solving the Poisson equation, and estimating the optimal isolevel by minimizing the data term originally removed from the total energy of equation (2.3) can be performed. In fact, this approach was shown to work in [109] to solve the surface reconstruction problem, for the function and vector field discretized on a regular voxel grid, and with the integral term of the energy discretized by finite differences.

The most important difference is that in both the FFT [96] and Poisson [72] surface reconstruction approaches the problem is formulated for an indicator implicit function. The choice of an indicator function for the implicit function creates complications, since the gradient of the implicit function does not exist in the traditional sense at the sample points, where the function is discontinuous. As a result the two methods estimate a low-pass filtered version of the indicator function. It is noted in [72] that in practice, care should be taken in choosing the smoothing filter, and that for non-uniformly distributed point samples the smoothing kernel width needs to be adapted. The adaptation process is only explained within the concept of the octree-based implementation. It is also reported in a recent surface reconstruction benchmark study [110] that these two methods tend to oversmooth the data, as we can also observe in some of the results presented herein. In the FFT approach, a truncated Fourier expansion automatically provides an ideal low-pass filtered version of

the indicator function, but the restriction to a regular voxel grid discretization limits its appeal, as in the case of [109]. In the Poisson Surface Reconstruction approach implemented on an octree, the vector field is constructed as a linear combination of compactly supported smooth basis functions associated with the octree cells. As a result, the function obtained by solving the Poisson equation is equally smooth. Since the vector field should approximate the low-pass filtered gradient of the indicator function, it is forced to be zero away from the data points.

2.12 Conclusion

In this chapter we have introduced a new variational formulation for the problem of reconstructing a watertight surface defined by an implicit equation from a finite set of oriented points. As in other surface reconstruction approaches discretizations of this continuous formulation reduce to the solution of sparse least-squares problems. Rather than forcing the implicit function to approximate the indicator function of the volume bounded by the surface, in our formulation the implicit function is a smooth approximation of the signed distance function to the surface. We introduced a very simple hybrid FE/FD discretization, which together with an octree partitioning of space, and the Dual Marching Cubes algorithm produces accurate and adaptive meshes. The software implementation used to create the figures shown in this chapter are available for download from [111].

Chapter 3

Smooth Color Map Estimation (SSD-C)

3.1 Introduction

This chapter addresses the problem of reconstructing a smooth color map together with the geometry and topology of a watertight surface from a finite set of *colored* oriented points. This problem has a wide range of applications in industry, entertainment, virtual reality, archeology, forensics, art, medical imaging, and many other areas. Colored oriented point clouds are nowadays obtained using various optical measuring devices, such as laser scanners and inexpensive structured lighting systems, by other computational means such as multi-view stereo reconstruction algorithms, and also result from large scale simulations. We are particularly interested here in the reconstruction of colored surfaces from multi-view image and video data captured using consumer level digital and video cameras.

In the image-based modeling context, texture atlases are built and mapped to resulting polygon mesh models as the last part of multi-view stereo pipeline [112], which greatly enhances the visual details. However, the problem is particularly challenging due to the non-uniform sampling nature of the data collection processes. Common problems to cope with include: non-uniform sampled data due to concave surfaces, missing data due to inaccessible regions, noisy data due to sensor incapacities, and distorted data due to misalignment of partial scans. In addition, the atlas creation

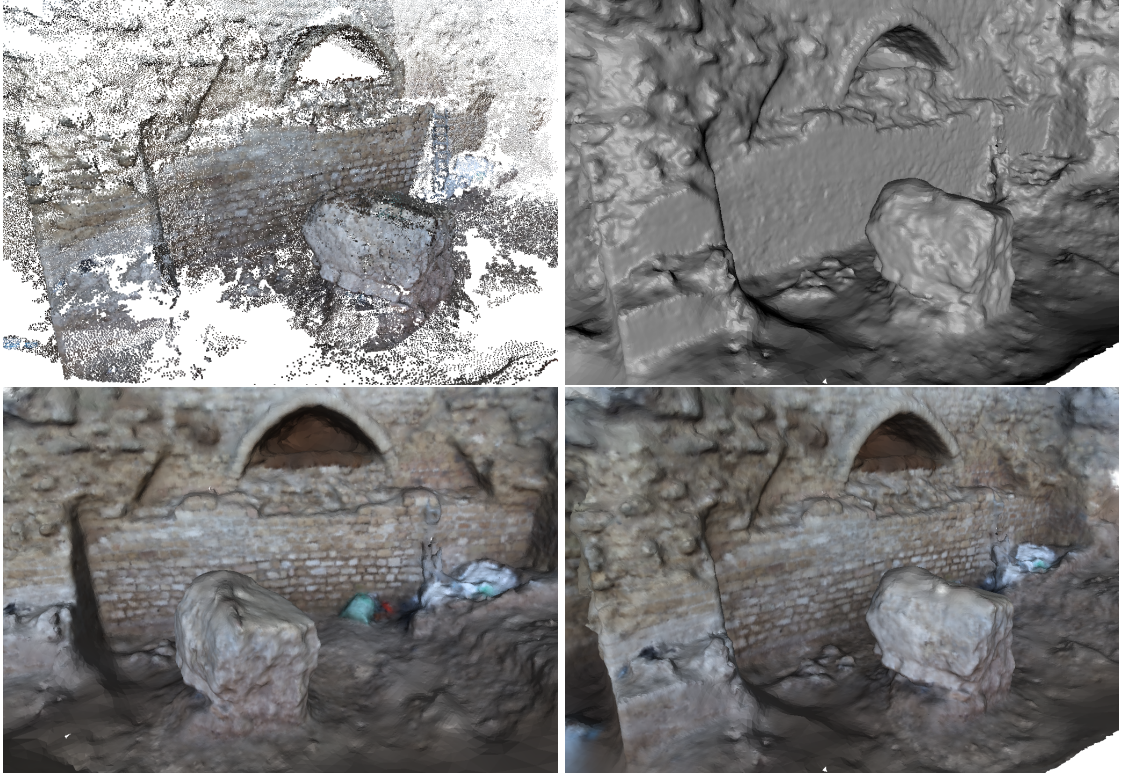


Figure 3.1: Reconstruction of the side of a castle model: The input point cloud (top-left), Surface reconstructed by the proposed algorithm (top-right), Two views from the surface and color map reconstructed by the proposed algorithm (bottom).

process gets complicated due to various photometric and geometric inaccuracies such as varying lighting conditions, non-Lambertian reflectance, and imperfect camera calibration, and causes unsatisfactory ghosting effects, and are unable to achieve perfect texture or color continuity. The method proposed estimates a smooth appearance as a less expensive alternative to creating texture atlases [113]. We are particularly concerned with developing algorithms which avoid most of these difficulties, and can scale up to large data sets.

In Chapter 2 we introduced the Smooth Signed Distance Surface Reconstruction (SSD) method for reconstructing the geometry and topology of a watertight surface. In this chapter we show how to extend this method to the reconstruction of colored surfaces by estimating an additional continuous color map defined on the reconstructed surface geometry. This color map approximates the colors associated with the data points at the point locations, and extrapolates with continuity those colors to areas where the sampling rate is low. The color map estimation step extends the variational formulation (2.3) to accommodate appearance estimation over surfaces defined by implicit

functions. As a result, each problem reduces to the solution of a sparse linear system which can be solved independently of the geometry and topology estimation by using an appropriate solver.

We devote Section 3.2, Section 3.4, and Section 3.5 to describing the variational formulation, the discretization, and the octree-based implementation, respectively. We present our results in Section 3.6. Figure 3.1 shows one example of surface and color map reconstructed with the proposed method which provides a good compromise between visual detail and color continuity while blurring effects can be observed near undersampled regions.

3.2 Continuous Formulation

Given a colored oriented point cloud $\mathcal{D}_{\text{color}} = \{(p_1, n_1, c_1), \dots, (p_N, n_N, c_N)\}$, where p_i is a surface location sample, n_i is the corresponding surface normal sample oriented towards the outside of the object, and c_i is the corresponding color sample, a signed distance function f that implicitly defines the surface S by $S = \{x : f(x) = 0\}$ can be estimated only from $\mathcal{D} = \{(p_1, n_1), \dots, (p_N, n_N)\}$ as explained in Chapter 2. In this chapter, we are interested in estimating a color map function g defined on the same bounded volume V , *i.e.*, $g : V \rightarrow \mathbb{R}$ independently from the implicit function estimation. Since the colors c_i are regarded as the samples of the surface appearance at the points p_i , the color map function should ideally satisfy $g(p_i) = c_i$ for all the points $i = 1, \dots, N$ in the data set. However, we again require that this condition be satisfied in the least-squares sense. As a result, we consider the problem of minimizing the following data energy

$$\mathcal{Q}_{\mathcal{D}}(g) = \frac{1}{N} \sum_{i=1}^N (g(p_i) - c_i)^2. \quad (3.1)$$

Depending on which family $\mathcal{G}$ of functions $g(x)$ is considered as the space of admissible solutions, the problem of minimizing the energy function $\mathcal{Q}_{\mathcal{D}}(g)$ of equation (3.1) may or may not be well conditioned. In particular, this energy function does not specify how the function should behave away from the data points, in particular, in regions of missing data. To constraint all the parameters we add the following regularization term to the energy function

$$\mathcal{Q}_{\mathcal{R}}(g) = \frac{1}{|V|} \int_V \|\nabla g(x)\|^2 dx \quad (3.2)$$

where $\nabla g(x)$ is the gradient vector of g , the 3×1 vector of first order partial derivatives of $g(x)$, and the norm of the vector is the ℓ_2 -norm. The integral is over the same volume V where we reconstruct the surface, and $|V| = \int_V dx$ is the measure of this volume. The total energy function in the proposed formulation is a weighted average of the data and regularization terms

$$\mathcal{Q}(g) = \mathcal{Q}_{\mathcal{D}}(g) + \mu \mathcal{Q}_{\mathcal{R}}(g) , \quad (3.3)$$

and μ is a positive constant, and increasing μ produces a smoother result. In the vicinity of the data points the data energy term dominates the total energy, and makes the function approximate the true appearance. Away from the data points the regularization energy term dominates the total energy and tends to make the color function $g(x)$ constant.

3.3 Linearly Parameterized Families

In this section we are concerned with the existence and uniqueness of a solution for the variational problem defined by the energy function $\mathcal{Q}(g)$ in equation (3.3). We restrict the analysis here to families of functions linearly parameterized by a finite number of parameters. Explicitly, a member of this family can be written as a linear combination of certain basis functions

$$g(x) = \sum_{\alpha \in \Lambda} g_{\alpha} \psi_{\alpha}(x) \quad (3.4)$$

where α denotes an index which belongs to a finite set Λ , say with K elements, $\psi_{\alpha}(x)$ is a basis function, and g_{α} is the corresponding coefficient. The K basis functions are chosen in advance, and are not regarded as variables in the problem. The function can also be written as an inner product

$$g(x) = \Psi(x)^{\top} G , \quad (3.5)$$

where $G = (g_{\alpha} : \alpha \in \Lambda)$ and $\Psi(x) = (\psi_{\alpha}(x) : \alpha \in \Lambda)$ are K -dimensional column vectors. Restricting the energy $\mathcal{Q}(g)$ to this function results in a non-homogeneous and quadratic function $G^{\top} B G - 2 d^{\top} G + e$ in the K -dimensional parameter vector G . The matrix B is usually positive definite, and the resulting minimization problem has a unique minimum. The global minimum is obtained by solving the system of linear equations $B G = d$.

3.4 Discretization with Discontinuous Function

A hybrid of finite-element and finite differences scheme is presented in Chapter 2 to discretize the energy function $\mathcal{E}(f)$ of equation (2.3): a finite element discretization for the function $f(x)$, and finite differences discretizations for the gradient $\nabla f(x)$ and the Hessian $Hf(x)$. Since all these terms are written as homogeneous linear functions of the same parameter vector F , the problem still reduces to the solution of linear equations $AF = b$. As this particular discretization is applicable to adaptive grids, we adopt a similar approach to discretize the energy function $\mathcal{Q}(g)$ of equation (3.3). We use a finite element discretization for the color function $g(x)$, and finite differences discretization for the gradient $\nabla g(x)$.

To simplify the notation, we first consider a regular hexahedral grid partition of space. We assume that the volume V is a unit cube $V = [0, 1] \times [0, 1] \times [0, 1]$, and that each of the three axes is split into M equally long segments, resulting in M^3 hexahedral cells and $(M+1)^3$ vertices. In this discretization, the indices $\alpha \in \Lambda$ are the grid cell multi-indices $\alpha = (i, j, k)$, for $0 \leq i, j, k < M$. The coefficients g_α are then the values of the color function $g(x)$ at the grid centroids $p_\alpha = (\frac{i+0.5}{M}, \frac{j+0.5}{M}, \frac{k+0.5}{M})$. Determining which cell C_α each point p_i falls into reduces to quantization. If $p_i \in C_\alpha$ then

$$g(p_i) \approx g_\alpha \quad (3.6)$$

Since the piecewise constant function leads to zero first derivatives within each cell, the square norm of the gradient $\nabla g(x)$ is a generalized function supported on the square faces shared by neighboring voxels (as Dirac's δ "function"), and the integral over the volume V reduces to a finite sum over the faces. Hence, we only need an approximation to $\|\nabla g(x)\|^2$ at the faces shared by cells. If C_α and C_β are two such cell that share a face, then we use finite differences

$$\nabla_{\alpha\beta} g = \frac{1}{\Delta_{\alpha\beta}} (g_\alpha - g_\beta) \quad (3.7)$$

where $\Delta_{\alpha\beta}$ is the Euclidean distance between the centers of the cells. As a result, the integral in the regularization term of equation (3.2) becomes a sum over all the neighboring grid cell pairs

$$\mathcal{Q}_{\mathcal{R}}(g) \approx \frac{1}{W} \sum_{(\alpha, \beta)} (g_\alpha - g_\beta)^2 w_{\alpha\beta} . \quad (3.8)$$

The associated smoothing weight is

$$w_{\alpha\beta} = \frac{A_{\alpha\beta}}{\Delta_{\alpha\beta}}$$

where $A_{\alpha\beta}$ is the area of the common face shared by two cells C_α and C_β , and $\Delta_{\alpha\beta}$ is the Euclidean distance between the two cells. $W = \sum_{\alpha\beta} w_{\alpha\beta}$ is the normalization factor so that $w_{\alpha\beta}$'s sum up to one.

3.5 Octree-Based Implementation

Similar to the discretizations presented in Section 2.6 for the estimation of the implicit function f , the finite-differences discretization presented in the previous section for regular hexahedral grids extends in a natural way to octree cells. There is one function value g_α associated with each leaf cell of the octree. As a result, within each cell C_α , we use g_α to discretize the function $g(x)$, *i.e.*, a piece-wise constant function, see equation (3.6). We use the discrete form of the regularization term (3.8) based on finite differences (3.7) between the pairs of octree leaf cells C_α and C_β that share a common face, *i.e.*, dual edges of the octree, see Figure 2.5 for an illustration. Finally, we reach to the octree-based discretization of the continuous energy (3.3)

$$\mathcal{Q}(g) \approx \frac{1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} (g_\alpha - c_i)^2 + \frac{\mu}{W} \sum_{(\alpha, \beta)} (g_\alpha - g_\beta)^2 w_{\alpha\beta}, \quad (3.9)$$

where $\alpha^* = \{i : p_i \in \mathcal{D}_\alpha\}$, *i.e.*, the set of indices of points enclosed in octree leaf cell C_α . The point cloud $\mathcal{D}$ is partitioned into disjoint sets $\mathcal{D}_\alpha$ during the recursive octree construction process. Note that the minimization of the discrete energy (3.9) still reduces to the solution of a linear system $BG = d$. After accumulating the elements of matrix B and vector d , for reasonably sized problems one can use a direct solver. Instead, we use an iterative linear solver. Most iterative methods designed to solve linear systems do not require matrices to be built explicitly, *i.e.*, the result of the matrix vector product $BG^{(k)}$ is only needed for a given vector $G^{(k)}$ to perform an iteration k [78]. We accumulate the $K \times 1$ result vector of the product $BG^{(k)}$ and the $K \times 1$ right

hand side vector d through following sums:

$$\begin{aligned}
B G^{(k)} &= \frac{1}{N} \sum_{\alpha} |\mathcal{D}_{\alpha}| \Psi_{\alpha} \left\{ \Psi_{\alpha}^T G^{(k)} \right\} \\
&+ \frac{\mu}{W} \sum_{(\alpha, \beta)} \Psi_{\alpha} \left\{ \Psi_{\alpha}^T G^{(k)} - \Psi_{\beta}^T G^{(k)} \right\} w_{\alpha\beta} \\
&- \frac{\mu}{W} \sum_{(\alpha, \beta)} \Psi_{\beta} \left\{ \Psi_{\alpha}^T G^{(k)} - \Psi_{\beta}^T G^{(k)} \right\} w_{\alpha\beta}
\end{aligned} \tag{3.10}$$

$$d = \frac{1}{N} \sum_{\alpha} \Psi_{\alpha} \sum_{i \in \alpha^*} c_i$$

where the first term is the contribution from the data term, and the last two terms are the contributions from the regularization term after expanding the finite difference. Ψ_{α} is a $K \times 1$ sparse vector with a single non-zero entry with value 1 for the multi-index α associated with octree leaf cell C_{α} . Then, $\Psi_{\alpha}^T G^{(k)}$ simply evaluates to scalar $g_{\alpha}^{(k)}$, and $\Psi_{\beta}^T G^{(k)}$ to $g_{\beta}^{(k)}$. As a result, each term in a given sum simply updates a single value of $B G^{(k)}$, *i.e.*, the value associated with the octree leaf C_{α} or C_{β} .

We first estimate the implicit function $f(x)$ using an iterative cascading multigrid approach as described in Chapter 2. That is, we start by solving the problem on a much coarser level than desired using a conjugate gradient solver. Then, we use it to initialize the solution at the next level, which is then refined by the conjugate gradient solver. We then estimate the color function $g(x)$ on the same level that is used to get the final estimate of the implicit function $f(x)$ using another conjugate gradient solver. To compute an RGB color map, we repeat this minimization procedure for each color channel independently.

We use the Dual Marching Cubes algorithm [11] to construct a polygonal approximation of the isolevel zero of the implicit function $f(x)$. Dual Marching Cubes requires values of the implicit function defined at the centroids of the octree leaf cells. The computed implicit function f is defined at the vertices of the primal graph. An implicit function value is computed for the centroid of each leaf cell by averaging the values associated with the eight cell corners. On the other hand, the results of the color function g are already defined at centroids of the octree leaf cells (at the vertices of the dual graph), so we can directly transfer coefficients g_{α} to the Dual Marching Cubes algorithm, which later does linear interpolation to obtain a color map function defined on the vertices of the extracted isosurface, *i.e.*, to assign color per vertex attribute on the final mesh.

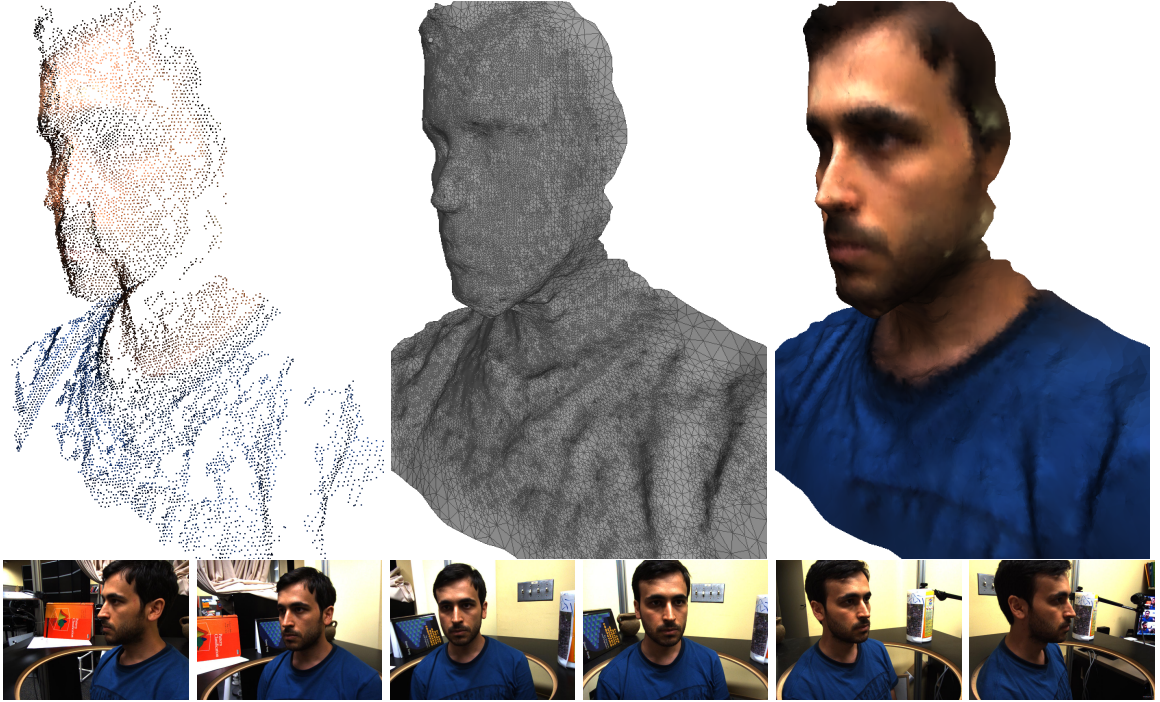


Figure 3.2: Reconstruction of a human frontal view. Top row: the input point cloud (top-left), surface geometry (top-middle), and color map (top-right) reconstructed by the proposed algorithm. Bottom row: 6 images that are used for shape acquisition.

3.6 Results

We apply our algorithm to colored oriented point clouds retrieved using the well-known multi-view stereo (MVS) pipeline consisting of two publicly available algorithms Bundler [37], and the Patch-based MVS (PMVS) [38]. The goal of multi-view stereo is to reconstruct a 3D model from images taken from known camera viewpoints. To learn the viewpoints, we use *Bundler* which takes a set of images as input and accurately estimates the camera viewpoint per each image. We then use the Patch-based MVS (PMVS) algorithm which takes the same set of images and the estimated camera parameters as input and produces 3D models represented as colored point clouds with accuracy nearly comparable with laser scanners [114]. PMVS is based on the idea of correlating measurements from several images at once to derive globally consistent 3D surface samples. We use a standard digital camera for all the data sets except for the face dataset. In this case, we have to use a set of synchronized cameras as the person could move during data collection procedure. In addition, each camera is calibrated with respect to a common coordinate system.

Our initial test case is the frontal view of a human, raw dataset of 12,346 points retrieved from

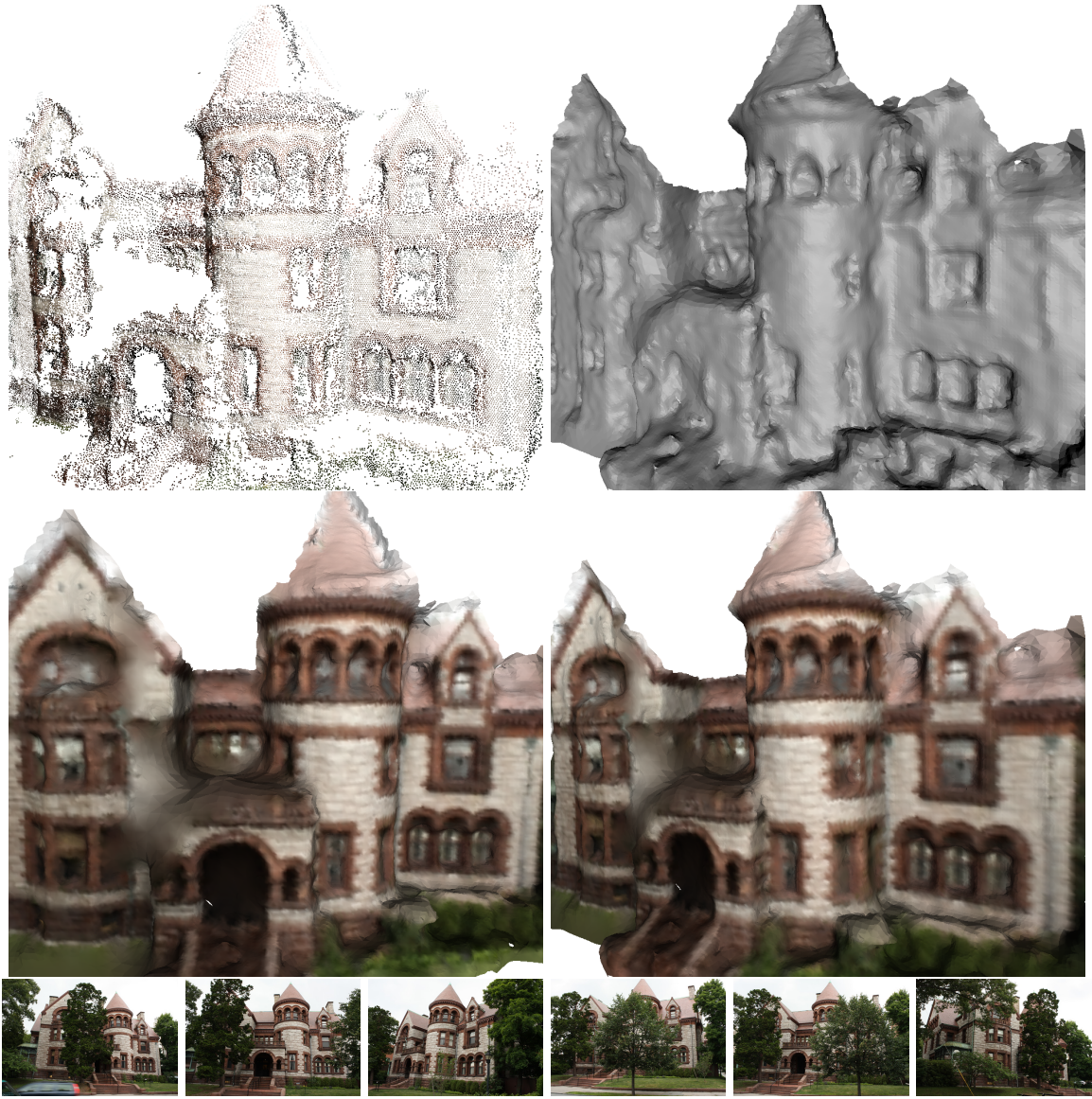


Figure 3.3: Reconstruction of a building facade. Top row: The input point cloud, and surface geometry reconstructed by the proposed algorithm. Middle row: Two views from the surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 14 images that are used for shape acquisition.

6 photographs. Each image is captured at resolution 1024×768 with a time-synchronized camera system consisting of 6 cameras. The proposed method results in a model consisting of 64,909 polygons when a maximal octree of depth 9 is used. Figure 3.2 shows the reconstruction of the model, and the input point cloud.

Our second test case is the facade of the Applied Mathematics building of Brown University, a

raw dataset of 66,935 points retrieved from 14 photographs. Each image is captured at resolution 1282×854 with a standard digital camera. The proposed method results in a model consisting of 77,603 polygons when a maximal octree of depth 10 is used. Figure 3.3 shows the reconstruction of the model, and the input point cloud.

Our third test case is the brick with Mesopotamian cuneiform inscriptions, a raw dataset of 333,401 points retrieved from 21 photographs. Each image is captured at resolution 3072×2304 with a standard digital camera. The proposed method results in a model consisting of 1,034,663 polygons when a maximal octree of depth 11 is used. Figure 3.4 shows the reconstruction of the model, and the input point cloud.

Our last test case is a shoe-print, raw dataset of 180,885 points. This dataset is retrieved from 11 photographs. Each image is captured at resolution 4272×2848 . The proposed method results in a model consisting of 897,273 polygons when a maximal octree of depth 11 is used. Figure 3.5 shows the reconstruction of the model, and the input point cloud.

All the experiments are conducted on a PC with a Intel Core 2 Duo @2.26GHz processor and a 4GB ram. A non-threaded in-core CPU implementation takes 3-15 minutes depending on the maximum depth of the tree that will be used for reconstruction. It consumes 90% of the time for surface reconstruction, and the remaining 10% for color map reconstruction. We demonstrate that this method produces watertight surfaces with pleasing smooth color maps even on data sets with various complications: non-uniform sampling, sensor noise, and missing regions.



Figure 3.4: Reconstruction of a brick with cuneiform, Mesopotamian, 859-840 BCE, clay, Overall 35 x 35 x 11 cm, Williams College Museum of Art, Williamstown, MA, Gift of Professor Edgar J. Banks and Dr. John Henry Haynes, Class of 1876, (20.1.33.A). Top row: the input point cloud (left), and surface geometry (right) reconstructed by the proposed algorithm. Middle row: Two views from surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 21 images that are used for shape acquisition.



Figure 3.5: Reconstruction of a shoe print. Top row: the input point cloud (left), and surface geometry (right) reconstructed by the proposed algorithm. Middle row: Two views from surface and color map reconstructed by the proposed algorithm. Bottom row: 6 examples from the set of 11 images that are used for shape acquisition.

3.7 Conclusion

This chapter presented a complete framework for the problem of reconstructing a 3D scene’s surface and color map from a finite set of colored oriented points. We used the variational formulation introduced in Chapter 2 for the problem of reconstructing a watertight surface defined by an implicit equation. We then extended this by introducing a new variational formulation for the problem of estimating a smooth appearance defined over the surface. We showed that a simple hybrid FE/FD discretization still reduced the problem to the solutions of sparse linear systems. A proof-of-concept implementation has been shown to produce high-quality polygon meshes with pleasing color-per-vertex function.

Chapter 4

Generalization of SSD to Volumetric Data (GSSD)

4.1 Introduction

In the last decade, multi-view stereo (MVS) has seen much attention and a plethora of algorithms have been introduced. Please refer to [114] for a review and taxonomy of MVS algorithms, and to [115] for a tutorial on MVS. In particular, the Middlebury benchmark [114] has fueled the development of algorithms tailored for small objects under controlled environments, where the top performing methods are capable of challenging the accuracy of laser scanners. However, most of these approaches are not suitable for dealing with high resolution aerial imagery of large scale urban scenes. In particular, algorithms that partially rely on shape-from-silhouette techniques [116–118], under the assumption that a single object is visible and can be segmented from the background, are not applicable to aerial imagery due to unavailability of fully visible silhouettes.

The Patch-based Multi-view Stereo (PMVS) algorithm proposed by Furukawa and Ponce [38] is considered state of the art amongst methods based on feature extraction and matching. It produces colored oriented point clouds, and performs well for small objects as well as for large urban scenes, and can recover significant geometry in such scenes (see Fig. 4.4 top right image). However, methods based on feature extraction and matching often show significant errors caused by severe occlusions, highly reflective surfaces and areas of constant appearance. Such phenomena are commonplace in

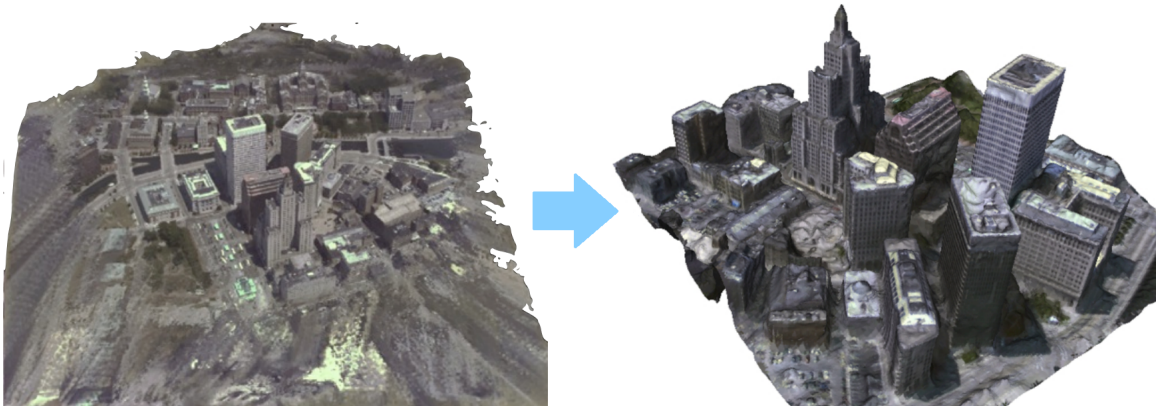


Figure 4.1: Surface Reconstruction of an urban scene from volumetric likelihood of surface occupancy μ using the proposed *generalization* of SSD, namely GSSD. The likelihood of surface occupancy μ in the volume is learned by using a multi-view stereo method (CPVM [80, 81]) from aerial imagery captured by flying a helicopter around the city of Providence, RI.

urban scenes, resulting in the generated point clouds often containing gross errors in localization and significant regions where feature points are not detected at all (see Fig. 4.4 middle right and bottom right images). Surface reconstruction algorithms, such as those reviewed in [53, 62, 110] are typically applied to the resulting point clouds. Although these methods produce excellent quality surfaces when applied to clean and accurate data, they often produce inaccurate surfaces when the input point cloud contains too many inaccuracies and outliers (see Fig. 4.5 first column).

Probabilistic models have been proposed as an alternative approach to handle complicated scene geometry. These models do not make hard decisions about surface geometry and/or appearance; instead they explicitly represent uncertainties by assigning probabilities to multiple hypothesis within the volume. Early works along this line [119, 120] can be regarded as extensions of Space Carving [121], and more recently, algorithms based on generative models for the reverse image formation process have been introduced [122, 123]. Using Bayesian inference, these algorithms infer the maximum a-posteriori probabilities in the volume from the joint probability of all the images. Pollard and Mundy [79] propose an online alternative, which in theory, can handle an infinite number of images. However, none of these volumetric methods implemented with regular grid discretizations gracefully scale up to large scenes because of the cubic space and time complexity. Crispell *et al.* [81, 124] addressed these limitations with a continuous formulation of Pollard and Mundy’s method implemented in an octree. Moreover, a GPU implementation presented in [80] is capable of learning high resolution probabilistic volumetric models of large urban scenes efficiently where one pass of

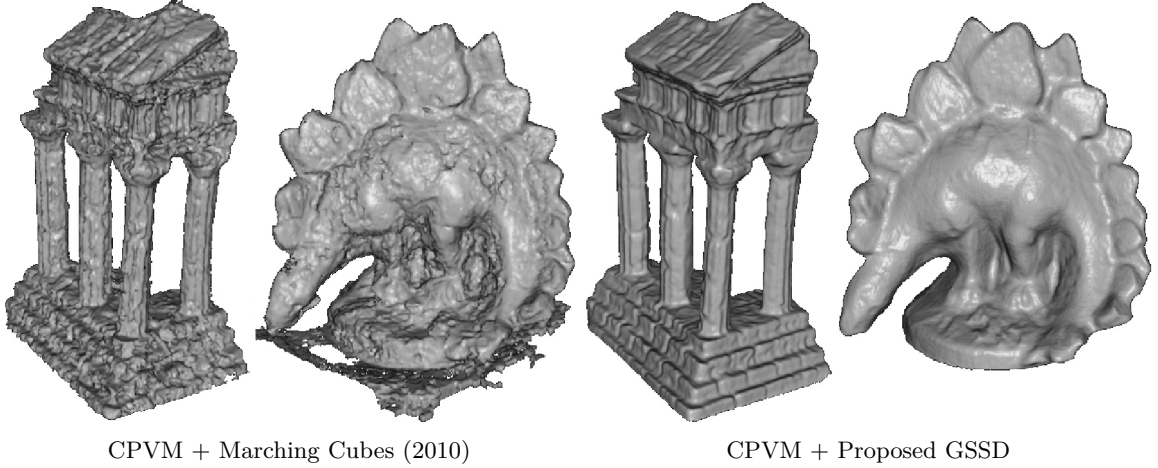


Figure 4.2: Surface Reconstructions of Temple and Dino models from volumetric likelihood of surface occupancy. Two models on the left are obtained by directly applying Marching Cubes algorithm to extract manually selected isolevels of the surface occupancy as described in [124]. Two models on the right are obtained by the proposed method. The surface occupancy was learned by CPVM [80, 81] from multiple photographs of the models provided with the Middlebury benchmark [114].

the online update takes approximately one second.

It is not clear how to estimate surfaces from probabilistic volumetric models. A simple approach is to compute the isosurface associated with a certain probability threshold [123–125]. Since a different threshold may be necessary in different regions of space, this method often fails to produce satisfactory results (see two models on the left in Figure 4.2), and the lack of surface orientation due to unsigned nature of surface density functions (see Figure 1.8 for a 2D illustration) may result in “double-sided” surfaces. Yezzi *et al.* [126] propose a surface evolution approach to obtain smooth surfaces. Also related are global optimization methods such as graph cuts to extract surfaces from a volumetric photo-consistency function [127–129]. While graph cuts allows for flexible energy functions and exact solutions, the method is not applicable to large scenes due to its very high memory requirements [129].

The main contribution this chapter is the *generalization* of the SSD introduced in Chapter 2 which was originally designed to reconstruct surfaces from a finite set of oriented points to allow reconstruction of surfaces from volumetric occupancy data. The proposed generalization of SSD is referred to as GSSD for the rest of the chapter. SSD estimates an implicit signed distance function defined over a bounded volume using adaptive octree-based discretizations. Using its continuous volumetric

formulation, we extend the formulation of SSD to incorporate non-negative continuous surface densities, and also singular measures defined over oriented points. With singular measures, the original formulation of SSD becomes a special case of GSSD. Similar to SSD, finite-element/finite-difference discretization of GSSD formulation reduces to solving a set sparse system of linear equations. The resulting algorithms are efficient and scalable to large data sets, and generate highly detailed surfaces from probabilistic volumes. Fig. 4.1 shows an application of the proposed GSSD method which is run over the *entire volume* using the surface density as input. In the example, the surface density was estimated using a multi-view stereo method CPVM [80, 81](Appendix A.2) from aerial images of an urban scene.

4.2 Generalized Energy Formulation

The original formulation of SSD introduced in Chapter 2 reconstructs surfaces from oriented point cloud data. In this section, this formulation is generalized, targeted to reconstruct surfaces from probabilistic volumes. To generalize SSD, the energy function $\mathcal{E}(f)$ of equation (2.4) is modified by replacing the finite oriented point cloud with a continuous distribution of oriented points. The proposed energy function is

$$\mathcal{E}(f) = \int_V f^2 d\mu + \lambda_1 \int_V \|\nabla f - n\|^2 d\mu + \lambda_2 \int_V \|Hf\|^2 d\sigma, \quad (4.1)$$

where $d\mu(x)$ and $d\sigma(x)$ are finite measures (in the sense of measure theory), and $n(x)$ is a vector field defined in the volume. When these measures are defined by non-negative continuous densities $\mu(x)$ and $\sigma(x)$ (*i.e.*, when $d\mu(x) = \mu(x)dx$ and $d\sigma(x) = \sigma(x)dx$), the values of $\sigma(x)$ and $\mu(x)$ should be chosen to make sure that the first two data terms dominate near high probability areas (*i.e.*, $\mu(x)$ should be large and $\sigma(x)$ small), and the regularization term dominates near low probability areas ($\mu(x)$ small and $\sigma(x)$ large). Note that this formulation also allows for singular measures, or generalized functions as densities. In particular, the original SSD formulation (2.4) can be regarded as a special case of the continuous formulation (4.1), where

$$d\mu(x) = \sum_{i=1}^N \frac{1}{N} \delta(x - x_i) dx \quad \text{and} \quad d\sigma(x) = \frac{1}{|V|} dx \quad (4.2)$$

or more generally with a weight $\mu_i = \mu(x_i)$ assigned to each point x_i (see Appendix A.3 for a simple point weighting scheme which assigns high weights to sharp feature points such as those on edges or corners, while it assigns low weights to points that are part of planar regions).

In this analysis, f is restricted to belong to a finite dimensional vector space of functions:

$$f(x) = \sum_{\alpha \in \Lambda} f_{\alpha} \phi_{\alpha}(x) = \Phi(x)^{\top} F, \quad (4.3)$$

where α denotes an index which belongs to a finite set Λ , say with K elements, $\phi_{\alpha}(x)$ is a basis function, and f_{α} is the corresponding coefficient. Then, the energy function $\mathcal{E}(f)$ results in a non-homogeneous quadratic function $F^{\top} A F - 2 b^{\top} F + c$ in the K -dimensional parameter vector $F = (f_{\alpha})_{\alpha \in \Lambda}$. The matrix A is symmetric and positive definite, and the resulting minimization problem has a unique minimum. The global minimum is determined by solving the system of linear equations $A F = b$. The coefficients of matrix A and vector b requires computing inner products of every pair of basis functions. Depending on how large the support of chosen basis functions, a large number of basis functions may overlap at any given point in the volume V . As a result, accumulating the coefficients of the matrix A and the vector b may require significant computation. An octree based finite-element/finite differences scheme is presented in Chapter 2 for the minimization of $\mathcal{E}(f)$ of (2.4), where the problem still reduces to the solution of linear equations $A F = b$, but the matrix A is much sparser, resulting in a fast and space-efficient algorithm.

We use a similar octree-based discretization, *i.e.*, $\mu(x)$ is a piecewise constant function, one value per octree leaf, and similarly $n(x)$ is a piecewise constant function, one vector per octree leaf. The parameter vector F has K elements, one value per vertex of the primal graph of the octree.

The primal graph represents the signed distance function f , and the dual graph of the octree (which has the octree leaf centroids as vertices) represents the surface density μ . $\{\alpha_0, \alpha_1, \dots, \alpha_7\} \subseteq \Lambda$ denotes the primal vertex indices associated with dual vertex (*i.e.*, an octree leaf) α , as depicted in Figure 2.3.

Note that $f(x)$ is a piecewise trilinear function, *i.e.*, its value at a point x in the volume V is determined through

$$f(x) = \sum_{h=0}^7 w_h f_{\alpha_h}$$

where $w_0, \dots, w_7$ are the trilinear coordinates of x in leaf α . By taking the piecewise definitions of

f and μ into account, the integrals of energy function $\mathcal{Q}(f)$ of (4.1) conveniently become finite sums over the octree leaves:

$$\mathcal{E}(f) \approx \frac{1}{W_1} \sum_{\alpha} f(x_{\alpha})^2 \mu_{\alpha} + \frac{\lambda_1}{W_1} \sum_{\alpha} \|\nabla f(x_{\alpha}) - n_{\alpha}\|^2 \mu_{\alpha} + \frac{\lambda_2}{W_2} \sum_{(\alpha, \beta)} \|\nabla f(x_{\alpha}) - \nabla f(x_{\beta})\|^2 \sigma_{\alpha\beta} , \quad (4.4)$$

where x_{α} is the centroid location of leaf α , μ_{α} is the surface density at leaf α , n_{α} is the oriented normal vector associated with leaf α . Note that μ_{α} , and n_{α} are estimated by CPVM, and kept fixed during surface estimation. The signed distance function f and its gradient ∇f are written using the elements of the parameter vector F :

$$f(x_{\alpha}) = \frac{1}{8} \left(f_{\alpha_0} + f_{\alpha_1} + f_{\alpha_2} + f_{\alpha_3} + f_{\alpha_4} + f_{\alpha_5} + f_{\alpha_6} + f_{\alpha_7} \right) , \quad (4.5)$$

$$\nabla f(x_{\alpha}) = \frac{1}{4\Delta_{\alpha}} \begin{pmatrix} f_{\alpha_4} - f_{\alpha_0} + f_{\alpha_5} - f_{\alpha_1} + f_{\alpha_6} - f_{\alpha_2} + f_{\alpha_7} - f_{\alpha_3} \\ f_{\alpha_2} - f_{\alpha_0} + f_{\alpha_3} - f_{\alpha_1} + f_{\alpha_6} - f_{\alpha_4} + f_{\alpha_7} - f_{\alpha_5} \\ f_{\alpha_1} - f_{\alpha_0} + f_{\alpha_3} - f_{\alpha_2} + f_{\alpha_5} - f_{\alpha_4} + f_{\alpha_7} - f_{\alpha_6} \end{pmatrix} , \quad (4.6)$$

where Δ_{α} the side length of leaf α . The third energy term is a sum over the pairs of octree leaves α and β that share a common face as depicted in Figure 2.5. The associated smoothing density $\sigma_{\alpha\beta}$ is determined naturally from the octree subdivision, *i.e.*, $\sigma_{\alpha\beta} = \frac{A_{\alpha\beta}}{\Delta_{\alpha\beta}}$, where $A_{\alpha\beta}$ is the area of the common face and $\Delta_{\alpha\beta}$ is the Euclidean distance between the centroids of these leaves. $W_1 = \sum_{\alpha} \mu_{\alpha}$ and $W_2 = \sum_{(\alpha, \beta)} \sigma_{\alpha\beta}$ are normalization factors so that μ_{α} 's sum up to one and $\sigma_{\alpha\beta}$'s sum up to one.

4.3 Data preprocessing

In order to generate the results in this Chapter, a series of preprocessing is applied to the volumetric data to obtain an accurate surface density $\mu(x)$ and oriented vector field $n(x)$, which we explain the steps here. The probabilistic geometry and appearance of all scenes is learned using the GPU implementation of CPVM [80, 81] whose theory is briefly described in Appendix A.2. The visibility information plays an important role during online Bayesian learning. Surfaces are resolved with increasing accuracy as new views become available. The appearance distributions in empty cells

outside of objects fail to explain the intensity of background object, causing occlusion density ρ to converge to very low (ideal) values on empty (but visible) space. On the other hand, ρ converges to infinity (ideal) at surface locations. As the value of the occlusion density increases near surfaces, cells located inside of objects are updated with decreasing weight. Hence, after convergence of the model, the information inside objects could be meaningless due to ambiguous regions that were learned during early stages of the online training process. The erroneous information inside objects can potentially hinder the accuracy of surface extraction. These cells can be detected using the visibility information already present in the model. For each cell, a measure of its visibility, $vis_score(x)$, is computed using a number of viewing directions [124]. Cells with low $vis_score(x)$ are detected and the corresponding ρ is set to 0 to eliminate online learning artifacts. In practice, the viewing directions can be selected from the cameras used during learning or simply by defining a canonical set of directions, *e.g.*, samples from a unit sphere. For scenes with poorly distributed cameras, the former method might be preferred to avoid possibly unreliable visibility computation due to poorly resolved surfaces.

Oriented vector field $n(x)$ is computed using the gradient information of the occlusion density $\nabla\rho(x)$. The gradient direction is computed via convolving first order derivatives of the three-dimensional gaussian kernel with the volume. For all experiments, six oriented kernels were used and their responses interpolated into an estimate of $\nabla\rho(x)$. Closed objects in the scenes contain mostly empty space, therefore when computing surface normals from the gradient direction, there is an orientation ambiguity. In order to have surface normals oriented consistently towards the outside of objects, the normal directions are oriented to the hemisphere that yields the maximum visibility.

We get a good approximation to the the surface density $\mu(x)$ by combining occlusion density $\rho(x)$, $vis_score(x)$ and gradient magnitude of occlusion density $\|\nabla\rho(x)\|$

$$\mu(x) = \rho(x) \times vis_score(x) \times \|\nabla\rho(x)\| \quad (4.7)$$

which has a simple explanation, *i.e.*, the desired surface should pass through volume of high occlusion density $\rho(x)$ and high visibility $vis_score(x)$. Moreover, $\|\nabla\rho(x)\|$ is included to increase robustness to outliers such as isolated cells (in the air) with high occlusion density.

4.4 Implementation

The implementation of GSSD is straightforward, and similar to the implementation of SSD described in Section 2.8 of Chapter 2. Given the surface density $\mu(x)$ and normal vector field $n(x)$, we estimate $f(x)$ by minimizing the discrete energy (4.4), which reduces to the solution of a sparse linear system $AF = b$. The linear system is solved using a cascading multi-grid method, along with a Jacobi preconditioned conjugate gradient solver. That is, the problem is first solved on a much lower depth of the octree than desired, then the solution obtained at a given depth is interpolated to the next depth; and used to initialize the iterative solver [78]. The iterative solver accumulates the $K \times 1$ result vector of the product $AF^{(k)}$ and the $K \times 1$ right hand side vector b by following sums:

$$\begin{aligned}
AF^{(k)} &= \frac{1}{W_1} \sum_{\alpha} \Phi_{\alpha} \left\{ \Phi_{\alpha}^T F^{(k)} \right\} \mu_{\alpha} \\
&+ \frac{\lambda_1}{W_1} \sum_{\alpha} \nabla \Phi_{\alpha} \left\{ \nabla \Phi_{\alpha}^T F^{(k)} \right\} \mu_{\alpha} \\
&+ \frac{\lambda_2}{W_2} \sum_{(\alpha, \beta)} \nabla \Phi_{\alpha} \left\{ \nabla \Phi_{\alpha}^T F^{(k)} - \nabla \Phi_{\beta}^T F^{(k)} \right\} \sigma_{\alpha\beta} \\
&- \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} \nabla \Phi_{\beta} \left\{ \nabla \Phi_{\alpha}^T F^{(k)} - \nabla \Phi_{\beta}^T F^{(k)} \right\} \sigma_{\alpha\beta} \\
b &= \frac{\lambda_1}{W_1} \sum_{\alpha} \nabla \Phi_{\alpha} n_{\alpha} \mu_{\alpha}
\end{aligned} \tag{4.8}$$

where the first two terms are contributions from the data terms, and the last two are contributions from the regularization term after expanding the finite difference. Φ_{α} is a $K \times 1$ sparse vector with 8 non-zero entries corresponding to the weights defined in equation (4.5), and $\nabla \Phi_{\alpha}$ is a $K \times 3$ sparse gradient matrix with 24 non-zero entries corresponding to the weights defined in equation (4.6).

Once the signed distance function $f(x)$ is estimated, a polygonal approximation (mesh) of the isolevel zero is constructed using the Dual Marching Cubes (DMC) algorithm [11]. Since CPVM produces color information represented as a volume texture at high resolution, estimating surface colors reduces to volume texture evaluation, and they are represented as polygon mesh vertex attributes. This approach avoids most of the pitfalls of texture mapping from images where occlusion and extreme warping produce unsuitable textures [113].

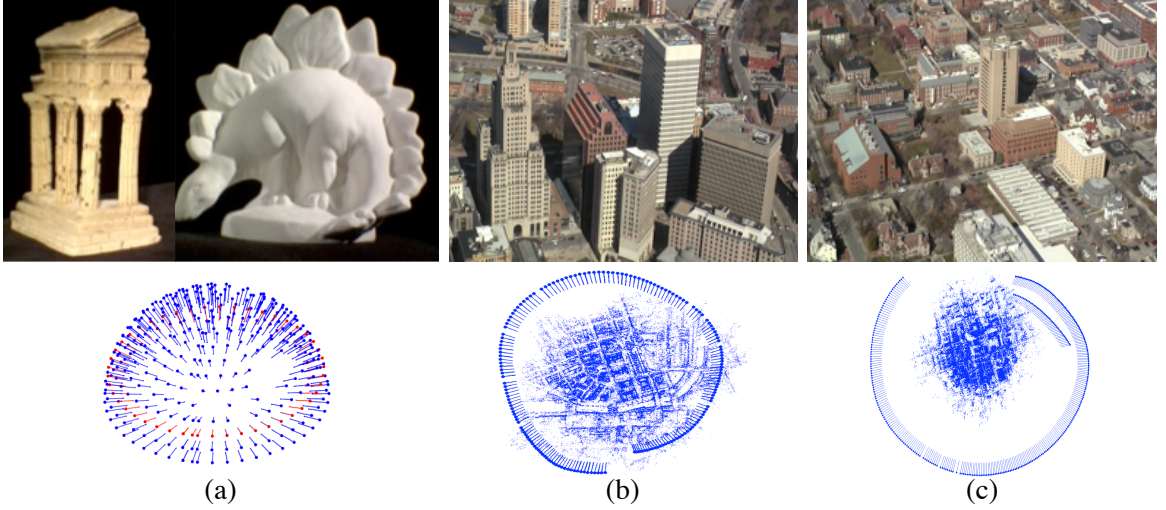


Figure 4.3: Sample frames and camera locations: (a) Middlebury dataset [114], “Full” set of cameras in blue, “Ring” in red; (b) “Downtown” site; (c) “SciLi” site.

4.5 Results

The proposed GSSD algorithm is evaluated as part of a multi-view stereo reconstruction framework (CPVM [80, 81]+GSSD). A series of experiments are conducted to evaluate the framework in three aspects. First, it is examined how well the CPVM estimates the underlying geometry. Then, the quality of the reconstructed surfaces using GSSD is assessed. Finally, comparisons to other MVS reconstruction frameworks are presented. PMVS [38] with Poisson Reconstruction [72] (PMVS+Poisson) is used as a baseline framework because it achieves high ranking in the Middlebury benchmark [114] and is applicable to large scale scenes. The results focus on the reconstruction of 3D models from aerial imagery where both the underlying geometry and the reconstructed surfaces are shown to be superior to PMVS+Poisson. In addition, numerical accuracy of the proposed algorithm is demonstrated on the Middlebury benchmark where highly competitive results are obtained.

The dataset consists of images from two challenging urban sites and two benchmark models. Fig. 4.3a presents a sample image from the Temple and Dino objects that are part of the Middlebury dataset [114]. Both objects were reconstructed using both the “Full” set of cameras, (312 views for Temple, 363 views for Dino), and the “Ring” set (47 views for Temple, 48 view for Dino). Fig 4.3b and 4.3c correspond to images from two urban sites in Providence, RI, USA (publicly available in [130]). The urban sites, here referred to as “Downtown” and “SciLi”, cover an area of approximately $(500\text{m}) \times (500\text{m})$ and have an approximate resolution of 30 cm/pixel. The camera

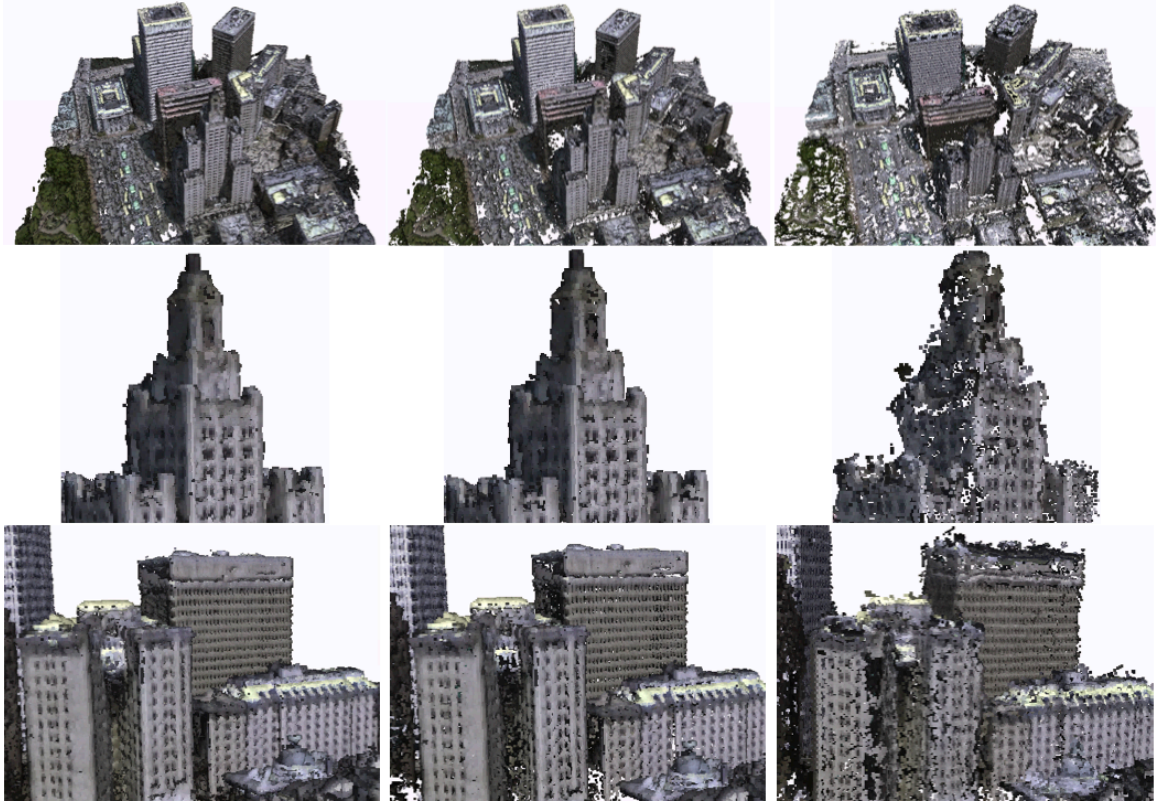


Figure 4.4: Point cloud visualization of “Downtown” (from left to right column wise): point cloud using the top (4.7) 70% of CPVM samples; point cloud using the top (4.7) 10% of CPVM samples; PMVS point cloud.

paths follow a circle around each site; the size of all images is 1280x720 pixels, 174 views are available for “Downtown” and 337 for “SciLi”. The camera matrices for all aerial image sequences were obtained using the Bundler software [37]. The resulting models were trained in approximately 2 hours for “Downtown” and 3 hours for “SciLi” and both contain roughly 30 million leaf cells.

Urban Scenes

This section begins with an examination of the quality of the surface density learned in the CPVM. In order to make comparisons to PMVS easy to visualize and fair, locations (octree leaf cell centroids) are sampled from CPVM to generate a point cloud. The cell locations are filtered using the surface density (4.7) as a threshold criteria. The point clouds are visualized in Fig. 4.4, where the full “Downtown” scene as well as details are shown for the top 70% and 10% of the sampled cells, together with the PMVS point cloud. Comparisons in Fig. 4.4 demonstrate three clear advantages

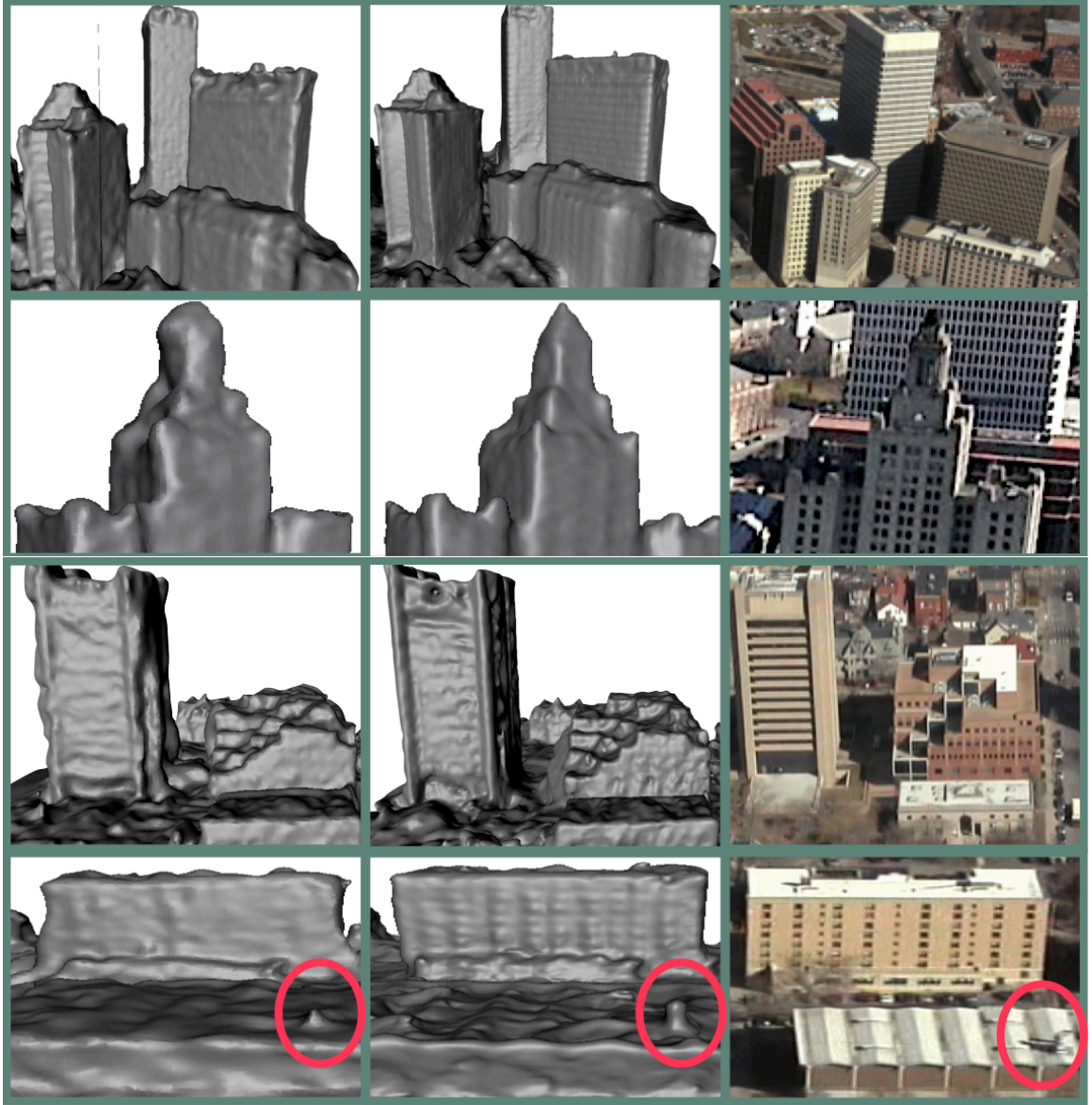


Figure 4.5: Reconstructed surfaces from the “Downtown” (top two rows) and “SciLi” (bottom two rows) sites, using PMVS+Poisson (leftmost column) and CPVM+GSSD (mid column). Sample images for the corresponding scenes (rightmost column).

of CPVM over PMVS. Namely, (a) both systems do well on planar textured surfaces, however in regions of high curvatures, CPVM produces much more accurate results than PMVS, *e.g.*, edges of buildings in bottom row; (b) CPVM is able to resolve details at higher resolution than PMVS, *e.g.*, pillars and tip of the building in middle row; (c) information is very dense in the probabilistic model, *e.g.*, the point cloud of CVPM even with 10% of the surface voxels is denser than PMVS.

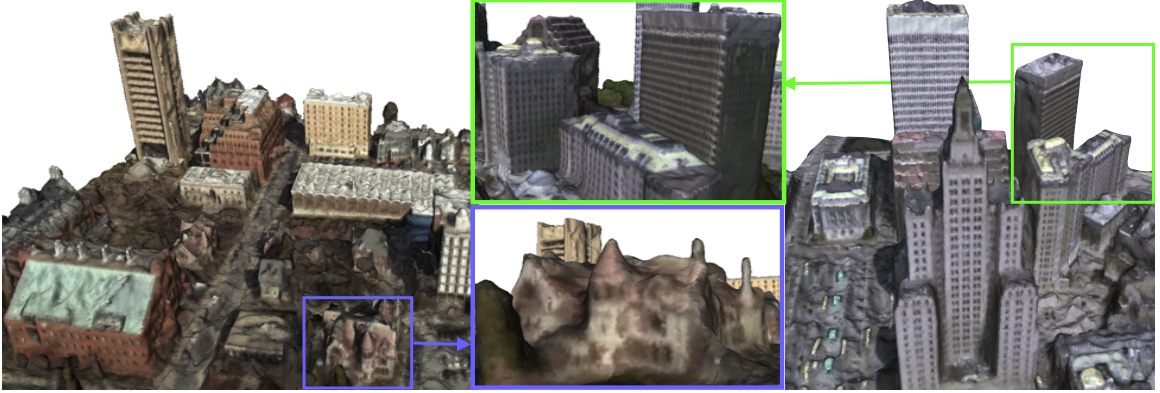


Figure 4.6: Renderings of 3D models (colored mesh) of the “SciLi” (left) and the “Downtown” (right) sites reconstructed by the framework CPVM+GSSD. Note the high resolution texture on buildings.

Fig. 4.5 presents the comparisons of surfaces reconstructed for both the “SciLi” and the “Downtown” sites using PMVS+Poisson and CPVM+GSSD. The figure demonstrates that the proposed method produces pleasing surfaces while staying faithful to the high detail information available in the data, notice the sharpness of building walls and corners, and ability to capture small details such as roof-top pipes and tiny windows, as shown in bottom row. The produced surfaces are well defined and crisp, even in regions of high surface curvature, as seen in second row. All these qualities are a direct result of the variational formulation of equation (4.4). The first two data terms make sure that the surface pass through the highly surface-like regions while the third term forces certain smoothness constraints. It has been observed that when PMVS and CPVM have difficulties modeling the appearance and underlying geometry of highly specular surfaces, the reconstruction algorithms can not recover accurate surfaces (see supplementary video¹ for an example).

Fig. 4.6 shows renderings of the colored meshes. Notice the high resolution and completeness of the appearance information such as chimney and cone of the highlighted house (in purple), as well as texture detail on buildings (in green). All meshes were obtained using GSSD parameters $\lambda_1 = 1$ and $\lambda_2 = 4$ (see (4.4)). The running time for GSSD was roughly 12 minutes using an Intel Xeon @2.9 Ghz. Both urban scenes contained approximately a million cells with significantly high μ .

¹<http://vimeo.com/45316105>

Middlebury Evaluation Performance

Quantitative evaluations reported in the Middlebury benchmark [114] under the submission name “Generalized-SSD” show that the proposed method is highly competitive both in terms of accuracy(mm) and completeness(%). Table 4.1 presents results for a selection of methods obtained from [114]. In [124], Crispell *et al.* reconstruct a surface by extracting the isosurface of the implicit function *vis_score* using Marching Cubes. The superiority of GSSD compared to this method is clear in both Temple and Dino (Ring) datasets (see Figure 4.2 for a qualitative comparison). Also included in the table are the results for PMVS+Poisson obtained in [38]. At the time of writing, this method achieved some of the best results at [114] including top scores in Full Dino. The framework CPVM+GSSD performs competitively, especially in the Full Temple dataset. It should be noted that most of the best performing algorithms reported at [114], including [38], take advantage of the fact that reasonable silhouettes can be extracted. This fact is not exploited in this chapter, as the main focus is on applicability to realistic urban scenes.

Table 4.1: Quantitative evaluation taken from the Middlebury benchmark [114]. The first value corresponds to accuracy (mm) and the second to completeness (%). All meshes were obtained using GSSD parameters $\lambda_1 = 1$ and $\lambda_2 = 1$ (see (4.4))

	Temple				Dino			
	Full		Ring		Full		Ring	
PMVS + Poisson [38]	0.54	99.3	0.55	99.1	0.32	99.9	0.33	99.6
CPVM + GSSD (proposed)	0.53	99.4	0.81	95.8	0.55	98.1	0.6	96.0
CPVM + Marching Cubes [124]	-	-	1.89	92.1	-	-	2.61	91.4

4.6 Conclusion

The main contribution of this chapter is the *generalization* of the SSD which was originally designed in Chapter 2 to reconstruct surfaces from a finite set of oriented points to allow reconstruction of surfaces from probabilistic occupancy volumes. As for evaluation, we first used CPVM [80, 81](Appendix A.2) method to estimate probabilistic volumes from aerial imagery of large scale urban scenes, and then the proposed GSSD (4.2) method to reconstruct highly detailed surfaces from probabilistic volumes. The advantages of this framework (CPVM+GSSD) were shown via comparisons to state of the art methods, namely PMVS [38] and Poisson surface reconstruction [72], chosen due to their popularity and software availability. In addition to experiments in aerial scenes,

the framework was also tested in the Middlebury evaluation [\[114\]](#) where it performed competitively in terms of accuracy and completeness. Most remarkably, it achieved results on par with algorithms that exploit high resolution silhouettes. An important future direction is the application of GSSD to other volumetric data such as those obtained in medical imaging.

Chapter 5

SSD from Incomplete Surface Normal Data (SSD-Inc)

5.1 Introduction

Most surface reconstruction methods reviewed in Sections 1.3.3 and 2.1 perform the reconstruction either from a set of *unoriented* point sets [63, 97, 100, 131, 132] or require good *oriented* normal estimates [1, 54, 64, 65, 69, 71, 72] associated with these points. Oriented surface normals play a critical role in surface reconstruction by providing inside/outside information near the surface. As a result, the methods that use normal information are typically more robust to noise and outliers [53] than the methods that do not use this information. A notable exception is the spectral segmentation approach [63] which has shown robust to noise and outliers without the use of normals. However, the method may lead to poor approximation to the surface in regions of missing data due to coarse Delaunay-based discretization. Even though there are measuring techniques, such as photometric stereo [31], capable of explicitly measuring surface normals, most sensors do not provide this information; in which case, point normals are estimated from the point locations using local principal component analysis (PCA) [67] followed by orienting them toward the sensor direction. This routine is typically implemented efficiently with a kd-tree data structure [133]. However, normals estimated this way can get highly unreliable in regions of missing or irregularly sampled data or near thin features; as a result, they often lead to inaccurate surface reconstructions. On the other hand,

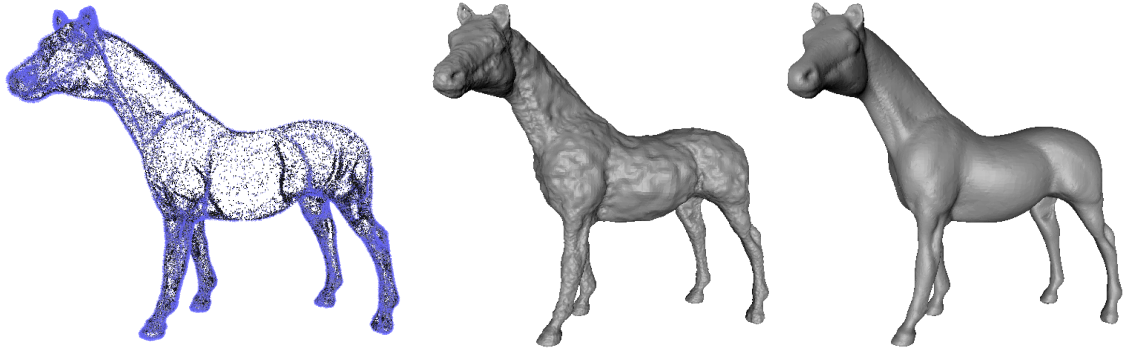


Figure 5.1: Left: Irregularly sampled input point cloud with extremely inaccurate normal vectors. The point cloud is obtained by adding synthetical noise to the directions of the normals of the point set originally taken from [86]. Center: Surface reconstructed with SPR [65], trusting the given normal vectors. Right: Surface reconstructed with the proposed method, which simultaneously estimates surface normals.

most optical measuring systems implicitly or explicitly provide the coordinate of the sensor location through calibration information. The vector from a measured point to the sensor location is called the “line of sight” associated with the point. A useful property of line of sight vectors is that they are consistently oriented toward the outside (or inside) of the object being scanned. Even though line of sight vectors are reliable and easily accessible, they are either not used by surface reconstruction methods or simply discarded after point normals are oriented by the PCA routine described above.

The methods designed to fuse multiple depth maps into a single volumetric model such as [66, 134, 135] successfully exploit this information retrieved from the scanner. Similarly, the surface reconstruction methods based on discrete segmentation approaches [129, 136] use the line of sight information. While segmentation approaches based on graph-cuts or s-t cuts allow for flexible energy functions and exact solutions, they are not applicable to large scenes due to their high memory requirements. An additional post-processing step is required in [129] to remove artifacts due to regular grid discretization [137]. More recent implicit surface reconstruction methods based on variational global optimization [1, 54, 65] have proven efficient and robust to noise [53]. However, these methods, also the SSD introduced in Chapter 2, rely on good normal estimates. Formally, these methods are concerned with the problem of reconstructing a surface S defined by an implicit equation $S = \{x : f(x) = 0\}$, approximating a finite set of oriented points $\mathcal{D} = \{(p_1, n_1), \dots, (p_N, n_N)\}$, where p_i is as surface location sample, and n_i is the corresponding surface normal sample oriented towards the outside of the object. They typically reduce the problem to solving a set of linear equations to estimate f , and report excellent reconstruction results. We regard this type of data $\mathcal{D}$ where one

can estimate the result from in one step (such as by solving a linear system) as “complete” data. If more information is available, the problem becomes simpler as expected. The definition is more closely related to what part of data is observed or not. For example, $\mathcal{D}_{\text{inc}} = \{p_1, \dots, p_N\}$ can be regarded as “incomplete” data if the scanning sensor does not have the capability to measure surface normals explicitly, *i.e.*, surface normals are not observed.

The main contribution of this chapter is the introduction of a new variational formulation for the problem of reconstructing a watertight surface S defined by an implicit equation $S = \{x : f(x) = 0\}$ *directly* from a finite set of *partially* oriented points

$$\tilde{\mathcal{D}} = \{(p_1, \tau_1), \dots, (p_N, \tau_N)\},$$

where p_i is as surface location sample, and τ_i is a unit vector oriented towards the outside of the object. Note that τ_i can be an extremely noisy surface normal sample as well as a normalized line of sight vector associated with p_i . The only assumption on this set is that the vectors must be consistently oriented. We refer this set of vectors as *incomplete* surface normals, and the dataset consisting of points and *incomplete* surface normals as *partially* oriented points for the rest of the chapter.

The proposed method uses incomplete surface normal information to constrain the space of solutions by measuring how well an inside/outside labeling defined by the implicit equation agrees with incomplete surface normal data and likelihood of the surface resulting from positional data and this labeling. The method is based on Majorize-Minimize (MM) technique which leads to an iterative non-linear minimization routine that is guaranteed to converge to a fixed point by design [138]. In the first step, “feasible” surface normals are estimated from the current estimate of the implicit function, and in the second step the surface estimate is refined using points and feasible normals. Fortunately, both alternating steps are quite efficient, and the second step reduces to solving SSD introduced in Chapter 2. Figure 5.1 shows one example of surface reconstructed by the proposed method in comparison with surface reconstructed by SSD in Chapter 2. Note the proposed method faithfully reconstructs the surface despite the fact that the input point cloud lacks accurate surface normals.

5.2 Constrained Energy formulation

Given a finite set of *partially* oriented points $\tilde{\mathcal{D}} = \{(p_1, \tau_1), \dots, (p_N, \tau_N)\}$, where p_i is as surface location sample, and τ_i is a unit vector oriented towards the outside of the object, and associated with p_i . We ideally want to estimate f by means of minimizing the following constrained energy:

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \mathcal{E}(f) = \frac{1}{N} \sum_{i=1}^N f(p_i)^2 + \frac{\lambda_1}{N} \sum_{i=1}^N (\|\nabla f(p_i)\| - 1)^2 + \frac{\lambda_2}{|V|} \int_V \|Hf(x)\|^2 dx, \\ \text{subject to} \quad & \frac{\nabla f(p_i)^\top \tau_i}{\|\nabla f(p_i)\|} > \epsilon \quad \text{for } i = 1, \dots, N, \end{aligned} \quad (5.1)$$

$\nabla f(x)$ is the gradient of $f(x)$, $Hf(x)$ is the Hessian of $f(x)$, V is a bounding volume, $\{\lambda_1, \lambda_2\}$ are regularization parameters, and $0 < \epsilon < 1$ is an error tolerance parameter.

5.3 Majorize-Minimize (MM) technique

Analyzing the objective (5.1), we first notice that *incomplete* normals τ_i only appear in constraints. If ignored, it would be the same as estimating f with only point coordinates p_i . Even though it is a harder problem to solve than the original problem since all inside/outside information provided by τ_i 's is lost, trying to design an optimization routine gives several insights, in particular, with the use of Majorize-Minimize (MM) technique. MM works by finding a surrogate function that majorizes the objective function to be minimized. Optimizing the surrogate function drives the objective function downward until a local optimum is reached, and by design, MM iterations are guaranteed to converge to a local minimum or a saddle point of the original objective function [138]. The choice of the surrogate function is critical as optimizing the surrogate must be simpler than optimizing the original problem. Formally, in order to minimize $\mathcal{L}(\theta)$ with an MM procedure, we need a surrogate function $\mathcal{S}(\theta, \tilde{\theta})$ such that:

$$\begin{aligned} \mathcal{S}(\theta, \theta) &= \mathcal{L}(\theta) \\ \mathcal{S}(\theta, \tilde{\theta}) &\geq \mathcal{L}(\theta), \quad \theta \neq \tilde{\theta} \end{aligned} \quad (5.2)$$

If $\theta^{(k+1)}$ denotes the minimum of the surrogate $\mathcal{S}(\theta, \theta^{(k)})$, the MM procedure forces $\mathcal{L}(\theta)$ downhill which can be seen through the relations

$$\mathcal{L}(\theta^{(k+1)}) \leq \mathcal{S}(\theta^{(k+1)}, \theta^{(k)}) \leq \mathcal{S}(\theta^{(k)}, \theta^{(k)}) = \mathcal{L}(\theta^{(k)}) \quad (5.3)$$

which follow directly from the construction (5.2), and definition of $\theta^{(k+1)}$. Given an estimate $\theta^{(0)}$, a typical MM minimization procedure is alternated between

- building the surrogate function $\mathcal{S}(\theta, \theta^{(k-1)})$
- updating the estimate $\theta^{(k)} \leftarrow \arg \min_{\theta} \mathcal{S}(\theta, \theta^{(k-1)})$

for $k = 1, 2, 3, \dots$ until a fixed point is found. The choice of $\theta^{(0)}$ matters as in every non-linear minimization problems. Without loss of generality, it is not necessary to find the absolute minimum of the surrogate function, *i.e.*,

$$\mathcal{S}(\theta^{(k+1)}, \theta^{(k)}) \leq \mathcal{S}(\theta^{(k)}, \theta^{(k)}) \quad (5.4)$$

is only needed. One advantage of MM becomes apparent when dealing with box constraints over variables to be estimated. In other words; if $\theta^{(k)}$ goes out of the feasible space at any k , it is simply replaced with an arbitrary value from the feasible space (typically with the closest to $\theta^{(k)}$), and the surrogate function is built from the feasible value.

5.4 Surrogate Objective

Going back to our original problem, we are looking to find a surrogate objective that majorizes the objective (5.1). Consider the following objective as a surrogate candidate

$$\mathcal{Q}(f, g) = \frac{1}{N} \sum_{i=1}^N f(p_i)^2 + \frac{\lambda_1}{N} \sum_{i=1}^N \left\| \nabla f(p_i) - \frac{\nabla g(p_i)}{\|\nabla g(p_i)\|} \right\|^2 + \frac{\lambda_2}{|V|} \int_V \|Hf(x)\|^2 dx, \quad (5.5)$$

with first and third terms are already the same as the original objective. Therefore, we only need to verify that the second term satisfies the majorization conditions (5.2). Since the second terms are sums of individual terms in both objectives, this task reduces to showing that

$$q_i(f, g) \triangleq \left\| \nabla f(p_i) - \frac{\nabla g(p_i)}{\|\nabla g(p_i)\|} \right\|^2$$

majorizes

$$e_i(f) \triangleq (\|\nabla f(p_i)\| - 1)^2.$$

One can inspect that the equality condition in equation (5.2) is satisfied

$$\begin{aligned}
q_i(f, f) &= \left\| \nabla f(p_i) - \frac{\nabla f(p_i)}{\|\nabla f(p_i)\|} \right\|^2 \\
&= \nabla f(p_i)^\top \nabla f(p_i) - 2 \frac{\nabla f(p_i)^\top \nabla f(p_i)}{\|\nabla f(p_i)\|} + 1 \\
&= (\|\nabla f(p_i)\| - 1)^2 = e_i(f)
\end{aligned} \tag{5.6}$$

by construction. And the inequality condition in equation (5.2) is satisfied

$$\begin{aligned}
q_i(f, g) &= \nabla f(p_i)^\top \nabla f(p_i) - 2 \frac{\nabla f(p_i)^\top \nabla g(p_i)}{\|\nabla g(p_i)\|} + 1 \\
&\geq \nabla f(p_i)^\top \nabla f(p_i) - 2 \frac{\nabla f(p_i)^\top \nabla f(p_i)}{\|\nabla f(p_i)\|} + 1 = e_i(f)
\end{aligned} \tag{5.7}$$

since $\nabla f(p_i)^\top \nabla g(p_i) \leq \|\nabla f(p_i)\| \|\nabla g(p_i)\|$ by Cauchy-Schwarz inequality. These two relations conclude the proof that the objective (5.5) majorizes the original objective (5.1), and is a valid surrogate objective. That is, starting with an arbitrary $f^{(0)}$, convergence to a local minima is guaranteed by MM - provided that we have a method to minimize the surrogate objective $\mathcal{Q}(f, f^{(k-1)})$. Here we note that minimization of the surrogate objective is the same as solving SSD in Chapter 2 with

$$n_i = \frac{\nabla f(p_i)^{(k-1)}}{\|\nabla f(p_i)^{(k-1)}\|},$$

where $\nabla f(p_i)$ can be determined through finite differences (2.12, 2.13). However, the problem is highly non-linear, and without the use of inside/outside information given by τ_i 's, one should not expect to obtain useful results. In the next section, we introduce a practical algorithm where we can easily incorporate τ_i 's (the constraints of the original objective) into alternating steps of MM to ensure that the iterative procedure converges to a good local minima.

5.5 Solution and Algorithm

Equipped with the surrogate objective (5.5), we propose the following MM procedure to minimize the original objective (5.1):

1. Initialize $f^{(0)}$ to something arbitrary

2. Repeat steps (a) and (b) for $k = 1, 2, 3 \dots$ until convergence or maximum number of iterations are reached

(a) update $\mu_i^{(k)}$ for all $i = 1, \dots, N$,

$$\mu_i^{(k)} \leftarrow \begin{cases} \frac{\nabla f(p_i)^{(k-1)}}{\|\nabla f(p_i)^{(k-1)}\|}, & \text{if } \tau_i^\top \frac{\nabla f(p_i)^{(k-1)}}{\|\nabla f(p_i)^{(k-1)}\|} > \epsilon \\ \tau_i, & \text{otherwise} \end{cases}$$

(b) estimate $f^{(k)}$ as the (loose) minimizer of

$$f^{(k)} \leftarrow \arg \min_f \left\{ \frac{1}{N} \sum_{i=1}^N f(p_i)^2 + \frac{\lambda_1}{N} \sum_{i=1}^N \|\nabla f(p_i) - \mu_i^{(k)}\|^2 + \frac{\lambda_2}{|V|} \int_V \|Hf(x)\|^2 dx \right\},$$

The first part (a) of the alternating steps uses the old estimate of the function $f^{(k-1)}$ to get a “feasible” set of normals $\mu_i^{(k)}$. If $\frac{\nabla f(p_i)^{(k-1)}}{\|\nabla f(p_i)^{(k-1)}\|}$ already satisfies the constraint determined by τ_i , it is included into the “feasible” set, *i.e.*, constraint is not active. If not, it is replaced with τ_i itself, *i.e.*, constraint is active. The second part (b) uses points p_i and “feasible” normals $\mu_i^{(k)}$ to get a new function estimate $f^{(k)}$ which reduces to solving SSD in Chapter 2 with $n_i = \mu_i^{(k)}$. Note that an *exact* minimization of the SSD objective is not needed. Instead, a sufficient decrease is adequate for MM to work (see remark (5.4)).

Reconstruction from 2D cross-sectional curves

The MM algorithm described above has the advantage of decoupling the estimation of the feasible normals μ_i ’s from the estimation of the implicit function f . As a result, it can also be used, with a slight modification, to reconstruct surfaces from sets of (parallel or non-parallel) 2D *cross-sectional* curves which are typically obtained by image-processing outputs of medical cross-sectional imaging techniques or in 3D modeling from 2D drawings. In this case, the data is given as a composition of several cross-sectional curves

$$\tilde{\mathcal{D}} = \bigcup_j \tilde{\mathcal{D}}_j$$

where $\tilde{\mathcal{D}}_j$ is a set of *partially* oriented points associated with the j^{th} slit that crosses the shape. The normal τ_i associated with the point $p_i \in \tilde{\mathcal{D}}_j$ is *incomplete* because one can only measure the projection of the true surface normal onto the plane defined by the j^{th} slit. Therefore, the gradient

$\nabla f(p_i)$ vector at a point $p_i \in \tilde{\mathcal{D}}_j$ should be ideally spanned by the two perpendicular unit vectors: the partial normal τ_i associated with p_i , and the slit normal, say v_j , associated with $\tilde{\mathcal{D}}_j$. This geometric observation defines another set of constraints on f , *i.e.*,

$$\nabla f(p_i)^\top (\tau_i \times v_j) \approx 0 \quad (5.8)$$

in addition to the original constraints of the objective defined in (5.1). The slit normal v_j is either provided with $\tilde{\mathcal{D}}_j$ or can be easily estimated by fitting $\tilde{\mathcal{D}}_j$ a plane. Similar to the MM algorithm described in previous section, the constraints (5.8) are applied during the computation of feasible normals by replacing the step 2(a) of the algorithm with

$$\mu_i^{(k)} \leftarrow \begin{cases} \frac{P(\tau_i, v_j) \nabla f(p_i)^{(k-1)}}{\|P(\tau_i, v_j) \nabla f(p_i)^{(k-1)}\|}, & \text{if } \tau_i^\top \frac{P(\tau_i, v_j) \nabla f(p_i)^{(k-1)}}{\|P(\tau_i, v_j) \nabla f(p_i)^{(k-1)}\|} > \epsilon \\ \tau_i, & \text{otherwise} \end{cases}$$

where $P(\tau_i, v_j)$ is the 3×3 projection matrix defined by

$$P(\tau_i, v_j) \triangleq I_3 - (\tau_i \times v_j)(\tau_i \times v_j)^\top$$

with the 3×3 identity matrix I_3 . In words, the estimated gradient vector $\nabla f(p_i)^{(k-1)}$ at a point $p_i \in \mathcal{D}_j$ is first projected onto the planar space spanned by $\{\tau_i, v_j\}$, then normalized. Finally, the resulting vector is checked if it satisfies the orientation constraint determined by τ_i . If it does, it is included into the feasible set. If not, it is replaced with τ_i itself.

5.6 Implementation

We use same octree-based discretization described in Section 2.7 of Chapter 2. For a given octree depth d , an iteration k of MM reduces to solving a sparse linear system of the form $A^{(d)}F = b^{(k)}$, which is (loosely) solved using a Jacobi Preconditioned Conjugate Gradient (JPCG) solver [78]. We use the term loosely as a predetermined number of JPCG iterations are adequate for MM to work. This is no loss of generality as shown in Section 5.3. Note that the left hand side of the linear system $A^{(d)}$, is independent of k , and is only updated if the octree depth d changes. On the other hand, the right hand side of the linear system $b^{(k)}$ directly depends on k , and is updated

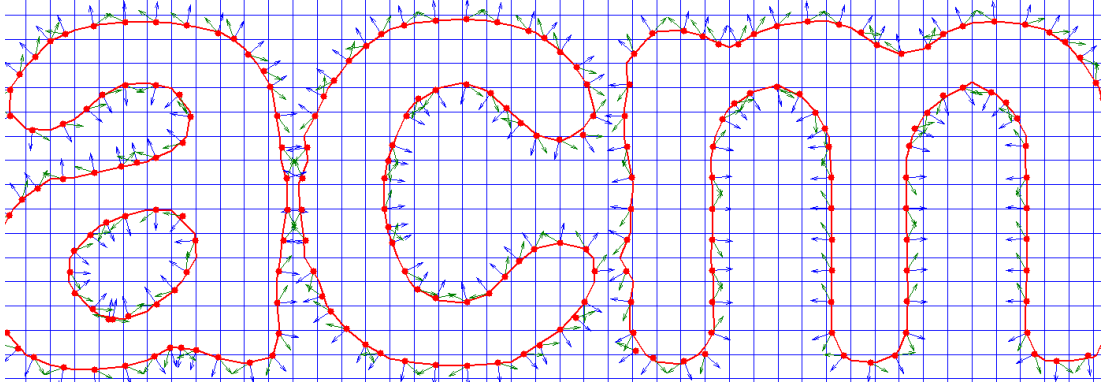


Figure 5.2: Curve estimation from synthetically generated points using proposed method. Red dots: Input point samples. Green needles: Input lines of sight. Blue needles: Recovered curve normals.

as $\mu_i^{(k)}$ vectors are updated. For an efficient implementation, iterations of the conjugate gradient solver are interrupted on a periodic schedule to update $\mu_i^{(k)}$ vectors, and the right hand side $b^{(k)}$ of the linear system accordingly. In order to improve the speed of convergence and the quality of the obtained solution, the problem is first solved on a much lower depth of the octree than desired, then the solution obtained at a given depth is interpolated to the next depth; and used to initialize the iterative solver. In other words, $A^{(d)}$ is updated once every time we go deeper in the octree depth; and for a given octree depth (d), $b^{(k)}$ is updated multiple times determined by the schedule of JPCG solver. In our implementation, we update $b^{(k)}$ at every 5 JPCG iterations.

5.7 Results

The proposed method is tested in 2D and 3D settings with synthetically generated data as well as real sensor data from an urban scene. Even though we are interested in point clouds and surfaces embedded in three dimensional space, our formulation is valid in space of arbitrary dimension. As a result, we first show a 2D result to better illustrate the difference between line of sight vectors and normal vectors. The curve estimation example from synthetically generated 2D points is given Figure 5.2, and shows the robustness of our method to extreme angles between line of sight vectors and normals vectors, in this example, up to $\frac{\pi}{2}$ radians.

The second example, Figure 5.3, is from an urban scene, namely “Downtown” site. The point cloud is acquired by sampling the voxel centroids of high occupancy density from a probabilistic volumetric model estimated by CPVM as described in Section 4.3 in which there is no elegant way

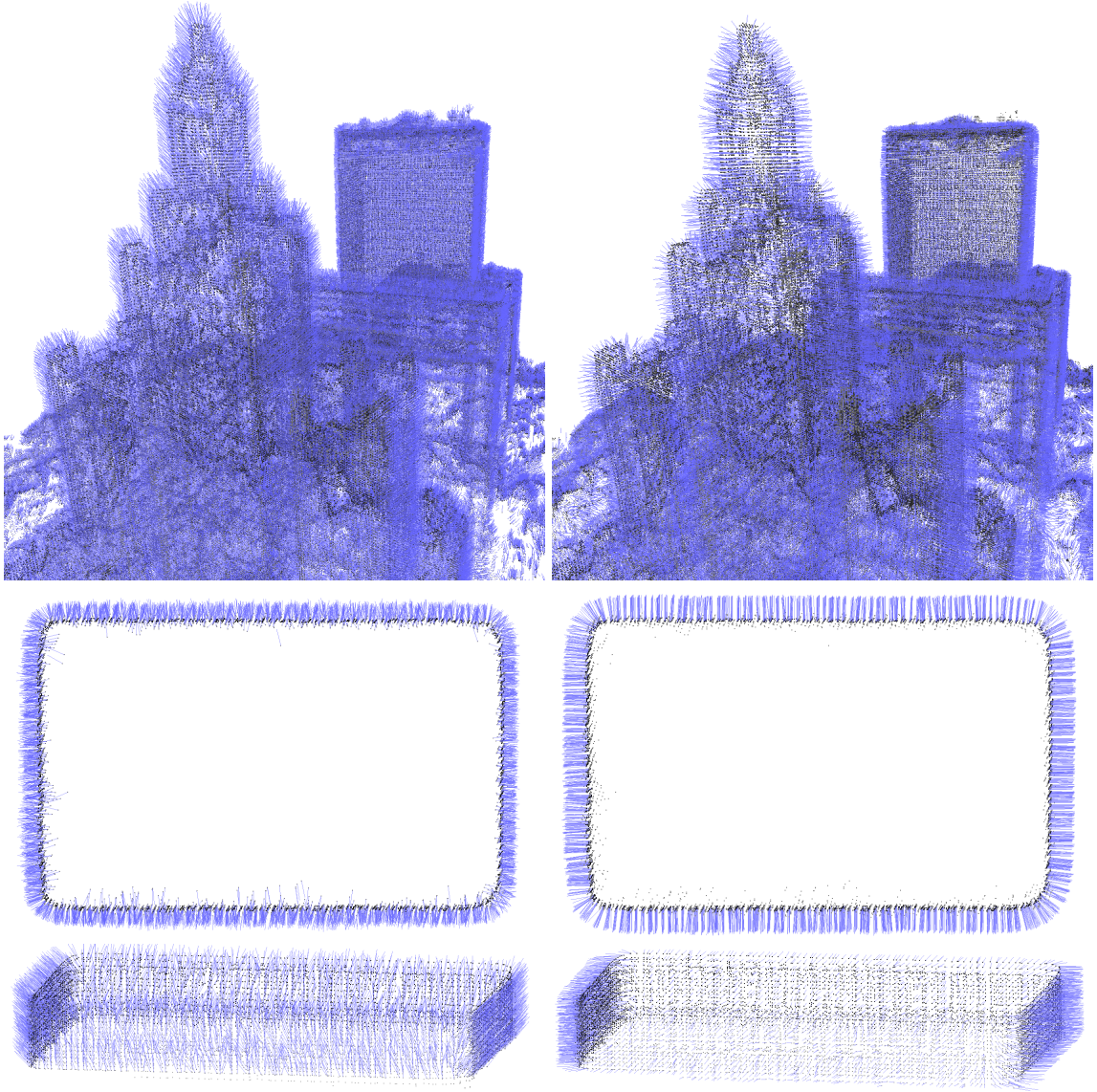


Figure 5.3: Reconstruction of “Downtown” site with zoom-in cross-sectional views. Left Column: Input point cloud rendered as points and line of sight vectors. Right column: Reconstruction of the site using the proposed method, rendered as points and recovered normals.

of estimating accurate surface normals. Running a 3D analogue of 2D discrete first order difference filters used in image processing (such as Sobel filters) over the volumetric density gives unreliable response near the surface due to unsigned nature of the density. On the other hand, it is possible to get properly oriented line of sight vectors for each (sampled) voxel by running a visibility weighted average over sensor rays that updated the density of that voxel. Since the scene is captured with a sensor placed on a helicopter, the line of sight vectors are looking upward, outside of the building

surfaces as expected. On the other hand, after the reconstruction, the proposed method properly recovers the surface normals from point samples and line of sight vectors. We use point cloud rendering of the reconstruction result in this example as opposed to mesh visualization for a fair comparison with the input point cloud, and to better illustrate the effectiveness of the proposed method.

5.8 Conclusion

This chapter first introduced a general variational formulation for the problem of reconstructing a watertight surface defined by an implicit equation from a finite set of *unoriented* points based on *unconstrained* Majorize-Minimize (MM) technique which led to an iterative non-linear minimization routine that is guaranteed to converge to a fixed point by design. However, due to highly nonlinear nature of the problem, we argued that the iterations would most likely converge to an unappealing local minimum without using any inside/outside information about the surface to be estimated. Then, we elegantly improved the formulation by incorporating *incomplete* normal data as constraints which prevented the routine from getting stuck at a poor local minimum. The resulting algorithm is practical and offers a good compromise between using the surface normals and not using any. The *incomplete* normals are not necessarily surface normals. They can be any set of vectors that are consistently oriented towards the outside (or inside) of the surface to be estimated. For example, (approximate) line of sight vectors that most 3D optical measuring devices provide these days together with positional data can be used for this purpose. Since the proposed method does not make strong assumptions on the type of *incomplete* normals, it can also be used, with slight modification, to reconstruct surfaces from sets of (parallel or non-parallel) 2D *cross-sectional* curves which appear in 3D modeling applications based on 2D drawings or sketches or are obtained by processing outputs of medical cross-sectional imaging techniques such as CT, MRI, PET, SPECT, and ultrasonography.

Chapter 6

SSD for Big Data - an Implementation on Octree Duals (SSD-Dual)

6.1 Introduction

In this chapter, we are concerned with the problem of reconstructing very large scenes consisting of hundreds of millions of oriented points. We introduce an octree-based dual-graph (which has the octree leaf centroids as vertices) discretization for the continuous objective (2.3) of SSD introduced in Chapter 2. Similar to octree-based primal-graph (which has the octree leaf corners as vertices) discretizations used in Chapter 2, dual discretizations reduce to the solution of sparse linear systems of equations. Figure 6.1 shows a dual-graph discretization in comparison to primal graph discretization depicted on a quadtree¹.

The original implementation of SSD uses a top-down octree construction strategy which first loads all points in memory, places them in a bounding box (which serves as the root of the octree), and then starting from the root leaf, recursively subdivides the octree as a function of points using a test condition, *e.g.*, a leaf is subdivided if it contains more than 1 points. This procedure is

¹Note that the discretizations are depicted on quadrees for the sake of clarity, whereas the text refers to the 3D case as formulations and algorithms generalize to any dimension.

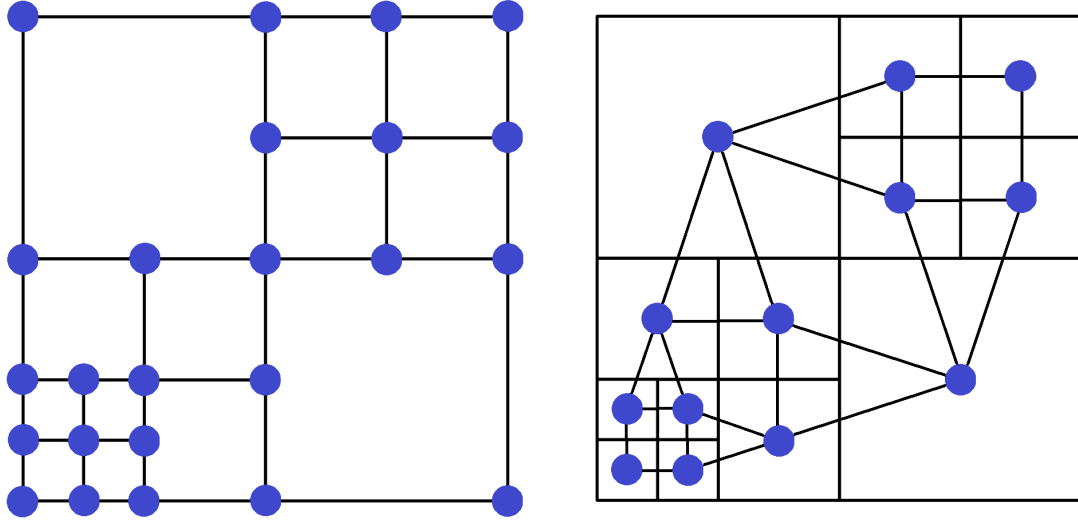


Figure 6.1: Quadtree based discretizations. Left: Primal graph. Right: Dual graph.

repeated for every leaf until the test fails or maximum (or user desired) depth is reached. Top-down octree construction strategy is very useful when dealing with uneven sampling and sensor noise in point measurements because the octree faithfully adapts to sampling density of the input cloud. In other words, it does not create an excessive number of cells, and as a result, surface reconstruction algorithms deal with less number of variables to estimate, and becomes less prone to variance in data. The disadvantage on the other hand is the requirement to load all points at once, and keep them all in memory until the octree construction is completed.

The main contributions² of this chapter are the derivation of sufficient statistics that are adequate to solve the linear system with less memory requirements than point cloud data, and the introduction of an online algorithm to accumulate these statistics from point cloud data, *i.e.*, points are never fully loaded or stored in memory. To operate online, an octree is first built in a bottom-up fashion rather than the recursive top-down strategy used in SSD. This step accumulates the sufficient statistics for each occupied leaf at the deepest level of the octree. Then, the sufficient statistics are expanded recursively from leaves to the internal nodes up to the root. Finally, the linear system is solved from top-down similar to the cascading multigrid approach used in SSD - which is estimating the function at a coarser level than desired using the coarser sufficient statistics, interpolating it to the next level, and refining the estimate by using the sufficient statistics of the next level, and repeating until the maximum or desired level is reached. Figure 6.2 shows a close-up comparison of

²The method proposed in this chapter is an extended version of the method originally developed by the author in 2013 as part of Kitware Inc.'s Super-Resolved 3D reconstruction project funded by DARPA.

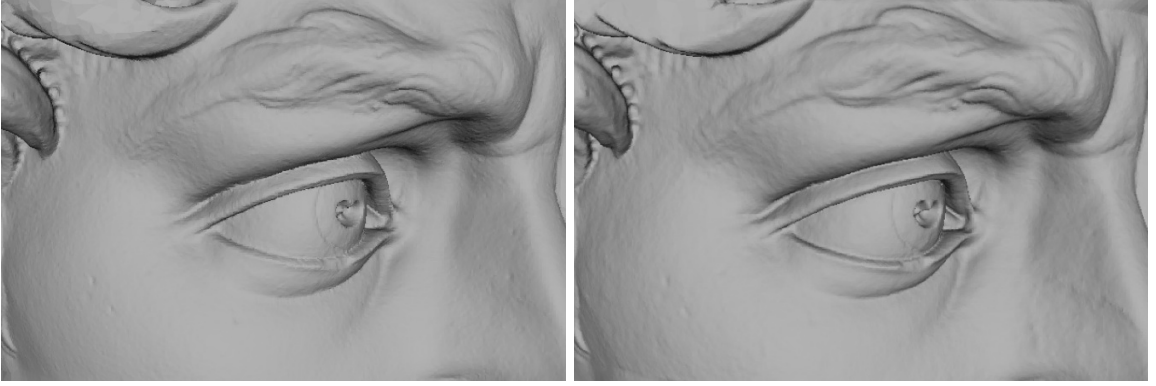


Figure 6.2: Surface reconstruction of David’s head using the proposed dual discretization (right), and using the original primal discretization (left). Left eye of the model is zoomed in to reveal visual similarities and differences between the results of two implementations. Similar behavior with minor visual differences is expected.

the surface reconstructed using the proposed dual graph discretization to the surface reconstructed using the original primal graph based discretization. Note that similar quality with minor differences is expected since both methods minimize the same objective using different discretization schemes.

6.2 Dual Formulation

In the original formulation of SSD introduced in Chapter 2, the primal graph (which has the octree leaf corners as vertices) represents the signed distance function $f(x)$, which is a piecewise trilinear function, *i.e.*, its value at a point x in the volume V is determined through $f(x) = \sum_{h=0}^7 w_h f_{\alpha_h}$ where $w_0, \dots, w_7$ are the trilinear coordinates of x in leaf α . The indices of the eight vertices of a leaf α are depicted in Figure 2.3. The gradient $\nabla f(x)$ and the Hessian $Hf(x)$ are determined through finite differences as given in equations (2.11), (2.12), and (2.14). With an alternative formulation, SSD can be implemented using the dual graph (which has the octree leaf centroids as vertices). In this case, the dual graph represents the signed distance function $f(x)$ which is no longer a piecewise trilinear function but it can be made piecewise linear by introducing an auxiliary vector field $v : V \rightarrow \mathbb{R}^3$ such that

$$v(x) = \nabla f(x) . \quad (6.1)$$

Similar to the discrete definition of gradient (2.12) in SSD, we define $v(x)$ as a piecewise constant function, one vector per octree leaf. That is, the value of $v(x)$ at a point x in the volume enclosed

by leaf α is determined through

$$v(x) \approx v_\alpha , \quad (6.2)$$

and the value of $f(x)$ at the same point is determined through

$$f(x) \approx f_\alpha + v_\alpha^\top (x - x_\alpha) \quad (6.3)$$

where x_α is the centroid location of leaf α , f_α is the value of the centroid. By taking the piece-wise definitions of f and v into account, the integrals and sums of energy function $\mathcal{E}(f)$ of (2.3) conveniently become finite sums over octree leaves:

$$\begin{aligned} \mathcal{E}(f) \approx & \underbrace{\frac{1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} (f_\alpha + v_\alpha^\top (p_i - x_\alpha))^2 + \frac{\lambda_1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} \|v_\alpha - n_i\|^2}_{\text{data terms}} + \underbrace{\frac{\lambda_2}{W} \sum_{(\alpha, \beta)} \|v_\alpha - v_\beta\|^2 \sigma_{\alpha\beta}}_{\text{regularization term}} \\ & \text{(subject to } v = \nabla f) \end{aligned} \quad (6.4)$$

where α^* represents the set of indices of points that are enclosed by leaf α . The regularization term is a sum over the pairs of octree leaves α and β that share a common face as depicted in Figure 2.5. The associated weight $\sigma_{\alpha\beta}$ is determined from the octree subdivision, *i.e.*, $\sigma_{\alpha\beta} = \frac{a_{\alpha\beta}}{b_{\alpha\beta}}$, where $a_{\alpha\beta}$ is the area of the common face and $b_{\alpha\beta}$ is the Euclidean distance between the centroids of these leaves. $W = \sum_{(\alpha, \beta)} \sigma_{\alpha\beta}$ is a normalization factor so that $\sigma_{\alpha\beta}$'s sum up to one.

Minimizing (6.4) subject to $v = \nabla f$ is a constrained least squares problem, and handling of which could be complicated with standard methods since f and v are both unknown. Instead, we add a penalty term to account for this constraint

$$\begin{aligned} \tilde{\mathcal{E}}(f, v) = & \frac{1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} (f_\alpha + v_\alpha^\top (p_i - x_\alpha))^2 + \frac{\lambda_1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} \|v_\alpha - n_i\|^2 + \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} \|v_\alpha - v_\beta\|^2 \sigma_{\alpha\beta} \\ & + \underbrace{\frac{\lambda_3}{W} \sum_{(\alpha, \beta)} \{(f_\alpha + v_\alpha^\top (x_\beta - x_\alpha) - f_\beta)^2 + (f_\beta + v_\beta^\top (x_\alpha - x_\beta) - f_\alpha)^2\} \sigma_{\alpha\beta}}_{\text{penalty (coupling) term}} . \end{aligned} \quad (6.5)$$

Similar to the original discretization, minimizing (6.5) is now an unconstrained least squares problem with linearly parametrized unknowns f and v which can be jointly estimated by solving a sparse

linear system of the form

$$A F = b, \quad (6.6)$$

where $F = (w_\alpha : \alpha \in \Lambda)$ is a $4K$ -dimensional column vector formed with 4-dimensional entries

$$w_\alpha \triangleq \begin{pmatrix} f_\alpha \\ v_\alpha \end{pmatrix}.$$

6.3 Sufficient Statistics

One way to solve a sparse linear system $A F = b$ that arise from a quadratic objective defined by primal (2.3) or dual formulation (6.5) is to first accumulate the non-zeros of matrix A and the vector b by traversing octree data structure which involves traversing every leaf, traversing every point within the leaf, and traversing every directed neighbor of the leaf. Once this is done, matrix A and the right hand side vector b can be passed to a direct sparse solver or to an iterative solver. The latter is typically preferred because A can still be very big. Another advantage of iterative solvers (such as conjugate gradient method) is that they do not even require the whole matrix A stored in the memory, instead they only need the result vector of the product $A F^{(k)}$ for every iteration k which can be calculated by a traversal similar to the one done for accumulation. This strategy was used in original implementation of the SSD introduced in Chapter 2 with great reduction in memory usage. However, since a traversal over the whole point set has to be done for every single iteration, time consumption becomes a problem specially for large scenes consisting of millions of points. On the other hand, building the matrix A from scratch every time octree changes could be a problem in terms of memory since non-zeros of A quickly grows as octree depth increases for very large scenes. This is clearly a trade-off between space and time usage which needs to be chosen based on available resources.

In this section, we discuss a compromise between keeping the whole matrix A in memory or not keeping it at all. Matrix A and the right hand side vector b contain all the information, *complete* sufficient statistics, that is required to minimize the objective (6.5). In detail, matrix A contains two types of information: one comes from the data terms A_d which depends on point cloud data, and

the other comes from the non data terms (regularization and coupling terms) $A_{\bar{d}}$ which depends on octree topology. Matrix A can be written as

$$A = A_d + A_{\bar{d}}$$

by construction. In general (also as mentioned in Section 2.8), neither A_d nor $A_{\bar{d}}$ has to be built or kept in memory to solve the linear system since an iterative solver only needs the result of matrix-vector product not the matrix itself. However, removing A_d from memory, requires that the point cloud data is kept in memory. Similarly, removing $A_{\bar{d}}$ from memory, requires that the octree is kept in memory. The latter (keeping octree in memory) is desirable, since the octree supports the signed distance function, and its topology information is needed for later stages of the framework in particular for isosurface extraction (contouring) algorithm. The former (keeping point cloud data in memory), on the other hand, is not desirable (or needed) after octree and A_d are built which is the strategy we advocate in this chapter. That is, *non-data* statistics $A_{\bar{d}}$ is neither built explicitly nor kept in memory. On the other hand, *data* statistics A_d is built independently and kept as symmetric block matrices - one per (occupied) octree leaf. In addition, since second data term is not homogenous, it is responsible for the right hand side vector b of the sparse linear system which we also keep as vectors - one per (occupied) octree leaf.

A_d is a diagonally dominant sparse matrix consists of non-zero block matrices, the form of which can be derived by expanding the data terms of dual energy (6.5), and taking the derivatives with respect to unknowns f and v . In detail, we first rewrite the first data term of the dual energy (6.5) using vector products

$$\tilde{\mathcal{E}}_{\mathcal{D}_1}(f, v) = \frac{1}{N} \sum_{\alpha} w_{\alpha}^{\top} \left\{ \sum_{i \in \alpha^*} \Delta_{\alpha i} \Delta_{\alpha i}^{\top} \right\} w_{\alpha} , \quad (6.7)$$

where

$$\Delta_{\alpha i} \triangleq \begin{pmatrix} 1 \\ p_i - x_{\alpha} \end{pmatrix}$$

is a 4×1 column vector for each point p_i in leaf α where x_{α} is the centroid of the leaf. Taking the partial derivative of (6.7) with respect to a particular w_{α} , we get

$$\frac{1}{2} \frac{\partial \tilde{\mathcal{E}}_{\mathcal{D}_1}}{\partial w_{\alpha}} = \left\{ \frac{1}{N} \sum_{i \in \alpha^*} \Delta_{\alpha i} \Delta_{\alpha i}^{\top} \right\} w_{\alpha} , \quad (6.8)$$

where the matrix in the curly brackets denotes the sufficient statistics related to the first data term. Similarly, we rewrite the second data term of the dual energy (6.5) using vector products

$$\tilde{\mathcal{E}}_{\mathcal{D}_2}(f, v) = \frac{\lambda_1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} (M_{\alpha} w_{\alpha} - n_i)^{\top} (M_{\alpha} w_{\alpha} - n_i), \quad (6.9)$$

where

$$M_{\alpha} \triangleq \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is a constant matrix to account for non-existence of f_{α} in the second data term. Taking the partial derivative of (6.9) with respect to a particular w_{α} , we get

$$\frac{1}{2} \frac{\partial \tilde{\mathcal{E}}_{\mathcal{D}_2}}{\partial w_{\alpha}} = \left\{ \frac{\lambda_1}{N} M_{\alpha}^{\top} M_{\alpha} \sum_{i \in \alpha^*} 1 \right\} w_{\alpha} - \frac{\lambda_1}{N} M_{\alpha}^{\top} \sum_{i \in \alpha^*} n_i, \quad (6.10)$$

where the matrix in the curly brackets denotes the sufficient statistics related to the second data term while the negative part (after negation) is also part of sufficient statistics which contributes to the right hand side b of the linear system.

Combining the factored statistics given in equations (6.8) and (6.10), we get the final form of the statistics for each occupied leaf α

$$(A_d)_{\alpha} = \frac{1}{N} \begin{bmatrix} S_{\alpha} & \Delta_{\alpha}^{\top} \\ \Delta_{\alpha} & \Sigma_{\alpha} \end{bmatrix} \quad \text{and} \quad (b)_{\alpha} = \frac{\lambda_1}{N} \begin{pmatrix} 0 \\ N_{\alpha} \end{pmatrix} \quad (6.11)$$

with

$$\begin{aligned} S_{\alpha} &\triangleq \sum_{i \in \alpha^*} 1 \\ \Delta_{\alpha} &\triangleq \sum_{i \in \alpha^*} (p_i - x_{\alpha}) \\ \Sigma_{\alpha} &\triangleq \lambda_1 S_{\alpha} \mathbf{I}_3 + \sum_{i \in \alpha^*} (p_i - x_{\alpha})(p_i - x_{\alpha})^{\top} \\ N_{\alpha} &\triangleq \sum_{i \in \alpha^*} n_i \end{aligned}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix.

6.4 Analysis of Sufficient Statistics

Only four types of statistics appear for a given leaf α , namely number of points S_α in the leaf, sum of point vectors with respect to leaf's centroid Δ_α , pseudo-covariance matrix of points Σ_α , and sum of normal vectors N_α . The 4×4 matrix $(A_d)_\alpha$ and 4×1 vector $(b)_\alpha$ serve as building blocks of matrix A_d and right hand side vector b . Since $(A_d)_\alpha$ is symmetric, it brings 10 independent coefficients. Together with the 3 non-zero coefficients that come from $(b)_\alpha$, one only need to store total 13 coefficients per octree leaf. An oriented point is represented with 6 coefficients (3 for point position, and 3 for its normal). Therefore, if there are more than 2 points per occupied octree leaf on average, it is better work with the statistics in terms of memory usage than the point cloud data directly. The benefit may not be obvious for small data sets containing a few million points. On the other hand, for large scenes containing hundreds of millions of points, octree resolution reaches to its maximum depth quickly while points are still consolidated into non-individual leaves containing more than 2 points. The other advantage of working with the statistics is in terms of time consumption. That is, each iteration of the solver is completed in one step, instead of traversing all points enclosed in a leaf because statistics are computed once every time the topology of octree is changed, *i.e.*, it is subdivided.

6.5 Implementation

We first describe an offline implementation of the proposed discretization which is similar to the offline approach used in the original implementation of SSD (Section 2.8 of Chapter 2). That is, the points are fully loaded into memory, and octree is constructed recursively in a top-down fashion as the function $f(x)$ is refined every time the octree depth d improves. The only difference is that the sufficient statistics are accumulated from scratch using the existing partitioning of points for a given octree depth d , and then $f(x)$ is refined from the values of the sufficient statistics, not from the points. Note that this is a proof-of-concept implementation to verify the correctness of the formulations proposed in this chapter, and whether the dual discretization produces surfaces with similar quality and behaviors as the original implementation of SSD. We refer the reader to sections 6.7 and 6.8 for a time and space efficient online implementation strategy. Independent of how statistics are accumulated (online or offline), the steps to refine $f(x)$ are the same for both schemes. In order to perform an iteration (k), we accumulate the $4K$ -dimensional result vector of

the product $A F^{(k)}$ and the $4K$ -dimensional right hand side vector b through following sums:

$$\begin{aligned}
A F^{(k)} &= \sum_{\alpha} B_{\alpha} F^{(k)} \\
&+ \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} C_{\alpha} \left\{ C_{\alpha}^{\top} F^{(k)} - C_{\beta}^{\top} F^{(k)} \right\} \sigma_{\alpha\beta} \\
&- \frac{\lambda_2}{W} \sum_{(\alpha, \beta)} C_{\beta} \left\{ C_{\alpha}^{\top} F^{(k)} - C_{\beta}^{\top} F^{(k)} \right\} \sigma_{\alpha\beta} \\
&+ \frac{\lambda_3}{W} \sum_{(\alpha, \beta)} D_{\alpha} \left\{ D_{\alpha}^{\top} F^{(k)} - E_{\beta}^{\top} F^{(k)} \right\} \sigma_{\alpha\beta} \\
&- \frac{\lambda_3}{W} \sum_{(\alpha, \beta)} E_{\beta} \left\{ D_{\alpha}^{\top} F^{(k)} - E_{\beta}^{\top} F^{(k)} \right\} \sigma_{\alpha\beta}
\end{aligned} \tag{6.12}$$

$$b = \sum_{\alpha} c_{\alpha}$$

where the first term is the contribution from the sufficient statistics, second and third terms are the contributions from the regularization term, and the last two terms are the contributions from the coupling term. B_{α} is a $4K \times 4K$ sparse matrix with a single 4×4 non-zero block of sufficient statistics $(A_d)_{\alpha}$ as defined in equation (6.11). The product $B_{\alpha} F^{(k)}$ simply updates the 4 values of $A F^{(k)}$ associated with leaf α with the result of $(A_d)_{\alpha} w_{\alpha}^{(k)}$. Similarly, c_{α} is a $4K \times 1$ sparse vector with 3 non-zero values from the sufficient statistics $(b)_{\alpha}$ as defined in equation (6.11), which simply updates the 4 values of b with $(b)_{\alpha}$. C_{α} is a $4K \times 3$ sparse matrix with a single non-zero block of identity I_3 for leaf α . Note that the term in the curly bracket associated with regularization term then simply reduces to

$$C_{\alpha}^{\top} F^{(k)} - C_{\beta}^{\top} F^{(k)} = v_{\alpha}^{(k)} - v_{\beta}^{(k)}.$$

D_{α} is a $4K \times 2$ sparse matrix with 5 non-zero entries. The first column consists of 4 values $[1 \ (x_{\beta} - x_{\alpha})^{\top}]^{\top}$, whereas the second column consists of $[1 \ 0 \ 0 \ 0]^{\top}$ at 4 indices associated with leaf α . The first column of E_{β} consists of $[1 \ 0 \ 0 \ 0]^{\top}$, whereas the second column consists of $[1 \ (x_{\alpha} - x_{\beta})^{\top}]^{\top}$ at 4 indices associated with leaf β . Note that the term in the curly bracket associated with coupling term then simply reduces to

$$D_{\alpha}^{\top} F^{(k)} - E_{\beta}^{\top} F^{(k)} = \begin{bmatrix} f_{\alpha}^{(k)} + v_{\alpha}^{(k)T} (x_{\beta} - x_{\alpha}) - f_{\beta}^{(k)} \\ f_{\alpha}^{(k)} - v_{\beta}^{(k)T} (x_{\alpha} - x_{\beta}) - f_{\beta}^{(k)} \end{bmatrix}.$$

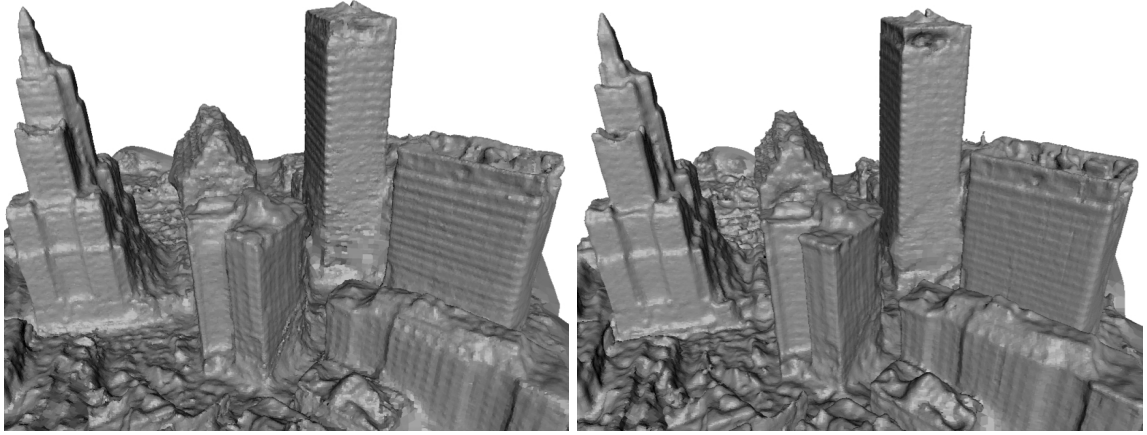


Figure 6.3: Surface reconstructions of “Downtown” site: using the original SSD-Primal (left), and the proposed SSD-Dual (right). Note similar quality with minor visual differences.

6.6 Results

We qualitatively compare surfaces reconstructed by the SSD method based on the original primal graph discretization scheme (SSD-Primal) and the SSD method based on the proposed dual graph discretization scheme (SSD-Dual). Note that both SSD-Primal and SSD-Dual methods solve the same variational problem with different discretization schemes. Therefore, significant visual differences are not expected. The SSD-Primal method that is used to generate results for this section follows the original implementation approach as described in Section 2.8, whereas the proposed SSD-Dual method follows the approach described in Section 6.5. Both implementations are offline in terms of handling input point cloud data, *i.e.*, input points are fully loaded into memory, and they stay in memory during surface estimation stage.

Figure 6.2 shows zoom-in visual comparisons of the left eye of Michelangelo’s David reconstructed at a maximal octree depth of 10 from head data that consists of 1 million accurate oriented point samples obtained from a laser scanner. Figure 6.3 shows visual similarities and differences of an urban scene, namely “Downtown” site, reconstructed from the input point cloud acquired by sampling voxel centroids of high occupancy density from the probabilistic voxel model learned by CPVM as described in Section 4.3 from aerial imagery of the scene captured by flying a helicopter around the city of Providence, RI. The resulting oriented point cloud is an example of real data consisting of

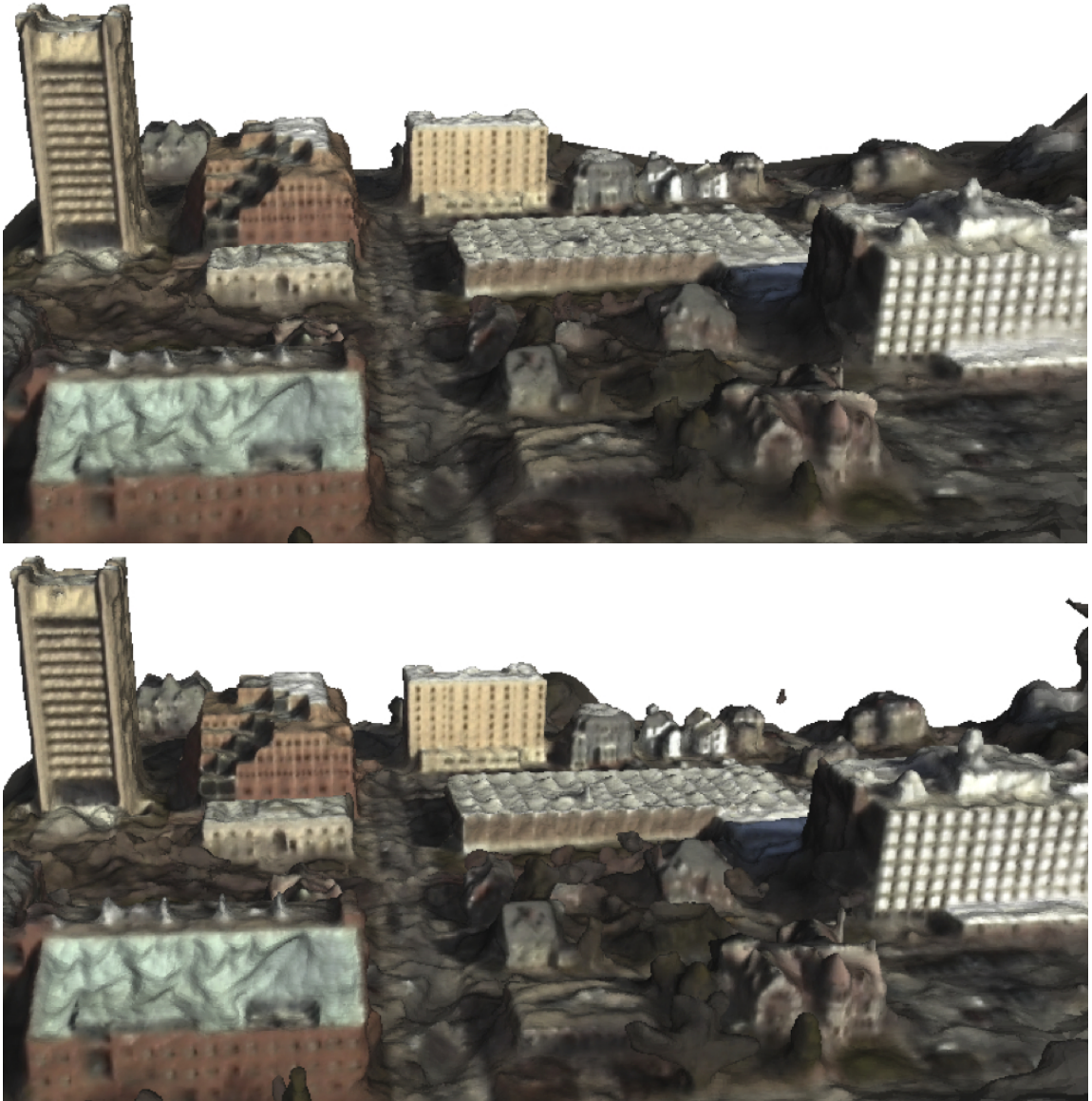


Figure 6.4: Surface reconstructions of “Brown University” site; using the original SSD-Primal (top), and the proposed SSD-Dual (bottom). Note similar quality with minor visual differences.

extremely noisy point samples and normals. Similarly, Figure 6.3 shows surface reconstruction results of “Brown University” site from the same dataset. All results show that the SSD method based on the proposed dual discretization (SSD-Dual) produces surfaces of visual quality either comparable to or better than the surfaces produced by the original SSD based on the primal discretization (SSD-Primal).

6.7 Online Implementation Strategy

In order to fully take advantage of sufficient statistics defined in this chapter, we propose to follow a bottom-up octree construction strategy instead of the top-down strategy used in SSD. A top-down subdivision strategy first loads all points in memory, places them in a bounding box (which serves as the root of the octree), and then starting from the root leaf, recursively subdivides the octree as a function of points using a test condition, *e.g.*, a leaf is subdivided if it contains more than 1 points. This procedure is repeated for every leaf until the test fails or maximum (or user desired) depth is reached. Top-down strategy is very useful when dealing with uneven sampling and sensor noise in point measurements because octree faithfully adapts to sampling density of the input cloud. In other words, it does not create excessive number of cells, and as a result, surface reconstruction algorithms deals with less number of variables to estimate, and becomes less prone to variance in data.

The disadvantage of the top-down strategy, on the other hand, is the requirement to load all points at once, and keep them all in memory until the octree construction is completed. Since a cascading multigrid solver was used in original implementation of SSD, this top-down behavior was desired because the signed distance was first estimated at a coarser level than desired, and then interpolated to next deeper level, and then refined further by the solver. However, the disadvantage makes the use of top-down approach prohibitive quickly for very large scenes consisting of hundreds of millions of points. Another disadvantage is that top-down approach does not gracefully allow the statistics to be integrated into the algorithm because statistics are typically sums over some information of points, *i.e.*, data is averaged out. If points are not in memory, there is no elegant way of quantizing statistics from parent leaf to its children; however, reverse is always possible. Section 6.8.1 describes an efficient routine to calculate the statistics of a parent only from the statistics of its children via simple linear algebra.

A bottom-up strategy loads points in an online fashion - one point or a small batch of points at a time, and places every point to the finest leaf possible determined by the maximum octree depth, assuming that the bounding box of the point cloud data is known (If that information is not available in the header of the input file, one can make an initial online pass over the file to determine the bounding box.). After all points are read, this procedure is followed by generation of missing siblings in the octree, and (optionally) another procedure to prune excessively generated cells by

merging children to adapt the octree resolution to the input point cloud density. The advantage of this approach is that it allows for integration of sufficient statistics into the octree construction procedure gracefully without the need to store points in memory. In detail, we propose the following implementation approach:

1. Open a file stream from the input point cloud data to be reconstructed
2. Read the bounding box from the file header if available (if not make a full pass and determine the bounding box)
3. Initialize an empty octree data structure
4. While the file stream is not empty:
Do
 - (a) Get a point from the stream, determine its location (at the deepest level) in the octree by quantization
 - (b) If there is node already, update the corresponding statistics using the information of the point. If not, insert a new node at the deepest level, and initialize statistics with the information of the point
 - (c) Discard the point
5. Complete the octree by creating new nodes to account for missing siblings
6. (Optional) Merge children by properly combining their sufficient statistics.

The advantage of the bottom-up approach is that the points are never fully loaded or stored in memory. An iterative linear solver uses the sufficient statistics to solve the sparse linear system of equations together with the topology of the octree. However, we can not use a cascading multigrid approach anymore - where solution was estimated on a coarser level, interpolated to the deeper level, and refined. Although not using the cascading multigrid approach increases the number of iterations for solver to converge, the computation (time) required for each iteration is much less since it is not required to traverse all points for a single iteration. Another advantage, and the main reason why we choose bottom-up approach in this chapter, is that its memory footprint is adapted to the density of the scene, not the number of points. As a result, large scenes can be reconstructed

with less memory footprint. The benefit comes with a price that number of iterations required is typically higher than the ones used in a top-down cascading multigrid approach.

In the next section, we introduce a hybrid approach (bottom-up construction, top-down solution) by taking advantage of the feature that the sufficient statistics of children can be combined efficiently to obtain the sufficient statistics of a parent.

6.8 Multigrid Method Using Only Sufficient Statistics

In the original implementation of the SSD, the octree data structure keeps only the leaf nodes during construction, *i.e.*, a parent is overwritten by its first child to save memory usage. However, if the dual formulation described in this chapter is implemented on an octree that already keeps the internal nodes in memory, and the octree has routines to inquire them; one can add an additional step:

7. Recursively combine sufficient statistics from leaves to their parents, and from the parents to their parents up to the root

to the end of the online implementation strategy described in Section 6.7. Thereby, all nodes (including internal nodes) of the octree have sufficient statistics. This minimal addition allows for the linear system to be solved from top-down using a cascading multigrid approach. In detail, an octree is first built in a bottom-up fashion as described in Section 6.7, then the sufficient statistics are recursively expanded from leaves to the internal nodes up to the root, and finally the problem is solved from top-down similar to the one used in SSD which is estimating the function at a coarser level than desired using the coarser sufficient statistics, interpolating it to the next level, and refining it by using the sufficient statistics of the next level, and repeating until the maximum or desired level is reached. The step to expand sufficient statistics from leaves to internal nodes does not rely on point cloud data, *i.e.*, a parent sets its sufficient statistics only using the values of the statistics of its children.

6.8.1 Expansion of Sufficient Statistics from Leaves to Internal Nodes

In order to get a formula for a parent's sufficient statistics from its children's statistics, we first expand the final form introduced in equation (6.11) to get the following canonical terms:

$$\begin{aligned}
\tilde{S}_\alpha &\triangleq \sum_{i \in \alpha^*} 1 = S_\alpha \\
\tilde{\Delta}_\alpha &\triangleq \sum_{i \in \alpha^*} p_i = \Delta_\alpha + S_\alpha x_\alpha \\
\tilde{\Sigma}_\alpha &\triangleq \sum_{i \in \alpha^*} p_i p_i^\top = \Sigma_\alpha - S_\alpha x_\alpha x_\alpha^\top - \lambda_1 S_\alpha \mathbf{I}_3 + \tilde{\Delta}_\alpha x_\alpha^\top + x_\alpha \tilde{\Delta}_\alpha^\top \\
\tilde{N}_\alpha &\triangleq \sum_{i \in \alpha^*} n_i = N_\alpha
\end{aligned} \tag{6.13}$$

which are intrinsically independent of the centroid x_α , and can be calculated only from the sufficient statistics of the leaf α . Let α_j be the j^{th} child, and the parent is denoted by $\pi(\alpha_1)$. Then, we can get the canonical terms of the parent $\pi(\alpha_1)$ by simply adding the canonical terms of its children:

$$\begin{aligned}
\tilde{S}_{\pi(\alpha_1)} &= \sum_{j=1}^8 \tilde{S}_{\alpha_j} \\
\tilde{\Delta}_{\pi(\alpha_1)} &= \sum_{j=1}^8 \tilde{\Delta}_{\alpha_j} \\
\tilde{\Sigma}_{\pi(\alpha_1)} &= \sum_{j=1}^8 \tilde{\Sigma}_{\alpha_j} \\
\tilde{N}_{\pi(\alpha_1)} &= \sum_{j=1}^8 \tilde{N}_{\alpha_j}
\end{aligned} \tag{6.14}$$

We finally undo the initial expansion step (6.13) to get the sufficient statistics of the parent $\pi(\alpha_1)$:

$$\begin{aligned}
S_{\pi(\alpha_1)} &= \tilde{S}_{\pi(\alpha_1)} \\
\Delta_{\pi(\alpha_1)} &= \tilde{\Delta}_{\pi(\alpha_1)} - \tilde{S}_{\pi(\alpha_1)} x_{\pi(\alpha_1)} \\
\Sigma_{\pi(\alpha_1)} &= \lambda_1 \tilde{S}_{\pi(\alpha_1)} \mathbf{I}_3 + \tilde{\Sigma}_{\pi(\alpha_1)} + \tilde{S}_{\pi(\alpha_1)} x_{\pi(\alpha_1)} x_{\pi(\alpha_1)}^\top - \tilde{\Delta}_{\pi(\alpha_1)} x_{\pi(\alpha_1)}^\top - x_{\pi(\alpha_1)} \tilde{\Delta}_{\pi(\alpha_1)}^\top \\
N_{\pi(\alpha_1)} &= \tilde{N}_{\pi(\alpha_1)}
\end{aligned} \tag{6.15}$$

6.8.2 Proportion of Internal Nodes Among All Octree Nodes

The critical question is: How big does the octree get if the internal nodes are kept in memory in addition to leaves? Note that our original goal in this chapter is to reduce the memory usage. Good news is that the increase is only 14.28% which can be proven through following tree relations:

$$T = L + I ,$$

$$T = E + 1 ,$$

$$E = nI ,$$

where T denotes the total number nodes, L denotes the number of leaf nodes, E denotes the number of edges, I denotes the number of internal nodes, and n is the number of children per internal node which is $n = 8$ for octrees. The relation between total number of nodes T , and the number of leaf nodes L is then:

$$\frac{T}{L} = \frac{8L - 1}{7L} \approx \frac{8}{7} \approx 1.1428 , \quad (6.16)$$

for large L . In other words, an octree consists of 87.5% leaf nodes, and 12.5% internal nodes. Octree traversal algorithms, in particular, finding neighbors of a given node, greatly benefit in terms of both implementation simplicity and speed from the existence of internal nodes - which come with such a small memory cost compared to leaf nodes. Note that SSD and variants described in this thesis heavily use octree neighbor search routines during the estimation of the signed distance function, and also during the isosurface extraction stage.

6.9 Further Reduction in Memory Footprint of Statistics

In this section, we discuss a modification to the first data term in dual energy formulation (6.5) so that it gives rise to an alternative statistics, namely *minimal* statistics, which comes with 60% less memory footprint than the sufficient statistics introduced in Section 6.3. The first data term of the dual discretization proposed in (6.5) approximates the value of f at a point p_i via the linear interpolation formula (6.3) which is a linear expansion from the value at the centroid f_α to approximate the value at the point $f(p_i)$. This expansion depends on the auxiliary vector field v which is another unknown.

Reversing this relation, one can expand the value $f(p_i)$ to approximate the value f_α

$$f_\alpha \approx f(p_i) + n_i^\top (x_\alpha - p_i) .$$

Moving the linear term to the other side, one can get

$$f(p_i) \approx f_\alpha - n_i^\top (x_\alpha - p_i) \quad (6.17)$$

which looks very similar to the original form (6.3). However, this form does not depend on unknown v_α , instead depends on normal data n_i . Plugging (6.17) back into the energy formulation (6.5), we get another approximation to the first data term:

$$\tilde{\mathcal{E}}_{\mathcal{D}_1}(f, \cdot) \approx \frac{1}{N} \sum_{\alpha} \sum_{i \in \alpha^*} (f_\alpha - f_i)^2 , \quad (6.18)$$

where $f_i \triangleq n_i^\top (x_\alpha - p_i) \in \mathbb{R}$. Taking the partial derivative with respect to f_α we simply get

$$\frac{1}{2} \frac{\partial \tilde{\mathcal{E}}_{\mathcal{D}_1}}{\partial f_\alpha} \approx \frac{1}{N} S_\alpha f_\alpha - \frac{1}{N} \sum_{i \in \alpha^*} f_i , \quad (6.19)$$

which depends on number of points S_α , and sum of their signed distances $F_\alpha \triangleq \sum_{i \in \alpha^*} f_i$. With this slight modification on the first data term, $(A_d)_\alpha$ becomes diagonal, and the coefficient F_α contributes to the right hand side $(b)_\alpha$. The statistics take the following *minimal* form:

$$(A_d)_\alpha = \frac{1}{N} \begin{bmatrix} S_\alpha & & & \\ & \lambda_1 S_\alpha & & \\ & & \lambda_1 S_\alpha & \\ & & & \lambda_1 S_\alpha \end{bmatrix} \quad \text{and} \quad (b)_\alpha = \frac{1}{N} \begin{pmatrix} F_\alpha \\ \lambda_1 N_\alpha \end{pmatrix} \quad (6.20)$$

In effect, one only needs to store total 5 coefficients per occupied octree leaf α for *minimal* statistics as opposed to 13 coefficients for *sufficient* statistics defined in equation (6.11).

While dealing with very large scenes, minimal statistics provide a major reduction ($\approx 61.5\%$) in memory usage. It is always advantageous compared to keeping oriented cloud data in memory since an oriented point requires 6 coefficients to be stored in memory. On the other hand, the

quality of the reconstruction result is expected to be inferior compared the one reconstructed from the sufficient statistics. One should choose one or the other depending on the type of application, desired quality, and available hardware resources.

6.10 Conclusion

In this chapter, we introduced an octree-based dual discretization for the continuous objective of SSD originally introduced in Chapter 2 to reconstruct very large scenes consisting of hundreds of millions of oriented points. The proposed dual discretization allowed the derivation of sufficient (and minimal) statistics that are adequate to solve the problem with less memory requirements than using point cloud data directly. We showed in results that an offline implementation of the proposed discretization (SSD-Dual) estimates surfaces with similar visual quality and behaviors (as expected) to the surfaces estimated by the original implementation (SSD-Primal) based on primal graph discretization. The proposed implementation does not require a primal graph or a dual graph built explicitly from the octree data structure, *i.e.*, an octree with a standard interface with available routines to traverse the octree leaves and the pairs of octree leaves that share a common face is only needed. In addition, we described an online implementation strategy in Section 6.7 to accumulate statistics from a large oriented point cloud without fully loading or storing it in memory, and an efficient cascading multigrid approach in Section 6.8 to solve the problem using only statistics as input. Actual implementation of algorithms introduced in those two sections are left for the future. Another important future work is to develop a memory-efficient isosurface extraction algorithm to handle very deep octrees (typically up to depth 21 due to 64-bit WORD size), and contour signed distance fields in a streaming fashion, *i.e.*, it should save polygons to disk as it contours the field without fully keeping the resulting polygon mesh in memory which requires that different parts of the polygon mesh are reloaded multiple times from the disk to ensure that the final mesh is watertight.

Chapter 7

Conclusions and Future Directions

This thesis presented a general and flexible global minimization framework to solve a set of related problems of reconstructing a watertight surface from a variety of 3D data representations such as oriented points, partially oriented points (including 2D cross-sectional curves), colored oriented points, and volumetric occupancy grids. We introduced new variational formulations that allow for a number of different discretizations, reduce to scalable least squares problems, and result in efficient space-adaptive (octree-based) algorithms that accept outputs of popular 3D shape acquisition methodologies (ranging from 3D laser scanners, RGB-D cameras, structured light systems to computational vision algorithms) as input, and were shown to generate detailed adaptive manifold polygon meshes of quality comparable with state-of-the-art algorithms.

Chapter 2 introduced the backbone of the global minimization framework which accepts an oriented point cloud as input, estimates an implicit function constrained to approximate the signed distance to the surface by a quadratic energy functional based on simple geometric cues, and outputs a polygon mesh surface of the underlying geometry by properly contouring the signed distance field. Dense oriented point clouds are typically obtained from laser scanners, or result from large scale simulations. Chapter 3 extended this framework to accept *colored* oriented point clouds, and to generate polygon meshes with pleasing color maps defined as color-per-vertex attributes. *Colored* oriented point are typically obtained by the multi-view stereo methods based on plane sweeping or dense feature matching over multiple 2D images. Chapter 4 extended the framework to accept volumetric occupancy grids where the underlying data is not a finite set of points anymore but a

volumetric likelihood of surface occupancy. This type of data are typically obtained by the multi-view stereo methods that are based on probabilistic volumetric modeling which learn the likelihood from video data, or can also be found in medical imaging. Chapter 5 extended the framework to point clouds where the accurate surface normals are not available potentially due to irregular sampling of the data, yet the optical measuring sensor still provides partial inside/outside information on the surface to be estimated. This type of data can be as general as any raw point cloud data such as those obtained from 3D laser scanners, RGB-D cameras, or inexpensive structured light systems. Since the method only relies on partially observed normals, it can also be used with sets of parallel/non-parallel 2D curves obtained by processing outputs of medical cross-sectional imaging techniques such as CT, MRI, PET, SPECT, and ultrasonography or simply for 3D modeling applications out of 2D drawings.

Chapter 6 introduced an octree-based dual graph discretization for the backbone method (Chapter 2) which was originally implemented using an octree-based primal graph discretization. The aim of proposing an alternative discretization is to ensure that the framework can gracefully reconstructs surfaces of very large scenes consisting of several hundreds of millions of oriented points. The dual discretization allowed the derivation of sufficient (and minimal) statistics that are adequate to solve the problem with less memory requirements than using point cloud data directly. An *online* implementation strategy to accumulate the statistics from a large oriented point cloud is also introduced. However, the actual implementation is left for the future. Since the method does not require either a primal graph or a dual graph built explicitly from the octree data structure, any standard octree data structure with routines available to traverse octree nodes and pairs of nodes that share a common face can be used to implement the method.

Future Directions

An important future direction is to explore non-smooth formulations that allow for surfaces with normal discontinuities (such as creases and corners) to be preserved, rather than approximated with smooth surfaces. In regression analysis, this is typically performed by replacing the squared norms of regularization term with ℓ_1 -norms. Estellers *et al.* [139] replaces ℓ_2 -norms of the formulation of Screened Poisson Reconstruction [65] method with Huber penalty (a smoothed version of ℓ_1 -norm) to avoid the shrinking bias of ℓ_2 -norm. Even though this modification significantly improves the

reconstruction of sharp features, smoothed versions of ℓ_1 -norm do not necessarily yield a *sparse* solution [87]. The regularization term of SSD and variants are based on finite difference of two gradient vectors in a small neighborhood. A *sparse* solution in this context means that the difference vector is *exactly* zero in most regions of the estimated signed distance function except with allowance for some large values (equivalently normal discontinuities) located at sharp features. There are several iterative methods available in optimization literature to minimize objectives that involve ℓ_1 -norms. In particular, ADMM [140] iterations can be easily incorporated into the conjugate gradient iterations of SSD reconstruction methods presented in this dissertation.

A variational formulation in general should combine all the following information of a scene: partial normals, missing normals, silhouettes, and depth discontinuities, and be accompanied by accurate isosurface extraction algorithms to achieve a detailed reconstruction. In particular, applying surface reconstruction methods to oriented points may yield a surface with extraneous surfaces in regions with missing data. On the other hand, it is often possible to estimate a hull volume that should contain the surface to be reconstructed. For example, if oriented points were obtained through a multi-view stereo algorithm from multiple calibrated 2D images, it is often possible to extract a collection of silhouettes from those images for an object of interest. Given a collection of calibrated silhouettes, one can also obtain a visual hull [141] volume. Alternatively, if oriented points were obtained by a range scanner, it is often possible to access raw range scans (depth maps). Given a collection of calibrated depth maps, one can obtain a depth hull [142] volume. These volumes can be exploited in order to get a more accurate surface reconstruction for the object of interest by adding an additional term similar to the proposed in [85] to the energy functionals of SSD reconstruction methods. The aim of this term is to encourage the reconstructed surface to stay in a specified hull volume. However, dissimilar to [85], the term should not use predetermined distance values to the surface since they are not known before the surface is estimated. Instead, the term should be designed to constrain that the signed distance field is *positive* outside the given hull which is the only known information in the beginning of estimation together with the input point cloud data. As a side note, if the implicit function is constrained to be an indicator function (with value zero outside the surface and value one inside the surface) such as the one estimated in Poisson surface reconstruction method [65], it is more straightforward to force the reconstructed surface stay in a given hull volume by hardly constraining the value of the function to be zero outside of the hull, *i.e.*, by incorporating the hull as a Dirichlet constraint into the global formulation [84].

Appendix

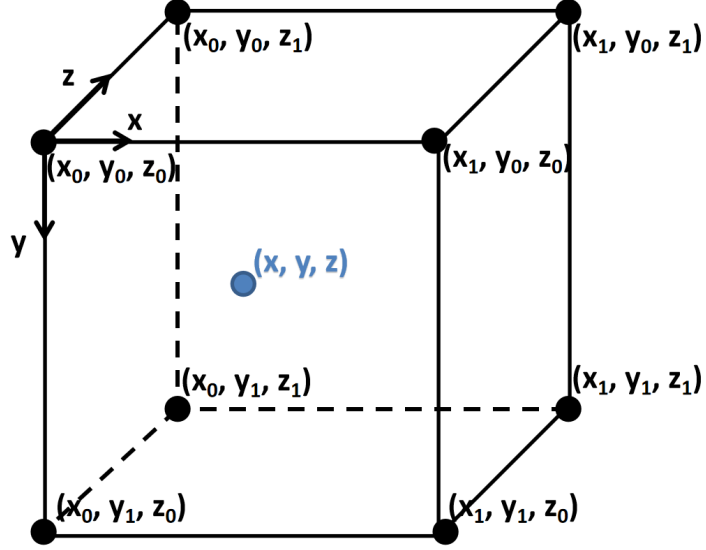


Figure A.1: A function evaluated at 8 vertices of a regular hexahedral grid. The value at a point (x, y, z) is approximated with trilinear interpolation formula (A.1), and its gradient with (A.3)

A.1 Analytical Gradient of Trilinear Interpolation Formula

As for the basis function $\Phi(\mathbf{x})$ in equation (2.6), the methods SSD (Chapter 2), SSD-C (Chapter 3), GSSD (Chapter 4), and SSD-Inc (Chapter 5) use coordinates (weights) of trilinear interpolation formula. This appendix section provides its explicit form and its analytical gradient.

If (x, y, z) belongs to a regular hexahedron as shown in Figure A.1, trilinear interpolation uses 8 vertex values $f_{ijk} = f(x_i, y_j, z_k)$ to approximate the value $f(x, y, z)$:

$$\begin{aligned}
 f(x, y, z) \approx I(t_x, t_y, t_z) = & (1 - t_x)(1 - t_y)(1 - t_z)f_{000} \\
 & + (1 - t_x)(1 - t_y)t_z f_{001} \\
 & + (1 - t_x)t_y(1 - t_z)f_{010} \\
 & + (1 - t_x)t_y t_z f_{011} \\
 & + t_x(1 - t_y)(1 - t_z)f_{100} \\
 & + t_x(1 - t_y)t_z f_{101} \\
 & + t_x t_y(1 - t_z)f_{110} \\
 & + t_x t_y t_z f_{111}
 \end{aligned} \tag{A.1}$$

where $t_x = \frac{x-x_0}{x_1-x_0}$, $t_y = \frac{y-y_0}{y_1-y_0}$, and $t_z = \frac{z-z_0}{z_1-z_0}$. Then, its gradient $\nabla f(x, y, z)$ takes the form:

$$\nabla f(x, y, z) \approx \begin{bmatrix} \frac{I_{t_x}(t_x, t_y, t_z)}{(x_1-x_0)} \\ \frac{I_{t_y}(t_x, t_y, t_z)}{(y_1-y_0)} \\ \frac{I_{t_z}(t_x, t_y, t_z)}{(z_1-z_0)} \end{bmatrix} \quad (\text{A.2})$$

with the partial derivatives

$$\begin{aligned} I_{t_x}(\cdot, t_y, t_z) = & - (1 - t_y) (1 - t_z) f_{000} \\ & - (1 - t_y) t_z f_{001} \\ & - t_y (1 - t_z) f_{010} \\ & - t_y t_z f_{011} \\ & + (1 - t_y) (1 - t_z) f_{100} \\ & + (1 - t_y) t_z f_{101} \\ & + t_y (1 - t_z) f_{110} \\ & + t_y t_z f_{111} \end{aligned} \quad (\text{A.3})$$

$$\begin{aligned} I_{t_y}(t_x, \cdot, t_z) = & - (1 - t_x) (1 - t_z) f_{000} \\ & - (1 - t_x) t_z f_{001} \\ & + (1 - t_x) (1 - t_z) f_{010} \\ & + (1 - t_x) t_z f_{011} \\ & - t_x (1 - t_z) f_{100} \\ & - t_x t_z f_{101} \\ & + t_x (1 - t_z) f_{110} \\ & + t_x t_z f_{111} \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned}
I_{t_z}(t_x, t_y, \cdot) = & - (1 - t_x) (1 - t_y) f_{000} \\
& + (1 - t_x) (1 - t_y) f_{001} \\
& - (1 - t_x) t_y f_{010} \\
& + (1 - t_x) t_y f_{011} \\
& - t_x (1 - t_y) f_{100} \\
& + t_x (1 - t_y) f_{101} \\
& - t_x t_y f_{110} \\
& + t_x t_y f_{111}
\end{aligned} \tag{A.5}$$

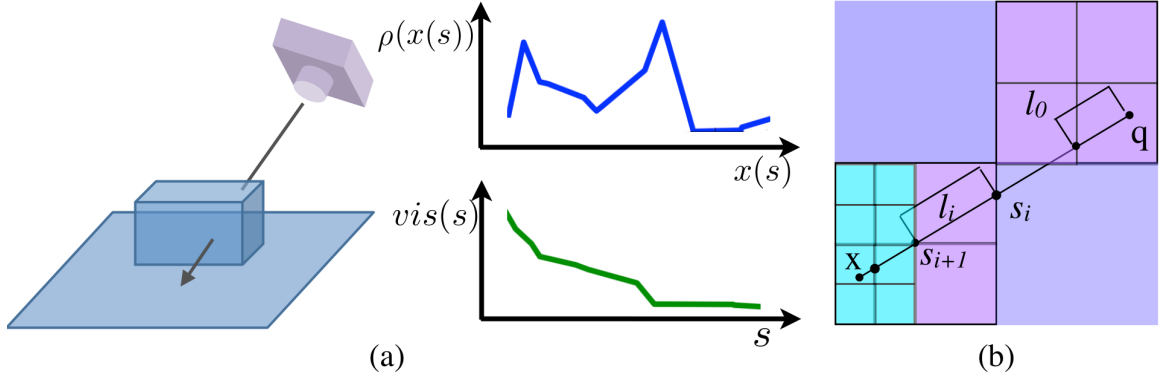


Figure A.2: (a) Plots of occlusion density ρ and vis as a ray travels through volume. ρ peaks (and visibility drops) as the ray pierces the two walls of the (almost) empty cube. (b) A parametrization of the ray into intervals according to octree cell intersections.

A.2 Continuous Probabilistic Volumetric Modeling (CPVM)

GSSD surface reconstruction method introduced in Chapter 4 takes volumetric surface densities $\mu(x)$ as input. The input densities shown in the chapter were obtained by running an implementation of Continuous Probabilistic Volumetric Modeling (CPVM) from aerial imagery. This appendix section provides a short overview of the method.

CPVM is a scalable volumetric approach to model uncertainties in geometry and appearance, as proposed by [81]. Crispell *et al.* propose a variable-resolution model based on a continuous density that removes the dependency on regular-sized grids of previous volumetric models [79, 123]. Although no constraints are imposed about the type of subdivision, an octree is used in practice to approximate the underlying continuous quantities as piecewise constant, achieving several orders of magnitude of storage savings.

Surface probabilities are closely related to a scalar function termed the *occlusion density* $\rho(x)$. The occlusion density at a point is a measure of the likelihood that the point occludes points behind it along any line of sight, assuming that the point itself is not occluded. Points along a ray (*e.g.*, the line of sight) may be parametrized by a distance s from q (*e.g.*, camera center) as $x(s) = q + sr$ $s \geq 0$. The visibility probability (A.6) of a point $x(s)$ is related to the integration of occlusion density from q to $q + sr$ (see [81] for derivation). Intuitively, the visibility along a ray will drop significantly when it hits a point of high occlusion, *i.e.*, a surface. An example of this relationship is shown in Fig A.2a.

$$vis(s) = e^{-\int_0^s \rho(t)dt} \quad (\text{A.6})$$

Learning the occlusion density from images follows an online Bayesian learning algorithm similar to that proposed by Pollard and Mundy [79]. In [79], the probability of a voxel being part of a surface, $P(X \in S)$, is updated with the intensity, I , observed in the pixel associated with a corresponding projection ray. Following Pollard's reasoning, for a discrete set of voxels, the surface update equation is expressed as:

$$P(X \in S|I_{N+1}) = P^N(X \in S) \frac{P^N(I_{N+1}|X \in S)}{P^N(I_{N+1})}, \quad (\text{A.7})$$

where the conditional $P^N(I_{N+1}|X \in S)$ and marginal $P^N(I_{N+1})$ are computed using the surface probabilities and appearance models stored in the voxels along the corresponding projection ray.

The definitions proposed by Crispell *et al.* allow for a generalization of (A.7). In [79], information along the projection ray is approximated by a series of voxels that intersect the ray, with no regard to the ray/voxel intersection geometry. However, in Crispell's model, the occlusion density is defined continuously and therefore, the model can account for the exact geometry of ray/voxel intersection, *i.e.*, the length of intersection segment. A ray is partitioned into a series of M intervals, where the i th interval is the result of the intersection of the ray and i th cell along the ray. The starting location and length of each interval are denoted by s_i and l_i respectively. Fig. A.2b depicts an illustration of this ray reasoning for the octree discretization.

The *segment length occlusion probability* is defined as the probability that a segment starting at s_i of length l_i is occluding and can be expressed as:

$$P(Q_{s_i}^{l_i}) = 1 - \frac{\text{vis}(s_i + l_i)}{\text{vis}(s_i)} = 1 - e^{-\rho_i l_i} \quad (\text{A.8})$$

Equation (A.7) can now be written in terms of $P(Q_{s_i}^{l_i})$, *i.e.*,

$$\begin{aligned} P(Q_{s_i}^{l_i}|I_{N+1}) &= P^N(Q_{s_i}^{l_i}) \frac{P^N(I_{N+1}|Q_{s_i}^{l_i})}{P^N(I_{N+1})} \\ &= P^N(Q_{s_i}^{l_i}) \frac{\text{pre}_i + \text{vis}(s_i)p_i(I_{N+1})}{\text{pre}_\infty + \text{vis}_\infty p_\infty(I_{N+1})} \end{aligned} \quad (\text{A.9})$$

$$\text{pre}_i \equiv \sum_{j=0}^{i-1} P^N(Q_{s_j}^{l_j}) \text{vis}(s_j) p_j(I_{N+1}) \quad (\text{A.10})$$

$$\text{vis}_\infty \equiv \prod_{i=0}^{M-1} [1 - P^N(Q_{s_i}^{l_i})] \quad (\text{A.11})$$

The term pre_i accounts for the probability of observing the given intensity I_{N+1} taking into account all segments between the camera center and segment $i - 1$. The term vis_∞ measures the probability of a ray passing unoccluded through the model; in such cases the observed appearance can be thought of as the appearance of “background”, which is modeled by the density p_∞ . The new update equations can be used to update $\rho(\mathbf{x})$ directly using (A.8).

Equation (A.9) has a simple interpretation, the occlusion density of a cell increases if the appearance model at the cell, explains the intensity observed in the image better than any other cell along the ray or by the background. The appearance at each cell is modeled with a Gaussian mixture distribution that is updated as in [79] using an online approach based on Stauffer and Grimson’s background modeling algorithm [143].

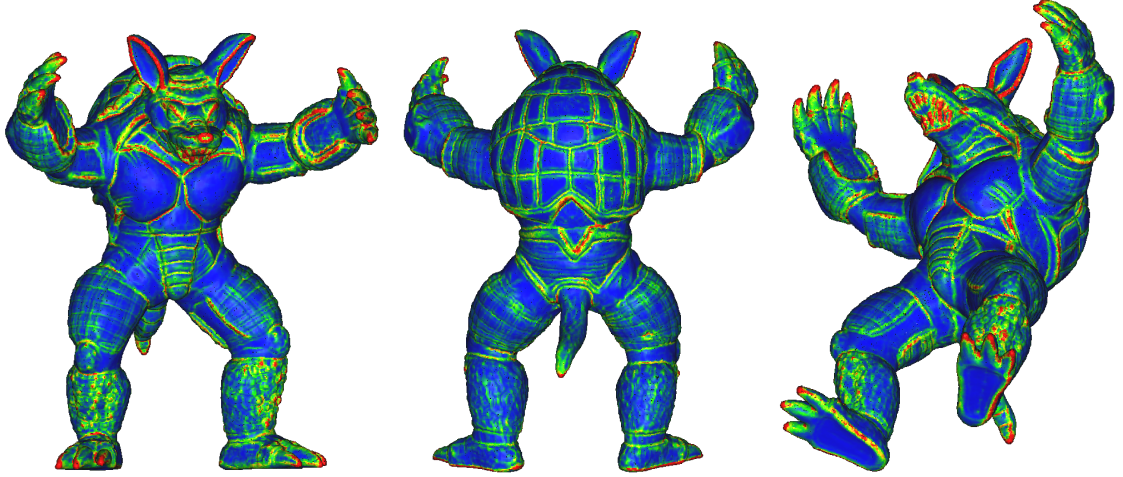


Figure A.3: Color-map visualization of a *weighted* oriented point cloud. The weights are calculated using the proposed scheme (A.14). Color blue shows areas with the lowest weight, whereas the red shows areas with the highest weight. The green shows areas with values in between.

A.3 A Point Weighting Scheme to Reveal Sharp Features

In Chapter 4, we showed that the original SSD formulation (2.4) can be regarded as a special case of the continuous formulation (4.1) if each point x_i is assigned with a uniform weight $\mu_i = \frac{1}{N}$. The formulation also allows for assigning non-uniform weights to points $\mu_i = \mu(x_i)$ which can be used to enhance trusted points or sharp features (such as edges and corners) in *weighted* least squares sense:

$$\mathcal{E}(f) = \frac{1}{M} \sum_{i=1}^N f(x_i)^2 \mu_i + \frac{\lambda_1}{M} \sum_{i=1}^N \|\nabla f(x_i) - n_i\|^2 \mu_i + \frac{\lambda_2}{|V|} \int_V \|Hf(x)\|^2 dx, \quad (\text{A.12})$$

with $M = \sum_i \mu_i$.

This appendix section presents a weighting scheme for a given input point cloud to reveal its high curvature points based on analysis of predominant directions of normals in the local neighborhood of each point in the cloud. Figure A.3 shows color-map rendering of a *weighted* oriented point cloud for which the weights are calculated using on the proposed scheme (A.14). Color blue shows areas with the lowest weight, and color red shows areas with the highest weight. Note the strong red color at pointy areas of the model such as the finger tips or the tip of the tail, whereas the strong blue dominates locally planar areas such as the torso or the soles of the feet.

We first find K nearest neighbors $\mathcal{D}_i = \{(p_{i_1}, n_{i_1}), \dots, (p_{i_K}, n_{i_K})\}$ of each point p_i in the point set. This step can be performed efficiently with a kd-tree data structure [133]. We then construct

the second moment matrix:

$$S_i = \frac{\sum_k \{n_{i_k} n_{i_k}^T\} w_{i_k}}{\sum_k w_{i_k}} \quad (\text{A.13})$$

with weights w_{i_k} inversely proportional to the distance between the point p_i and its k^{th} neighbor p_{i_k} , *e.g.*,

$$w_{i_k} \triangleq \frac{1}{\|p_i - p_{i_k}\|^2}.$$

The second moment matrix is known as the structure tensor in Computer Vision. The 2×2 structure tensor summarizes first order derivatives around a given pixel in an image. The analysis of its eigenvalues gives valuable information for corner or edge detection algorithms [144]. If there is one large eigenvalue, the pixel is classified as edge; whereas if there are two large eigenvalues, the pixel is classified as corner. With regard to point clouds, we observe that the eigenvalues of S_i given in equation (A.13) sum up to 1, *i.e.*,

$$e_{i_1} + e_{i_2} + e_{i_3} = 1$$

by construction since S_i is accumulated by the weighted average of outer products of unit normals. As a result, if all three eigenvalues are large, they take similar values $e_{i_j} \approx 1/3$. We are looking to design a measure on the set of eigenvalues of S_i which will serve as a weight μ_i for the point p_i . The measure should give a low value if the point is located near a planar region, *i.e.*, distribution of eigenvalues are “peaky”; and high value if the point comes from an interesting non-planar region, *i.e.*, distribution of eigenvalues are uniform. Inspired by the Rényi entropy measure [145] of order 2, we propose the following:

$$\mu_i = -\log\left(\sum_{j=1}^3 e_{i_j}^2\right) = -\log(\text{Tr}(S_i^2)) \quad (\text{A.14})$$

with $\text{Tr}(S_i^2)$ denotes the trace of the matrix square S_i^2 . This formula does not involve eigenvalue decomposition; hence, it is both computationally stable and efficient to calculate. Note that the outer product in equation (A.13) cancels out the orientation information of normals. As a result, the same weighting scheme can also be used to assign weights to *unoriented* point sets which improves visualization of features in those sets; however, orientation information is still needed for surface reconstruction by most variational methods as discussed in several parts of this dissertation.

A.4 Registration of Points to Signed Distance Function

The Iterative Closest Point (ICP) algorithm is widely used for alignment of 3D shapes or scans as mentioned in Section 1.3.2. If the corresponding points between the two shapes are known a-priori, the transformation can be computed in closed-form [44] based on the singular value decomposition. However, in general, the correspondences are not known between a new scan and the previously captured scans. The ICP addresses the correspondence problem iteratively by performing three main steps: point matching, objective minimization, and transformation of points. The matching step associates every point in the first shape with their nearest neighbor in the second shape which involves time consuming nearest neighbor searches. The search routine is typically implemented efficiently with a kd-tree data structure [133]. However, it can be completely avoided if the previously captured scans are represented by an implicit form such as smooth signed distance field estimated using one of the methods presented in this dissertation. This appendix section describes the steps involved for the minimization of a “point to signed distance field” objective with the aim to estimate the parameters of the rigid body transformation, *i.e.*, translation and rotation, iteratively.

Given a set of N points $\{p_1, \dots, p_N\}$ obtained from a new scan, the rotation matrix R and the translation vector t that transform the scan to the signed distance field f can be estimated by minimizing the following objective

$$\mathcal{E}(R, t) = \frac{1}{N} \sum_{i=1}^N f(Rp_i + t)^2. \quad (\text{A.15})$$

Note the objective (A.15) intrinsically has 6 degrees of freedom, 3 from the translation vector $t \in \mathbb{R}^3$, and 3 from the rotation matrix R (since $R \in \mathbb{SO}(3)$ - the 3D rotation group) which can be parametrized by exponential coordinates $v \in \mathbb{R}^3$ using the Euler-Rodrigues formula

$$R(v) = I_3 + \sin(\theta)\hat{U} + (1 - \cos(\theta))\hat{U}^2 \quad (\text{A.16})$$

where I_3 is the 3×3 identity matrix, $\theta = \|v\|$ and $\hat{U} = \left[\frac{v}{\|v\|} \right]_{\times}$ with

$$[u]_{\times} \triangleq \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \quad (\text{A.17})$$

is the cross product (skew-symmetric) matrix. Then the objective (A.15) takes the form

$$\mathcal{E}(v, t) = \frac{1}{N} \sum_{i=1}^N f(\mathbf{R}(v)p_i + t)^2 \quad (\text{A.18})$$

which can be linearized around a particular $(v^{(k)}, t^{(k)})$

$$\mathcal{E}(v^{(k)} + \delta v, t^{(k)} + \delta t) \approx \frac{1}{N} \sum_{i=1}^N \left(r_i(v^{(k)}, t^{(k)}) + \delta v^\top \frac{\partial r_i(v^{(k)}, t^{(k)})}{\partial v} + \delta t^\top \frac{\partial r_i(v^{(k)}, t^{(k)})}{\partial t} \right)^2 \quad (\text{A.19})$$

where the residuals and their partial derivatives are given by

$$\begin{aligned} r_i(v, t) &\triangleq f(\mathbf{R}(v)p_i + t) \\ \frac{\partial r_i(v, t)}{\partial v} &\triangleq \left[\frac{\partial \mathbf{R}(v)p_i}{\partial v} \right]^\top \nabla f(\mathbf{R}(v)p_i + t) \\ \frac{\partial r_i(v, t)}{\partial t} &\triangleq \nabla f(\mathbf{R}(v)p_i + t) \end{aligned} \quad (\text{A.20})$$

with the compact formula [146]

$$\left[\frac{\partial \mathbf{R}(v)x}{\partial v} \right] = -\mathbf{R}(v)[x]_\times \frac{v v^\top + (\mathbf{R}(v)^\top - \mathbf{I}_3)[v]_\times}{v^\top v} \quad (\text{A.21})$$

for the derivative of $\mathbf{R}(v)x$ with respect to the exponential coordinates v , where x is independent of v . Minimizing the linearized objective (A.19) with respect to $(\delta v, \delta t)$ reduces to solving the linear system

$$\underbrace{\frac{1}{N} \begin{bmatrix} \sum_{i=1}^N \frac{\partial r_i^{(k)}}{\partial v} \frac{\partial r_i^{(k)}}{\partial v}^\top & \sum_{i=1}^N \frac{\partial r_i^{(k)}}{\partial v} \frac{\partial r_i^{(k)}}{\partial t}^\top \\ \sum_{i=1}^N \frac{\partial r_i^{(k)}}{\partial t} \frac{\partial r_i^{(k)}}{\partial v}^\top & \sum_{i=1}^N \frac{\partial r_i^{(k)}}{\partial t} \frac{\partial r_i^{(k)}}{\partial t}^\top \end{bmatrix}}_A \underbrace{\begin{bmatrix} \delta v \\ \delta t \end{bmatrix}}_\delta = - \underbrace{\frac{1}{N} \begin{bmatrix} \sum_{i=1}^N r_i^{(k)} \frac{\partial r_i^{(k)}}{\partial v} \\ \sum_{i=1}^N r_i^{(k)} \frac{\partial r_i^{(k)}}{\partial t} \end{bmatrix}}_b \quad (\text{A.22})$$

where $r_i^{(k)} = r_i(v^{(k)}, t^{(k)})$, $\frac{\partial r_i^{(k)}}{\partial v} = \frac{\partial r_i(v^{(k)}, t^{(k)})}{\partial v}$, and $\frac{\partial r_i^{(k)}}{\partial t} = \frac{\partial r_i(v^{(k)}, t^{(k)})}{\partial t}$. Finally, the parameters are iteratively updated with

$$\begin{aligned} v^{(k+1)} &\leftarrow v^{(k)} + \delta v, \\ t^{(k+1)} &\leftarrow t^{(k)} + \delta t. \end{aligned} \quad (\text{A.23})$$

for $k = 0, 1, 2, \dots$ starting from $t^{(0)} = [0 \ 0 \ 0]^\top$ and $v^{(0)} = [0 \ 0 \ 0]^\top$; in which case, the formulas (A.16) and (A.21) take the form $\mathbf{R}(v^{(0)}) = \mathbf{I}_3$ and $\left[\frac{\partial \mathbf{R}(v^{(0)})x}{\partial v} \right] = -[x]_\times$, respectively.

The above are the typical steps for the minimization of any non-linear least squares problems

involving 3D rotations (and translations). If the linear system (A.22) is solved with a damping parameter $\lambda > 0$, *i.e.*, $(A + \lambda I_6)\delta = b$, we reach to the well-known Levenberg-Marquardt algorithm. In the presented algorithm, no nearest neighbor search is involved. However, each iteration of the algorithm requires function and gradient evaluations (A.20). Since the signed distance field is piecewise linear or trilinear over an octree grid, the location of a transformed point in the octree must be found for these evaluations, which reduces to simple quantization and hash-map lookup. The algorithm converges quite rapidly if the new scan is close to isosurface zero of the signed distance field and there is enough partial overlap between the scan and the signed distance field. In practice, there may not be enough overlap, then the objective function (A.15) should be modified both to ignore non-overlapping data points and to maximize overlapping data points with previously scanned regions. This can be performed with the use of an auxiliary surface density $\mu(x) : V \rightarrow \mathbb{R}^+$ by minimizing the following objective:

$$\tilde{\mathcal{E}}(\mathbf{R}, t) = \frac{\sum_{i=1}^N f(\mathbf{R}p_i + t)^2 \mu(\mathbf{R}p_i + t)}{\sum_{i=1}^N \mu(\mathbf{R}p_i + t)} . \quad (\text{A.24})$$

Surface density μ should be maintained separately from the signed distance field f . Octree construction process resembles this information since it is adapted to the point density during estimation of f . If x belongs to the octree leaf α , then $\mu(x) \approx \frac{S_\alpha}{V_\alpha}$; where S_α is the number of points enclosed in octree leaf α , and V_α is the volume of octree leaf α . However, the density $\mu(x)$ estimated this way takes zero values in most regions of the volume V . An estimator better fit to the problem can be obtained by using a (differentiable) kernel density estimator [87]. Another complication is that the objective (A.24) is no longer represented as a sum of squares of some residuals. A gradient descent routine can be implemented using the partial derivatives of the objective (A.24) with respect to v and t .

Bibliography

- [1] F. Calakli and G. Taubin. “SSD: Smooth Signed Distance Surface Reconstruction”. *Computer Graphics Forum* **30** 7 (2011). <http://mesh.brown.edu/ssd> (cit. on pp. [viii](#), [8](#), [9](#), [11](#), [12](#), [16](#), [68](#), [69](#)).
- [2] F. Calakli and G. Taubin. “SSD-C: Smooth Signed Distance Colored Surface Reconstruction”. In: *Expanding the Frontiers of Visual Analytics and Visualization*. London, UK: Springer London, 2012, pp. 323–338 (cit. on pp. [viii](#), [11](#)).
- [3] F. Calakli, A. O. Ulusoy, M. I. Restrepo, G. Taubin, and J. L. Mundy. “High Resolution Surface Reconstruction from Multi-view Aerial Imagery”. In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*. 2012, pp. 25–32 (cit. on pp. [viii](#), [11](#)).
- [4] F. Calakli and G. Taubin. *Method to reconstruct a surface from partially oriented 3-d points*. US Patent App. 16/402,022. 2019 (cit. on p. [viii](#)).
- [5] C. Dick, R. Burgkart, and R. Westermann. “Distance Visualization for Interactive 3D Implant Planning”. *IEEE Transactions on Visualization and Computer Graphics* **17** 12 (2011), pp. 2173–2182 (cit. on pp. [2](#), [3](#)).
- [6] S. D. Roth. “Ray casting for modeling solids”. *Computer Graphics and Image Processing* **18** 2 (1982), pp. 109–144 (cit. on p. [2](#)).
- [7] A. S. Glassner, ed. *An Introduction to Ray Tracing*. GBR: Academic Press Ltd., 1989 (cit. on p. [2](#)).

- [8] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. “Interactive Ray Tracing for Isosurface Rendering”. In: *Proceedings of the Conference on Visualization '98. VIS '98*. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, pp. 233–238 (cit. on p. 2).
- [9] W. E. Lorensen and H. E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques. SIGGRAPH '87*. 1987, pp. 163–169 (cit. on pp. 2, 9, 21, 28, 33).
- [10] T. Ju, F. Losasso, S. Schaefer, and J. Warren. “Dual Contouring of Hermite Data”. *ACM Trans. Graph.* **21** 3 (2002), pp. 339–346 (cit. on pp. 2, 10).
- [11] S. Schaefer and J. Warren. “Dual Marching Cubes: Primal Contouring of Dual Grids”. *Computer Graphics Forum* **24** 2 (2005), pp. 195–201 (cit. on pp. 2, 10, 33, 47, 61).
- [12] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe. “Unconstrained isosurface extraction on arbitrary octrees”. In: *Proceedings of the fifth Eurographics symposium on Geometry processing*. Eurographics Association. 2007, pp. 125–133 (cit. on pp. 2, 9).
- [13] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. “The Digital Michelangelo Project: 3D Scanning of Large Statues”. In: *SIGGRAPH '00*. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 131–144 (cit. on pp. 2, 5, 7).
- [14] F. Bernardini, H. Rushmeier, I. M. Martin, J. Mittleman, and G. Taubin. “Building a digital model of Michelangelo’s Florentine Pieta”. *IEEE Computer Graphics and Applications* **22** 1 (2002), pp. 59–67 (cit. on pp. 2, 5).
- [15] V. Isler, B. Wilson, and R. Bajcsy. “Building a 3D Virtual Museum of Native American Baskets”. In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. 2006, pp. 954–961 (cit. on p. 2).
- [16] Y. Qi, S. Cai, and S. Yang. “3D Modeling, Codec and Protection in Digital Museum”. In: *Second Workshop on Digital Media and its Application in Museum Heritages (DMAMH 2007)*. 2007, pp. 231–236 (cit. on p. 2).

- [17] E. Gay, D. Cooper, B. Kimia, G. Taubin, D. Cabrini, S. Karumuri, W. Doutre, S. Liu, K. Galor, D. Sanders, and A. Willis. “REVEAL intermediate report”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. 2010, pp. 1–6 (cit. on p. 2).
- [18] L. Dai. “The 3D Digital Technology of Fashion Design”. In: *2011 International Symposium on Computer Science and Society*. 2011, pp. 178–180 (cit. on p. 2).
- [19] F. Cordier, Hyewon Seo, and N. Magnenat-Thalmann. “Made-to-measure technologies for an online clothing store”. *IEEE Computer Graphics and Applications* **23** 1 (2003), pp. 38–48 (cit. on p. 2).
- [20] J. Rodriguez-Larena, F. Canal, and F. Campo. “An optical 3D digitizer for industrial quality control applications”. In: *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA '99 (Cat. No.99TH8467)*. Vol. 2. 1999, 1155–1160 vol.2 (cit. on p. 3).
- [21] G. Medioni, J. Choi, C. Kuo, A. Choudhury, Li Zhang, and D. Fidaleo. “Non-Cooperative Persons Identification at a Distance with 3D Face Modeling”. In: *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*. 2007, pp. 1–6 (cit. on p. 3).
- [22] N. Wongwaen and C. Sinthanayothin. “Computerized algorithm for 3D teeth segmentation”. In: *2010 International Conference on Electronics and Information Engineering*. Vol. 1. 2010, pp. V1-277-V1-280 (cit. on p. 3).
- [23] H. Xu, Y. Hu, Yuqing Chen, Z. Ma, and Defeng Wu. “A novel 3D surface modeling based on Spatial Neighbor Points Coupling in reverse engineering”. In: *2010 International Conference On Computer Design and Applications*. Vol. 5. 2010, pp. V5-59-V5-62 (cit. on p. 3).
- [24] Y. Lin, M. Song, D. T. P. Quynh, Y. He, and C. Chen. “Sparse Coding for Flexible, Robust 3D Facial-Expression Synthesis”. *IEEE Computer Graphics and Applications* **32** 2 (2012), pp. 76–88 (cit. on p. 3).
- [25] S. Chaudhuri, D. Ritchie, J. Wu, K. Xu, and H. Zhang. “Learning Generative Models of 3D Structures”. *Computer Graphics Forum* **39** 2 (2020), pp. 643–666. <https://learn3dgen.github.io> (cit. on p. 3).

- [26] D. Lanman and G. Taubin. “Build Your Own 3D Scanner: 3D Photography for Beginners”. In: *ACM SIGGRAPH 2009 Courses*. SIGGRAPH ’09. New Orleans, Louisiana: Association for Computing Machinery, 2009 (cit. on p. 3).
- [27] D. Moreno, K. Son, and G. Taubin. “Embedded phase shifting: Robust phase shifting with embedded signals”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 2301–2309 (cit. on p. 3).
- [28] C. Wu. “Towards Linear-Time Incremental Structure from Motion”. In: *2013 International Conference on 3D Vision - 3DV 2013*. 2013, pp. 127–134. <http://homes.cs.washington.edu/~ccwu/vsfm> (cit. on pp. 3, 4).
- [29] S. Fuhrmann, F. Langguth, and M. Goesele. “MVE - A Multi-View Reconstruction Environment”. In: *Eurographics Workshop on Graphics and Cultural Heritage*. Ed. by R. Klein and P. Santos. The Eurographics Association, 2014. <https://www.gcc.tu-darmstadt.de/home/proj/mve> (cit. on pp. 3, 7).
- [30] S. Giancola, M. Valenti, and R. Sala. *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*. SpringerBriefs in Computer Science. Springer International Publishing, 2018 (cit. on pp. 4, 5).
- [31] J. Ackermann and M. Goesele. “A Survey of Photometric Stereo Techniques”. *Found. Trends. Comput. Graph. Vis.* **9** 34 (2015), pp. 149–254 (cit. on pp. 4, 68).
- [32] Y. Zhao and G. Taubin. “Real-time stereo on GPGPU using progressive multi-resolution adaptive windows”. *Image and Vision Computing* **29** 6 (2011), pp. 420–432 (cit. on p. 4).
- [33] A. O. Ulusoy, F. Calakli, and G. Taubin. “One-shot Scanning using De-Bruijn Spaced Grids”. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE. 2009, pp. 1786–1792 (cit. on p. 4).
- [34] A. O. Ulusoy, F. Calakli, and G. Taubin. “Robust One-shot 3D Scanning using Loopy Belief Propagation”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE. 2010, pp. 15–22 (cit. on p. 4).
- [35] J. Garcia and Z. Zalevsky. *Range mapping using speckle decorrelation*. US Patent 7,433,024. 2008 (cit. on p. 4).

- [36] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. *Depth mapping using projected patterns*. US Patent 8,150,142. 2012 (cit. on p. 4).
- [37] N. Snavely, S. Seitz, and R. Szeliski. “Photo tourism: exploring photo collections in 3D”. In: *ACM Transactions on Graphics*. Vol. 25. 3. 2006, pp. 835–846. <https://www.cs.cornell.edu/~snavely/bundler> (cit. on pp. 4, 5, 48, 63).
- [38] Y. Furukawa and J. Ponce. “Accurate, Dense, and Robust Multi-View Stereopsis”. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **32** 8 (2010), pp. 1362–1376. <https://www.di.ens.fr/pmvs> (cit. on pp. 4, 5, 48, 54, 62, 66).
- [39] D. G. Lowe. “Distinctive image features from scale-invariant keypoints”. *International Journal of Computer Vision* **60** 2 (2004), pp. 91–110 (cit. on p. 4).
- [40] F. Bellochio, N. Borghese, F. S., and P. V. *3D Surface Reconstruction: multi-scale hierarchical approaches*. Springer Science & Business Media. New York, NY, USA: Springer New York Inc., 2013 (cit. on p. 5).
- [41] B. J. Brown and S. Rusinkiewicz. “Global Non-Rigid Alignment of 3-D Scans”. *ACM Trans. Graph.* **26** 3 (2007), 21–es (cit. on p. 6).
- [42] P. J. Besl and N. D. McKay. “A method for registration of 3-D shapes”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14** 2 (1992), pp. 239–256 (cit. on p. 6).
- [43] Y. Dez, F. Roure, X. Lladó, and J. Salvi. “A Qualitative Review on 3D Coarse Registration Methods”. *ACM Comput. Surv.* **47** 3 (2015) (cit. on p. 6).
- [44] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9** 5 (1987), pp. 698–700 (cit. on pp. 6, 109).
- [45] S. Rusinkiewicz and M. Levoy. “Efficient variants of the ICP algorithm”. In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152 (cit. on p. 6).
- [46] Y. Chen and G. Medioni. “Object Modelling by Registration of Multiple Range Images”. *Image Vision Comput.* **10** 3 (1992), pp. 145–155 (cit. on p. 6).

- [47] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. “Registration of Point Cloud Data from a Geometric Optimization Perspective”. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. SGP ’04. Nice, France: Association for Computing Machinery, 2004, pp. 22–31 (cit. on p. 6).
- [48] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. “Robust Articulated-ICP for Real-Time Hand Tracking”. *Computer Graphics Forum* **34** 5 (2015), pp. 101–114 (cit. on p. 6).
- [49] H. Q. Luong, M. Vlamincx, W. Goeman, and W. Philips. “Consistent ICP for the registration of sparse and inhomogeneous point clouds”. In: *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. 2016, pp. 262–267 (cit. on p. 6).
- [50] S. Rusinkiewicz. “A Symmetric Objective Function for ICP”. *ACM Trans. Graph.* **38** 4 (2019). <https://gfx.cs.princeton.edu/proj/trimesh2> (cit. on p. 6).
- [51] F. Pomerleau, F. Colas, and R. Siegwart. “A Review of Point Cloud Registration Algorithms for Mobile Robotics”. *Found. Trends Robot* **4** 1 (2015), pp. 1–104 (cit. on p. 6).
- [52] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “G2o: A general framework for graph optimization.” In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3607–3613 (cit. on p. 6).
- [53] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva. “A survey of surface reconstruction from point clouds”. In: *Computer Graphics Forum*. Vol. 36. 1. Wiley Online Library. 2017, pp. 301–329 (cit. on pp. 7, 8, 20, 55, 68, 69).
- [54] B. Ummerhofer and T. Brox. “Global, Dense Multiscale Reconstruction for a Billion Points”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1341–1349. <https://lmb.informatik.uni-freiburg.de/people/ummenhof/multiscalefusion> (cit. on pp. 7–9, 11, 12, 17, 68, 69).
- [55] N. Amenta, M. Bern, and M. Kamvysselis. “A New Voronoi-Based Surface Reconstruction Algorithm”. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 415–421 (cit. on p. 8).

- [56] 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 (1999), pp. 349–359 (cit. on pp. 8, 20).
- [57] N. Amenta, S. Choi, and R. K. Kolluri. “The Power Crust”. In: SMA ’01. Ann Arbor, Michigan, USA: Association for Computing Machinery, 2001, pp. 249–266 (cit. on p. 8).
- [58] N. Amenta, K. Tamal, and N. Leekha. “A Simple Algorithm for Homeomorphic Surface Reconstruction”. *International Journal of Computational Geometry & Applications* **12** 1&2 (2002), pp. 125–141 (cit. on p. 8).
- [59] T. K. Dey and S. Goswami. “Provable Surface Reconstruction from Noisy Samples”. In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG ’04. Brooklyn, New York, USA: Association for Computing Machinery, 2004, pp. 330–339 (cit. on p. 8).
- [60] H. Edelsbrunner. “Surface reconstruction by wrapping finite sets in space”. In: *Discrete and computational geometry*. Springer, 2003, pp. 379–404 (cit. on p. 8).
- [61] J. Giesen and M. John. “The flow complex: a data structure for geometric modeling”. *Computational Geometry* **39** 3 (2008), pp. 178–190 (cit. on p. 8).
- [62] T. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Univ Pr, 2007 (cit. on pp. 8, 20, 55).
- [63] R. Kolluri, J. R. Shewchuk, and J. F. O’Brien. “Spectral Surface Reconstruction from Noisy Point Clouds”. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. SGP ’04. Nice, France: Association for Computing Machinery, 2004, pp. 11–21 (cit. on pp. 8, 68).
- [64] S. Fuhrmann and M. Goesele. “Floating Scale Surface Reconstruction”. *ACM Trans. Graph.* **33** 4 (2014) (cit. on pp. 8, 17, 18, 68).
- [65] M. Kazhdan and H. Hoppe. “Screened Poisson Surface Reconstruction”. *ACM Trans. Graph.* **32** 3 (2013). <http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version4> (cit. on pp. 8, 9, 11, 12, 16, 17, 68, 69, 98, 99).

- [66] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 303–312 (cit. on pp. 8, 11, 21, 69).
- [67] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. “Surface Reconstruction from Unorganized Points”. *SIGGRAPH Comput. Graph.* **26** 2 (1992), pp. 71–78 (cit. on pp. 8, 21, 33, 68).
- [68] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. “Point Set Surfaces”. In: *Proceedings of IEEE Visualization 2001*. 2001, pp. 21–28 (cit. on p. 8).
- [69] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. Seidel. “Multi-Level Partition of Unity Implicits”. *ACM Transactions on Graphics (TOG)* **22** 3 (2003), pp. 463–470 (cit. on pp. 8, 21, 25, 27, 33, 68).
- [70] C. Shen, J. O’Brien, and J. Shewchuk. “Interpolating and Approximating Implicit Surfaces from Polygon Soup”. In: *Proceedings of ACM Siggraph*. 2004, pp. 896–904 (cit. on pp. 8, 21).
- [71] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, and T. Evans. “Reconstruction and Representation of 3D Objects with Radial Basis Functions”. In: *Proceedings of ACM Siggraph*. ACM. 2001, pp. 67–76 (cit. on pp. 9, 21, 25, 27, 68).
- [72] M. Kazhdan, M. Bolitho, and H. Hoppe. “Poisson surface reconstruction”. In: *Proceedings of the fourth Eurographics Symposium on Geometry Processing*. Eurographics Association. 2006, pp. 61–70. <http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version2> (cit. on pp. 9, 11, 16, 17, 21, 22, 25, 27, 33, 38, 39, 62, 66, 68).
- [73] J. Manson, G. Petrova, and S. Schaefer. “Streaming Surface Reconstruction Using Wavelets”. *Computer Graphics Forum* **27** 5 (2008), pp. 1411–1420 (cit. on pp. 9, 21, 25, 27, 33, 37).
- [74] M. Kazhdan and H. Hoppe. “An Adaptive Multi-Grid Solver for Applications in Computer Graphics”. *Computer Graphics Forum* **38** 1 (2019), pp. 138–150. <https://github.com/mkazhdan/PoissonRecon> (cit. on pp. 9, 11, 17).
- [75] A. Jacobson, L. Kavan, and O. Sorkine-Hornung. “Robust Inside-Outside Segmentation Using Generalized Winding Numbers”. *ACM Trans. Graph.* **32** 4 (2013) (cit. on p. 9).

- [76] W. Lu, Z. Shi, J. Sun, and B. Wang. “Surface Reconstruction Based on the Modified Gauss Formula”. *ACM Trans. Graph.* **38** 1 (2018) (cit. on pp. 9, 11).
- [77] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial, 2nd Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000 (cit. on p. 9).
- [78] J. Nocedal and S. Wright. *Numerical optimization*. 2. ed. Springer series in operations research and financial engineering. New York, NY: Springer, 2006. XXII, 664 (cit. on pp. 9, 32, 46, 61, 75).
- [79] T. Pollard and J. L. Mundy. “Change Detection in a 3-d World”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–6 (cit. on pp. 10, 55, 104–106).
- [80] A. Miller, V. Jain, and J. L. Mundy. “Real-Time Rendering and Dynamic Updating of 3-d Volumetric Data”. In: *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units. GPGPU-4*. Newport Beach, California, USA: Association for Computing Machinery, 2011 (cit. on pp. 10, 55–57, 59, 62, 66).
- [81] D. Crispell, J. Mundy, and G. Taubin. “A Variable-Resolution Probabilistic Three-Dimensional Model for Change Detection”. *IEEE Transactions on Geoscience and Remote Sensing* **50** 2 (2012), pp. 489–500 (cit. on pp. 10, 55–57, 59, 62, 66, 104).
- [82] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-time dense surface mapping and tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136 (cit. on p. 11).
- [83] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. “KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: Association for Computing Machinery, 2011, pp. 559–568 (cit. on p. 11).
- [84] M. Kazhdan, M. Chuang, S. Rusinkiewicz, and H. Hoppe. “Poisson Surface Reconstruction with Envelope Constraints”. *Computer Graphics Forum* **39** (2020), pp. 173–182. <http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version13.00> (cit. on pp. 12, 99).

- [85] C. Schroers, S. Setzer, and J. Weickert. “A Variational Taxonomy for Surface Reconstruction from Oriented Points”. *Computer Graphics Forum* **33** 5 (2014), pp. 195–204 (cit. on pp. 12, 99).
- [86] M. Kazhdan. *Screened Poisson Surface Reconstruction*. <http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version7.0>. Software and data download. 2013 (cit. on pp. 17, 69).
- [87] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009 (cit. on pp. 18, 99, 111).
- [88] R. Bolle and B. Vemuri. “On Three-Dimensional Surface Reconstruction Methods”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1991), pp. 1–13 (cit. on p. 19).
- [89] L. Kobbelt and M. Botsch. “A Survey of Point-Based Techniques in Computer Graphics”. *Computers & Graphics* **28** 6 (2004), pp. 801–814 (cit. on p. 19).
- [90] V. Pratt. “Direct Least-Squares Fitting of Algebraic Surfaces”. In: *Proceedings of ACM Siggraph*. 1987, pp. 145–152 (cit. on pp. 21, 25).
- [91] G. Taubin. “Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1991), pp. 1115–1138 (cit. on pp. 21, 25).
- [92] D. Moore and J. Warren. “Approximation of dense scattered data using algebraic surfaces”. In: *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*. Vol. 1. IEEE. 1990, pp. 681–690 (cit. on p. 21).
- [93] C. Bajaj, I. Ihm, and J. Warren. “Higher-Order Interpolation and Least-Squares Approximation Using Implicit Algebraic Surfaces”. *ACM Transactions on Graphics (TOG)* **12** 4 (1993), pp. 327–347 (cit. on p. 21).
- [94] B. Jüttler and A. Felis. “Least-Squares Fitting of Algebraic Spline Surfaces”. *Advances in Computational Mathematics* **17** 1 (2002), pp. 135–152 (cit. on pp. 21, 25).
- [95] H. Zhao, S. Osher, B. Merriman, and M. Kang. “Implicit and Nonparametric Shape Reconstruction from Unorganized Data Using a Variational Level Set Method”. *Computer Vision and Image Understanding* **80** 3 (2000), pp. 295–314 (cit. on p. 21).

- [96] M. Kazhdan. “Reconstruction of Solid Models from Oriented Point Sets”. In: *Proceedings of the 3rd. Eurographics symposium on Geometry processing*. Eurographics Association. 2005, p. 73 (cit. on pp. 21, 22, 25, 27, 39).
- [97] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. “Voronoi-based variational reconstruction of unoriented point sets”. In: *Proceedings of the fifth Eurographics symposium on Geometry processing*. Eurographics Association. 2007, pp. 39–48 (cit. on pp. 21, 68).
- [98] J. Boissonnat and F. Cazals. “Smooth surface reconstruction via natural neighbour interpolation of distance functions”. *Computational Geometry* **22** 1 (2002), pp. 185–203 (cit. on p. 21).
- [99] Y. Ohtake, A. Belyaev, and H. Seidel. “A Multi-Scale Approach to 3D Scattered Data Interpolation with Compactly Supported Basis Functions”. In: *International Conference on Shape Modeling and Applications*. 2003, pp. 153–161 (cit. on pp. 21, 25, 27).
- [100] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. “Computing and Rendering Point Set Surfaces”. *IEEE Transactions on Visualization and Computer Graphics* (2003), pp. 3–15 (cit. on pp. 21, 68).
- [101] S. Fleishman, D. Cohen-Or, and C. T. Silva. “Robust Moving Least-Squares Fitting with Sharp Features”. *ACM Trans. Graph.* **24** 3 (2005), pp. 544–552 (cit. on p. 21).
- [102] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. “Multilevel Streaming for Out-Of-Core Surface Reconstruction”. In: *Proceedings of the 5th. Eurographics Symposium on Geometry Processing*. Eurographics Association. 2007, pp. 69–78 (cit. on p. 21).
- [103] K. Zhou, M. Gong, X. Huang, and B. Guo. “Data-Parallel Octrees for Surface Reconstruction”. *IEEE Transactions on Visualization and Computer Graphics* **17** 5 (2011), pp. 669–681 (cit. on p. 21).
- [104] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. “Parallel Poisson Surface Reconstruction”. In: *Proceedings of the 5th International Symposium on Advances in Visual Computing*. Springer, 2009, pp. 678–689 (cit. on p. 21).
- [105] K. Kang, J.-P. Tarel, R. Fishman, and D. Cooper. “A Linear Dual-Space Approach to 3D Surface Reconstruction from Occluding Contours using Algebraic Surfaces”. In: *In International Conference on Computer Vision*. 2001, pp. 198–204 (cit. on p. 25).

- [106] T. Lewiner, V. Mello, A. Peixoto, S. Pesco, and H. Lopes. “Fast Generation of Pointerless Octree Duals”. *Computer Graphics Forum* **29** 5 (2010), pp. 1661–1669 (cit. on p. 30).
- [107] D. Crispell, D. Lanman, P. Sibley, Y. Zhao, and G. Taubin. “Beyond silhouettes: Surface reconstruction using multi-flash photography”. In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’2006)*. IEEE. 2006, pp. 405–412 (cit. on p. 36).
- [108] P. Cignoni, C. Rocchini, and R. Scopigno. “Metro: Measuring Error on Simplified Surfaces”. *Computer Graphics Forum* **17** 2 (1998), pp. 167–174 (cit. on p. 37).
- [109] P. Sibley and G. Taubin. “Vectorfield Isosurface-Based Reconstruction from Oriented Points”. In: *ACM Siggraph 2005 Sketches*. ACM. 2005, p. 29 (cit. on pp. 39, 40).
- [110] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva. “A benchmark for surface reconstruction”. *ACM Transactions on Graphics (TOG)* **32** 2 (2013), pp. 1–17 (cit. on pp. 39, 55).
- [111] F. Calakli and G. Taubin. *Smooth Signed Distance Surface Reconstruction*. <http://mesh.brown.edu/ssd>. Software and data download. 2011 (cit. on p. 40).
- [112] A. Baumberg. “Blending Images for Texturing 3D Models”. In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2002, pp. 38.1–38.10 (cit. on p. 41).
- [113] M. Waechter, N. Moehrl, and M. Goesele. “Let There Be Color! Large-Scale Texturing of 3D Reconstructions”. English. In: *ECCV 2014*. Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Vol. 8693. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 836–850 (cit. on pp. 42, 61).
- [114] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. “A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. 2006, pp. 519–528. <http://vision.middlebury.edu/mview/eval> (cit. on pp. 48, 54, 56, 62, 66, 67).
- [115] Y. Furukawa and C. Hernández. “Multi-view stereo: A tutorial”. *Foundations and Trends in Computer Graphics and Vision* **9** 1-2 (2015), pp. 1–148 (cit. on p. 54).
- [116] C. H. Esteban and F. Schmitt. “Silhouette and stereo fusion for 3D object modeling”. *Computer Vision and Image Understanding* **96** 3 (2004), pp. 367–392 (cit. on p. 54).

- [117] G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. “Multiview Stereo via Volumetric Graph-Cuts and Occlusion Robust Photo-Consistency”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29** 12 (2007), pp. 2241–2246 (cit. on p. 54).
- [118] K. Kolev and D. Cremers. “Integration of Multiview Stereo and Silhouettes Via Convex Functionals on Convex Domains”. In: *European Conference on Computer Vision*. Springer. 2008, pp. 752–765 (cit. on p. 54).
- [119] A. Broadhurst, T. W. Drummond, and R. Cipolla. “A Probabilistic Framework for Space Carving”. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. 2001, 388–393 vol.1 (cit. on p. 55).
- [120] R. Bhotika, D. J. Fleet, and K. N. Kutulakos. “A Probabilistic Theory of Occupancy and Emptiness”. In: *European Conference on Computer Vision*. Springer. 2002, pp. 112–130 (cit. on p. 55).
- [121] K. N. Kutulakos and S. M. Seitz. “A Theory of Shape by Space Carving”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. 1999, 307–314 vol.1 (cit. on p. 55).
- [122] P. Gargallo, P. Sturm, and S. Pujades. “An Occupancy-Depth Generative Model of Multi-View Images”. In: *Proceedings of the 8th Asian Conference on Computer Vision - Volume Part II. ACCV’07*. Tokyo, Japan: Springer-Verlag, 2007, pp. 373–383 (cit. on p. 55).
- [123] S. Liu and D. B. Cooper. “A complete statistical inverse ray tracing approach to multi-view stereo”. In: *CVPR 2011*. 2011, pp. 913–920 (cit. on pp. 55, 56, 104).
- [124] D. E. Crispell. *A Continuous Probabilistic Scene Model for Aerial Imagery*. PhD thesis. School of Engineering, Brown University, 2010 (cit. on pp. 55, 56, 60, 66).
- [125] J. .-. Franco and E. Boyer. “Fusion of multiview silhouette cues using a space occupancy grid”. In: *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*. Vol. 2. 2005, 1747–1753 Vol. 2 (cit. on p. 56).
- [126] A. Yezzi, G. Slabaugh, R. Cipolla, and R. Schafer. “A surface evolution approach of probabilistic space carving”. In: *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. 2002, pp. 618–621 (cit. on p. 56).

- [127] V. Lempitsky, Y. Boykov, and D. Ivanov. “Oriented Visibility for Multiview Reconstruction”. In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part III*. ECCV’06. Graz, Austria: Springer-Verlag, 2006, pp. 226–238 (cit. on p. 56).
- [128] C. Hernández, G. Vogiatzis, and R. Cipolla. “Probabilistic visibility for multi-view stereo”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8 (cit. on p. 56).
- [129] V. Lempitsky and Y. Boykov. “Global Optimization for Shape Fitting”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8 (cit. on pp. 56, 69).
- [130] M. I. Restrepo, O. A. Mayer, and J. L. Mundy. “Object Recognition in Probabilistic 3D Volumetric Scenes”. In: *In ICPRAM, 2012*. 6 (cit. on p. 62).
- [131] P. Mullen, F. De Goes, M. Desbrun, D. Cohen-Steiner, and P. Alliez. “Signing the Unsigned: Robust Surface Reconstruction from Raw Pointsets”. *Computer Graphics Forum* **29** 5 (2010), pp. 1733–1741 (cit. on p. 68).
- [132] J. Chen, G. Guennebaud, P. Barla, and X. Granier. “Non-Oriented MLS Gradient Fields”. *Computer Graphics Forum* **32** 8 (2013), pp. 98–109 (cit. on p. 68).
- [133] J. H. Friedman, J. L. Bentley, and R. A. Finkel. “An Algorithm for Finding Best Matches in Logarithmic Expected Time”. *ACM Trans. Math. Softw.* **3** 3 (1977), pp. 209–226 (cit. on pp. 68, 107, 109).
- [134] C. Zach, T. Pock, and H. Bischof. “A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration”. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8 (cit. on p. 69).
- [135] S. Fuhrmann and M. Goesele. “Fusion of Depth Maps with Multiple Scales”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. SA ’11. Hong Kong, China: Association for Computing Machinery, 2011 (cit. on p. 69).
- [136] P. Labatut, J.-P. Pons, and R. Keriven. “Robust and Efficient Surface Reconstruction From Range Data”. *Computer Graphics Forum* **28** 8 (2009), pp. 2275–2290 (cit. on p. 69).
- [137] V. Lempitsky. “Surface extraction from binary volumes with higher-order smoothness”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1197–1204 (cit. on p. 69).

- [138] D. R. Hunter and K. Lange. “A Tutorial on MM Algorithms”. *The American Statistician* **58** 1 (2004), pp. 30–37 (cit. on pp. 70, 71).
- [139] V. Estellers, M. Scott, K. Tew, and S. Soatto. “Robust Poisson Surface Reconstruction”. In: *Scale Space and Variational Methods in Computer Vision*. Ed. by J.-F. Aujol, M. Nikolova, and N. Papadakis. Cham: Springer International Publishing, 2015, pp. 525–537 (cit. on p. 98).
- [140] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers”. *Found. Trends Mach. Learn.* **3** 1 (2011), pp. 1–122 (cit. on p. 99).
- [141] A. Laurentini. “The visual hull concept for silhouette-based image understanding”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16** 2 (1994), pp. 150–162 (cit. on p. 99).
- [142] A. Bogomjakov, C. Gotsmann, and M. Magnor. “Free-Viewpoint Video from Depth Cameras”. In: *Proc. Vision, Modeling and Visualization (VMV)*. Eurographics. 2006, pp. 89–96 (cit. on p. 99).
- [143] C. Stauffer and W. E. L. Grimson. “Adaptive background mixture models for real-time tracking”. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 2. 1999, 246–252 Vol. 2 (cit. on p. 106).
- [144] C. Harris and M. Stephens. “A combined corner and edge detector”. In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151 (cit. on p. 108).
- [145] A. Rényi. “On Measures of Entropy and Information”. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif.: University of California Press, 1961, pp. 547–561 (cit. on p. 108).
- [146] G. Gallego and A. Yezzi. “A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates”. *Journal of Mathematical Imaging and Vision* **51** (2015), pp. 378–384 (cit. on p. 110).