

Washington University in St. Louis

Washington University Open Scholarship

McKelvey School of Engineering Theses & Dissertations

McKelvey School of Engineering

Spring 5-15-2019

Toward Controllable and Robust Surface Reconstruction from Spatial Curves

Zhiyang Huang

Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Huang, Zhiyang, "Toward Controllable and Robust Surface Reconstruction from Spatial Curves" (2019). *McKelvey School of Engineering Theses & Dissertations*. 448.
https://openscholarship.wustl.edu/eng_etds/448

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in McKelvey School of Engineering Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering & Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:

Tao Ju, Chair
Nathan Carr
Ayan Chakrabarti
Ulugbek Kamilov
Caitlin Kelleher

Toward Controllable and Robust Surface Reconstruction from Spatial Curves
by
Zhiyang Huang

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2019
St. Louis, Missouri

© 2019, Zhiyang Huang

Table of Contents

List of Figures	v
List of Tables	x
Acknowledgments	xi
Abstract	xiv
Chapter 1: Introduction	1
1.1 Reconstruction from cross-sections.....	3
1.2 Reconstruction from wire-frames	5
1.3 Overview.....	6
Chapter 2: Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections	8
2.1 Introduction.....	8
2.2 Related works	12
2.2.1 Modeling multi-labeled domains	12
2.2.2 Topology-aware modeling of two-labeled domains.....	12
2.3 Interface sets.....	14
2.3.1 Definition and properties	15
2.3.2 Discrete topological variations.....	17
2.4 Reconstruction algorithm.....	25
2.4.1 Enumeration	27
2.4.2 Selection.....	29
2.5 User interaction	31
2.6 Results	32
2.6.1 Performance.....	35
2.7 Conclusion and discussion.....	36

2.7.1	Limitations	36
Chapter 3: Repairing Inconsistent Curve Networks on Non-parallel Cross-sections		38
3.1	Introduction.....	38
3.2	Relate work	41
3.3	Problem formulation.....	42
3.3.1	Implicit representation	43
3.3.2	Deformation energy	47
3.3.3	Optimization formulation.....	48
3.4	Optimization.....	51
3.4.1	Initial labels	52
3.4.2	Updating labels	54
3.5	Experimental results.....	58
3.6	Conclusion and discussion.....	68
Chapter 4: Variational Implicit Point Set Surfaces		69
4.1	Introduction.....	69
4.2	Related Works.....	73
4.2.1	Surface reconstruction from points	73
4.2.2	Normal estimation.....	76
4.3	Definition	77
4.3.1	A general definition	78
4.3.2	Duchon's energy.....	80
4.3.3	Definition using Duchon's energy.....	82
4.4	Implementation	85
4.4.1	Initializing the optimization.....	86
4.4.2	Complexity analysis.....	89
4.5	Experiments	90
4.5.1	Results.....	90
4.5.2	Comparisons	94
4.5.3	Performance	100

4.5.4	Application: sketch surfacing	103
4.6	Conclusion and limitations	104
Chapter 5:	Conclusion and future work	106
5.1	Future work	108
5.1.1	Analytical formulation of critical offsets	108
5.1.2	Finer level topological control	108
5.1.3	Incremental framework for speeding up VIPSS	109
5.1.4	VIPSS for deforming points	111
5.1.5	Topologically-controlled VIPSS	111
5.1.6	Learning-based method for surfacing	112
References	113
Appendix A: Properties of VIPSS	[133]
A.1	Exact interpolation.....	[133]
A.2	Linear reproduction	[134]
A.3	Commutativity with similarity transformations	[134]

List of Figures

Figure 1.1:	Examples of surface reconstruction from cross-sectional curves (top) and wire-frames (bottom): (a) reconstruction of an atrium from cross-sections in 2-labeled domain, (b) reconstruction of a chicken heart from cross-sections in multi-labeled domain (the right one is a cut-away view), (c) reconstruction of a hand from free-sketch, (d) three step of reconstructing a hand from clean wire-frames.	2
Figure 1.2:	Example of topology description of shapes with respect to number of connected components and genus.	5
Figure 2.1:	Given several multi-labeled planes depicting the anatomical regions of a mouse brain (a), reconstruction without topology control (b1) leads to redundant handles for the red and yellow labels (black arrows in c1, d1) and disconnection for the green label (e1). Our method (b2) allows the user to prescribe the topology such that the red label has one tunnel (gray arrow in c2), the yellow label has no tunnels (d2), and the green label is connected (e2). The legends in (b1,b2) report, for each label in the reconstruction, the genus of each surface component bounding that label (e.g., “0,0” means two surfaces each with genus 0). User-specified constraints are colored red.	9
Figure 2.2:	Comparing the topology-oblivious multi-labeled method of Bermano et al. [18] and Liu et al. [94] (b), the topology-constrained two-labeled method of Zou et al. [166] (c), and our topology-constrained multi-labeled method (d) on two cross-sections with three labels (a). Cutaway views are shown in inserts. Legends report the per-component genus for each label (red numbers are constrained).	14
Figure 2.3:	Interface sets in 2D: (a) The input vector function $\vec{f} = \{f_1, f_2, f_3\}$, visualized as three height maps. (b,c,d): Three different choices of offsets $\vec{c} = \{c_1, c_2, c_3\}$ (top), superimposed height maps $f_i + c_i$ (middle), and the labeling (as color) and interface sets (as black curves) in the 2D domain (bottom).	18

Figure 2.4:	(a) A topological change of interface sets in PL interpolation can take place in arbitrary locations: lowering the offset of the yellow label causes the yellow region to disappear inside the triangle. (b): A topological change in our PC interpolation is restricted to the dual of the input complex. The offseted functions are shown at the top and the interface sets are shown at the bottom.	20
Figure 2.5:	Topology analysis of 3-labeled interface sets. (a) A complex C where each vertex is colored by the associated 3-vector. (b) The piecewise constant vector function shown as one height map for each label. (c) The active complex in the 3D offset space consisting of one triple half-plane (“triblade”) for each vertex of C . (d) A cutaway of the offset space (by the gray plane in (c)) showing the active complex (colored lines) and the critical complex (black lines). (e) Example interface sets in different pockets (see pocket labeling in (d)).	21
Figure 2.6:	Reconstruction algorithm: starting from the plane arrangement (a, showing 3 cells divided by the two cross-section planes), our algorithm first enumerates and scores topologies of interface sets within each cell (b, interface sets in each cell are ordered by decreasing scores), then one topology is selected per cell to achieve the topological constraints while maximizing the total score (c, showing solutions under two sets of constraints marked in red; letters by the cutaway views at the bottom are choices of cell topologies).	31
Figure 2.7:	User interactions: a 4-labeled input (a) and reconstructions without topological constraints (b), with genus-0 constraint on the red label (c), after the user picks a different topology in one of the cells (d), and after the user adds scribbles in that cell (e). All reconstructions are shown in cutaway views as their exterior shapes are similar to that in (b).	33
Figure 2.8:	The Liver data set (a) and reconstructions without topological constraints (b), with some (c) and more (d) constraints, and after applying scribbles (e). Cutaway views are shown in the inserts, and the legends report the per-component genus for each label (constrained genus are in red). Arrows point to topological issues, where the solution does not meet users expectation. See detailed explanations in Section 2.5..	34
Figure 2.9:	The chicken heart data set (a) and reconstruction with topological constraints on 5 labels (b, cutaway view in c). (d) compares the reconstruction of orange (top) and light-green (bottom) labels without and with topological constraints. The surfaces in (d2,d3,d5,d6) are colored by labels of adjacent sub-domains to reveal the intertwining of labels.	35

Figure 3.1:	Curve networks on two intersecting planes (p1,p2) with inconsistent (left) and consistent (right) labeling. The pictures at the bottom show the labeling on each plane as well as the labeling from the other plane on the intersection line (l).	40
Figure 3.2:	Vector function $\vec{f} = \{f_1, f_2, f_3\}$ defined as signed distance functions over a three-labelled 1D domain. Note that the difference function $f_2 - f_3$ (magenta dotted graph) is a distance-like function locally at the interface between labels 2 and 3.....	44
Figure 3.3:	Overview of our algorithm. Each input slice (a) is first triangulated (b), and a vector function is computed per slice to reproduce the input labels (c). Then functions on all slices are optimized together to enforce label consistency while minimizing deformations (d). Finally, the output curve networks are extracted as the interface sets of the optimized functions (e).	45
Figure 3.4:	Optimization process on the input in Figure 3.1 (left, plane p2), showing the labeling on the plane (as red, blue, gray colors) and interface set (green curves) in the input (a), after initializing the labels on the intersection lines (b) (see Section 4.1), and after the first (c) and final (d) iterations of label updates (see Section 4.2). Interface sets in previous steps are shown in white curves in subsequent steps in (b,c,d) for comparison, and locations where vertices change labels are indicated by arrows.	57
Figure 3.5:	Results of our method (on input in Figure 3.1 left, showing plane p2) for different values of λ in Equation 3.2. The labeling is shown as colored regions, and the input curve network is shown as gray curves for reference.....	59
Figure 3.6:	The result (top-right) of repairing an inconsistent two-labelled Atrium data set (top-left, several inconsistencies are highlighted), and surfaces reconstructed from these two sets of slices using [18] (bottom; observe the artifacts in bottom-left).	63
Figure 3.7:	The result (right) of repairing a two-labelled ferret brain data set that is highly inconsistent (left, one inconsistency is highlighted). The bottom pictures show the labeling on one of the planes (p) as well as labelling from other planes on intersection lines.	64
Figure 3.8:	The result (right) of repairing a 4-labelled liver data set (left, two inconsistencies are highlighted), showing the labeling on two planes (p1,p2) at the bottom.	65

Figure 3.9:	The result (right) of repairing another 4-labelled liver data set (left, two inconsistencies are highlighted), showing labelling on two planes (p1,p2) at the bottom.	66
Figure 3.10:	Result (bottom) of repairing an inconsistent 7-labelled mouse brain set (top), showing the labeling on three planes (p1,p2,p3) and labeling on other planes along the intersection lines. A few inconsistencies are highlighted in black boxes.....	67
Figure 4.1:	Given sparse, non-uniform, noisy and un-oriented points (b) sampled from a set of unstructured 3D curves (a), our variational definition (VIPSS with $\lambda = 0.003$) simultaneously produces oriented normals (c) and a smooth approximating surface (d). The input is challenging for state-of-the-art normal estimation methods such as [154], which fails around sparsely sampled thin features (the flippers) (e). Incorrect normals lead to poor reconstructions using existing implicit methods such as Screened Poisson [86] (f, fitting weight $\alpha = 0.5$).	71
Figure 4.2:	Examples of Duchon’s interpolants $f_{\mathbf{s},\mathbf{g}}$ that interpolate scattered points in 1D (top, red dots) and 2D (bottom, red circles) for different choices of the Hermite data $\{\mathbf{s}, \mathbf{g}\}$. In (a), $s_i = 0$ and \mathbf{g}_i is a constant vector at each point. In (b,c,d), $\{\mathbf{s}, \mathbf{g}\}$ are obtained by our variational formulation (4.9) with $\lambda = 0, 0.1, 1.0$ respectively. The zero-level set in (b,c,d) (black curves) is the VIPSS at the respective λ	82
Figure 4.3:	Initial vectors generated using the spectral method with $\lambda = 0$ (a) contains vectors with very small magnitudes and flipped orientations (see insert, vectors shown with 20x scaling), which leads to a high-energy result after optimization (b). Initial vectors generated with $\lambda = 0.01$ (c) and 0.1 (e) are more uniform, and they lead to the same low-energy result (d,f) after optimization with $\lambda = 0$	87
Figure 4.4:	Top row: sampling a torus surface with decreasing density (a,b,c,d with 500, 200, 50, 25 points respectively), varying sampling density (e), missing samples (f,g), and along 1-dimensional curves (h,i). Middle row: optimized vectors \mathbf{g} visualized as oriented disks (green/blue: front/back side). Bottom row: the VIPSS ($\lambda = 0$) colored by distance from the original torus surface (blue/red: small/large distance).	91
Figure 4.5:	VIPSS ($\lambda = 0$) for samples from Max Planck at different densities. ...	92
Figure 4.6:	Samples from Hand, Vertebra, Kitten (left, each containing 500 points), optimized vectors \mathbf{g} (middle), and VIPSS (right, $\lambda = 0$).	93

Figure 4.7:	Samples from wireframes Trebol, Dog, Phone (left, containing 500, 1000, 1000 points), optimized vectors \mathbf{g} (middle), and VIPSS (right, $\lambda = 0$).	95
Figure 4.8:	(a): Two sampling of the Kitten (500 points each) at low (top, 1%) and high (bottom, 5%) noise rate. (b,c): VIPSS with $\lambda = 0.001$ and 0.01.	96
Figure 4.9:	Comparing direction estimation on samples from a 3D wireframe using PCA [72] and VCM [100] with different parameters. Each un-oriented direction is shown by a yellow tangent disk and a line segment.	98
Figure 4.10:	Comparing normal estimation from a 800-point sample from the Bath-tub (cross-section shown in top-right) using VIPSS ($\lambda = 0$), variational method of [154], and PCPNet [67] (on a 3000-point sampling). The surfaces for these methods are generated using Hermite RBF interpolation (i.e., zero-level set of $f_{\mathbf{s},\mathbf{g}}$ where $\mathbf{s} = 0$ and \mathbf{g} are the estimated normals).	99
Figure 4.11:	Comparing Hermite RBF (Duchon’s interpolant) (b) with Screened Poisson [86] (c,d) and APSS [66] (e,f) at different parameters on the same oriented input with 1000 points (a). Orientations in (a) are computed by our method with $\lambda = 0$	101
Figure 4.12:	Comparing VIPSS ($\lambda = 0$) with methods that do not require oriented inputs: the Voronoi-based variational method of [5], the ball-pivoting method [19], the tight cocone [49], and the power crust [7].	102
Figure 4.13:	VIPSS ($\lambda = 0$) vectors (b) and surface (c) for samples from an unstructured sketch (a). The inserts take a closer look between the index and ring fingers (the line segments in the insert of (b) indicate $-\mathbf{g}$).	104
Figure 5.1:	Given a stack of 5-labeled slices (a) (only blue and green labels touch on each slice), reconstruction with genus-0 constraint on each label produces multiple patches of interface between the blue and green labels (c), whereas a further modification of the algorithm results in a contiguous interface (d) satisfying the same topology constraints. In (c,d) we only show the surfaces of the blue label colored by its adjacent labels, and junction curves and points are shown as grey wires and red balls. The exterior surface of the reconstruction in (d) is shown in (b).	110

List of Tables

Table 2.1:	Running time of the two stages of our algorithm on the mouse brain (Fig 2.1), liver (Fig 2.8), and chicken heart (Fig 2.9). Also showing the number of constrained labels, number of tetrahedra, and maximum number of per-cell topologies.	36
Table 3.1:	Data size and running time for the examples in Figures 3.6, 3.7, 3.8, 3.9, 3.10, showing the number of planes, number of labels, total number of vertices in the triangulations, number of vertices in the reduced intersection set I , λ value, and timing (in seconds) for each of the three stages of our method. ('-' indicates the solver fails to return within 2 hours)	61
Table 3.2:	Comparing our optimization method and the MIP solver in Gurobi on a subset of k planes in the ferret brain data, in terms of minimal energy and time (in seconds). Column 5, 6, 7 show the time on reduced set, and column 8, 9, 10 show the time on original set on intersection lines.	62
Table 4.1:	Running time (in seconds) of each step for Figure 4.5 recorded on a MacBook Pro with 2.5GHz Intel Core i7 CPU and 16 GB memory (implementation done in C++). Timing for Optimization step is written as initialization time (using the offset method of Section 4.4.1) + NLOPT time. Surfacing uses a 100^3 grid.	102

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Tao Ju for his guidance and support during the past five years. He is an amazing professor, showing me the beauty of computer science, as well as providing invaluable instruction and feedbacks for my research. None of the work in the dissertation would have been accomplished without him. His enthusiasm and rigorousness for research lighten me through the road of my Ph.D. career. I will always cherish my time as his student.

I would like to thank Nathan Carr, our amazing collaborator from Adobe. Nathan always brings us invaluable insights. He is patient and kind, and it is both enjoyable and beneficial to work with Nathan. I would like to also thank my other committee members Ayan Chakrabarti, Ulugbek Kamilov and Caitlin Kelleher for their thoughtful suggestion in my proposal and dissertation.

I would like to thank my lab-mates Ming Zou, Michelle Holloway, Hand Dou, Yajie Yan, Chen Liu, Hang Yan and Dan Zeng for the companionship on pursuing the Ph.D. degree. They are the best comrades and thought challengers. I will always value our time on collaborating projects and brainstorming ideas.

I would like to thank all my dear friends, for sharing all the wonderful stories and happiness in this journey.

I would like to thank my family. Their unconditional love and support make me feel fearless toward every challenge in my life.

Zhiyang Huang

Washington University in Saint Louis

May 2019

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Toward Controllable and Robust Surface Reconstruction from Spatial Curves

by

Zhiyang Huang

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2019

Professor Tao Ju

Reconstructing surface from a set of spatial curves is a fundamental problem in computer graphics and computational geometry. It often arises in many applications across various disciplines, such as industrial prototyping, artistic design and biomedical imaging. While the problem has been widely studied for years, challenges remain for handling different type of curve inputs while satisfying various constraints. We study studied three related computational tasks in this thesis. First, we propose an algorithm for reconstructing multi-labeled material interfaces from cross-sectional curves that allows for explicit topology control. Second, we addressed the consistency restoration, a critical but overlooked problem in applying algorithms of surface reconstruction to real-world cross-sections data. Lastly, we propose the Variational Implicit Point Set Surface which allows us to robustly handle noisy, sparse and non-uniform inputs, such as samples from spatial curves.

Chapter 1

Introduction

Modeling a shape from its partial or non-uniform sampling is a common problem in many scientific and engineering fields. An example problem is to reconstruct an interpolating surface from spatial curves describing the shape. This task often arises in biomedical research, where a 3D representation (e.g., a surface) of the target anatomical structure needs to be reconstructed from its planar cross-sectional curves obtained from 3D volume imaging by MRI or CT. Another important application of this problem is computer-aided industrial prototyping and design, where engineers and artists typically start by defining a wire-frame of the desired model, which is then turned into a surface representation for simulation, rendering, and manufacturing. Examples of surface reconstruction from curves are shown in Figure 1.1.

A reconstructed surface has to meet certain correctness criteria. Commonly, the surface should interpolate the curves, and should be geometrically valid (i.e., free of holes and intersections), so it properly defines the boundary of the shape described by those curves. However, the inherent sparsity and non-uniformness of curves pose a great challenge to the reconstruction algorithm, mainly in two aspects. The first one is the ambiguity of the shape conveyed by

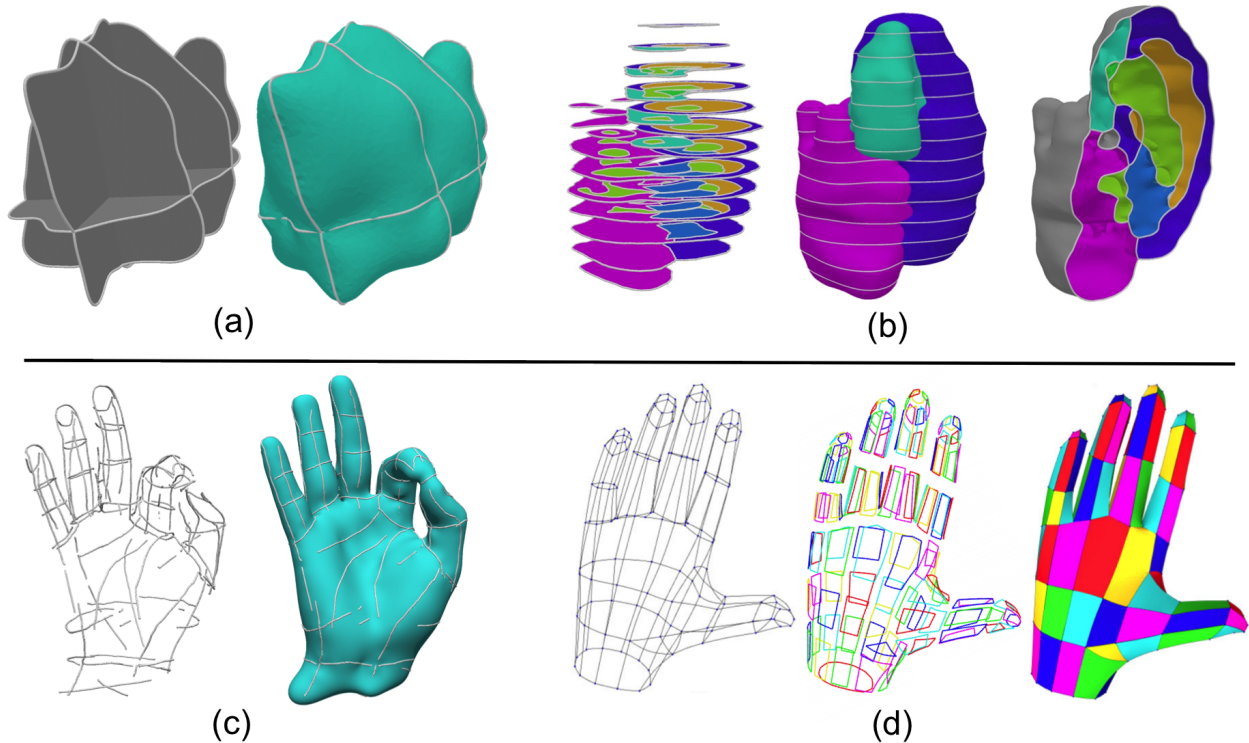


Figure 1.1: Examples of surface reconstruction from cross-sectional curves (top) and wire-frames (bottom): (a) reconstruction of an atrium from cross-sections in 2-labeled domain, (b) reconstruction of a chicken heart from cross-sections in multi-labeled domain (the right one is a cut-away view), (c) reconstruction of a hand from free-sketch, (d) three step of reconstructing a hand from clean wire-frames.

the curve representation. Compared to the high-quality point cloud where sampling are usually spread over the shape uniformly, in curve representation, the sampling of the shape is concentrated upon a few curve segments and the majority of space is left empty. This leads to ambiguity in interpreting the empty regions, and thus the global shape. The second challenge arises from the possible noise of curves in practices. While the set of curves could be incomplete, disjoint and inaccurately-placed, it is difficult to design geometry algorithms for handling all unexpected irregularities. The computer graphics community has studied this task for years with numerous algorithms being developed. An effective way for leveraging the ambiguity is looking for additional data/prior, for instance, the utilization of the 3D volume from MRI, or other domain knowledge about the shape such as topology. Such prior

is usually transformed into constraints on the output surface. How to incorporate those prior with the reconstruction algorithm, or in other words, how to enforce the constraints on the output surface, is an interesting problem being actively researched. On the other side, although important, little progress has been made to tackle the noise of the curves which is commonly seen in practice. Existing methods mostly require "clean" inputs which satisfy various assumption, and simply fail on "dirty" data (See Sec 1.1, 1.2).

As mentioned above, there are two common types of input curves that are drawing people's attention: cross-sectional curves of anatomical shapes and design-created wire-frames. We first briefly review the two bodies of works.

1.1 Reconstruction from cross-sections

Cross-section inputs often arise in biomedicine, where experts delineate boundaries of anatomical regions on 2D slices of a 3D medical image (e.g., MRI, CT). Cross-section inputs can also be found in other disciplines, such as material science (e.g., sectioned micrographs) and geology (e.g., seismic images). Cross-sections can be parallel or non-parallel, and the cross-section curves can partition each plane into regions of two (e.g., inside and outside) or more (e.g., bone, muscle, fat, etc.) labels (See Fig 1.1 (a) (b)). Take the chicken heart in Fig 1.1 (b) as an example, cross-section curves (left) partition each plane into regions of 7 labels indicated by different color, representing muscles or regions of different functionalities (e.g., yellow represents atrial).

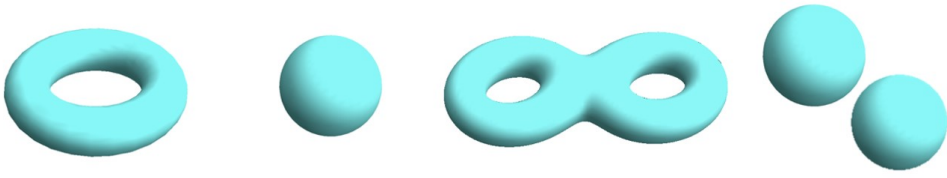
Since the 70's, extensive research has been conducted on reconstructing surfaces from cross-sectional curves. Earlier methods focused on handling parallel cross-sections that are partitioned by closed curve loops into inside and outside regions [87, 62, 26, 13, 111, 146, 12, 14]. The key idea in these methods is to divide the space (or more practically, a bounding box)

by the cross-sectional planes into “cells” and build surface pieces within each cell. When all planes are parallel, all cells have a uniform and simple shape (a slab) that is bounded by two planes. If the planes are arbitrarily oriented, each cell may be a general convex polytope bounded by an arbitrary number of planes, which makes the surfacing task more challenging.

Over the past decade, research on cross-section-based reconstruction has focused on handling non-parallel inputs. A number of methodologies have emerged including Delaunay meshing [28], projecting curves onto a medial structure [94, 15], solving implicit functions [18, 70, 166, 77], and template fitting [71]. Some of these algorithms are capable of handling even more general inputs such as multi-labeled cross-sections [94, 15, 18], partial planes [15], and unknown regions [18].

While all the above algorithms can generate geometrically valid surfaces that interpolate the curves, few can ensure that the output has a correct *topology*. Topology is an intrinsic property of a shape that is invariant under continuous deformations. For a 2-label domain, the topology of can be measured by the number of components and the number of handles or genus for each component (See Fig 1.2). Most anatomical structures have a fixed and known topology, and obtaining a topologically correct surface (i.e., a surface with the prescribed number of connected components and genus) is particularly important for downstream tasks such as shape matching and mechanical or fluid simulation. The only topology-aware reconstruction method from cross-sections that we are aware of is by Zou et al. [166], which is designed for the 2-labeled domain.

A common assumption of all existing reconstruction methods, when applied to non-parallel cross-sections, is that two intersecting cross-sections should be consistent along the intersection line (e.g., a point on the intersection line should have the same label on both cross-sections). However, this assumption often fails in practice, because curves on one cross-section are often



Components	1	1	1	2
Genus	1	0	2	0,0

Figure 1.2: Example of topology description of shapes with respect to number of connected components and genus.

drawn manually and independently from curves on other cross-sections. Given an inconsistent set of cross-sections, existing reconstruction methods would either fail to produce any surface, or produce surfaces with notable artifacts around the inconsistencies in the input.

1.2 Reconstruction from wire-frames

Descriptive wire-frames are often used in computer-aided design. Several computer-based tools exist [158, 21] that allow experts and artists to create wire-frames that effectively describe a 3D shape.

Surfacing such wire-frames is often known as lofting or skinning, which is a fundamental problem in computer-aided design. Existing methods follow a two-step strategy (See Fig 1.1 (d)), first identifying cycles in the wire-frames which bound individual surface patches, and secondly forming patches in each cycle and glue them to form a complete surface. There are numbers of methods [1, 165, 2] for locating curve cycles and several others [113, 1, 158] for surfacing individual patches.

The two-step strategy only works for "clean" inputs that are made up of complete curve segments meeting at well-identified joints (i.e., a spatial graph). While such networks can be

created using the aforementioned tools, the creation process is often tedious (e.g., the user has to explicitly connect the curves at joints). With the advance of AR/VR devices, more intuitive drawing interfaces have been developed that permit a free-hand drawing experience (e.g., the Tilt Brush by Google). However, the output of these tools typically consists of incomplete curve fragments without connectivity information (e.g., Fig 1.1 (c) left). Hence the existing two-step strategy cannot be used for surfacing such wire-frames.

1.3 Overview

In this thesis, we developed algorithms for reconstructing surfaces from spatial curves that can control/guarantee certain properties on the output side (Chapter 2), as well as explored ways to make the algorithm robust against noisy and irregular inputs (Chapter 3, 4).

In Chapter 2 we focus on reconstructing surfaces from cross-sectional curves with topological constraints. In this work, we extend Zou’s [166] method to multi-labeled domain. Given a set of cross-sections, each partitioned into regions of multiple labels by a curve network, our algorithm produces an interpolating surface network such that the surface bounding each label satisfies a prescribed topology such as numbers of components and genus simultaneously (see Figure 1.2). The key contribution is we extend the well-known level-set to interface-set, which allow us to explore a variety of topology variance in the multi-labeled domain rather than the 2-labeled domain. This work has been published in [77].

In Chapter 3, we address a critical but overlooked problem of handling real-world cross-section curves that are often inconsistent with each other by developing a novel algorithm that can restore consistency to any set of cross-sections in multi-labeled domain while minimizing the change to the curve shape. We formulate the problem into a disjunctive programming and propose an effective solution for the optimization. This work has been published in [75].

In Chapter 4, instead of fixing the curves, we explore another direction for handling noise and irregularity by proposing a new method for directly reconstructing an implicit surface from an un-oriented point set. Our method only involves a single parameter, is easy to be implemented without discretizing the space, and most importantly, is robust to sampling imperfection such as sparse and non-uniform inputs. We credit such nice properties to our key contribution which is the global variational definition of the implicit surface.

Chapter 2

Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections

2.1 Introduction

Computational modeling of multi-labeled domains arises in many disciplines, such as biomedicine (e.g., organs made up of multiple anatomical regions) and mechanical engineering (e.g., machine pieces made up of blocks of different materials). Such domains are often represented by the non-manifold network of surfaces that partition the domain into labeled sub-domains. This network, known as the *material interface*, is widely used in applications including geometric processing, physical simulations, and manufacturing.

To be useful for applications, a material interface has to meet certain correctness criteria. Most importantly, the material interface should be *geometrically valid* (i.e., free of holes and

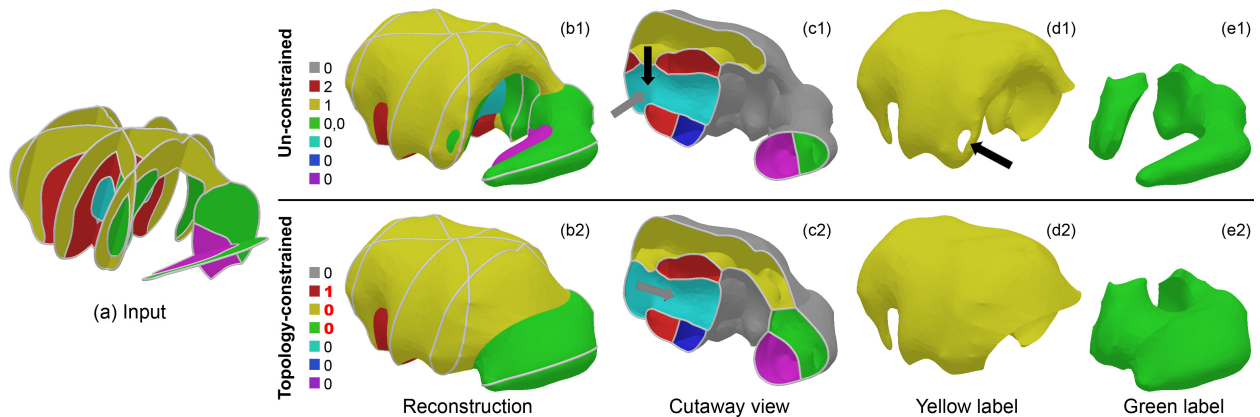


Figure 2.1: Given several multi-labeled planes depicting the anatomical regions of a mouse brain (a), reconstruction without topology control (b1) leads to redundant handles for the red and yellow labels (black arrows in c1, d1) and disconnection for the green label (e1). Our method (b2) allows the user to prescribe the topology such that the red label has one tunnel (gray arrow in c2), the yellow label has no tunnels (d2), and the green label is connected (e2). The legends in (b1,b2) report, for each label in the reconstruction, the genus of each surface component bounding that label (e.g., “0,0” means two surfaces each with genus 0). User-specified constraints are colored red.

intersections), so that it defines a proper partitioning of the domain into disjoint sub-domains. Furthermore, some applications are also sensitive to the *topology* of the surface. For example, fluid simulation within one or more sub-domains can be adversely affected if the surfaces bounding these sub-domains fail to have an expected number of connected components or genus. Extraneous components or genus can also be detrimental for many geometric processing tasks, such as mesh simplification and surface parameterization.

Topology control has been extensively studied in the context of modeling two-labeled domains, where the material interfaces are closed manifold surfaces. However, to date, no such control has been seen in modeling domains containing three or more labels in the absence of a template. Ensuring correct topology in a multi-labeled context is arguably more challenging, because the topology of different labels are intertwined: modifying the topology of one label may affect the topology of several other labels. The intertwining makes it difficult even for

humans to manually fix topological errors on a complex material interface without introducing geometric errors (e.g., the mouse brain in Figure 2.1 (b1-e1)).

In this paper, we present a novel algorithm for enforcing topological constraints when reconstructing multi-labeled material interfaces. Our algorithm is designed for inputs consisting of cross-sections of the subject. Such inputs often arise in biomedicine, where experts delineate boundaries of anatomical regions on 2D slices of a 3D medical image (e.g., MRI, CT). Cross-sectional inputs can also be found in other disciplines, such as material science (e.g., sectioned micrographs) and geology (e.g., seismic images). To model a multi-labeled domain, each cross-section contains a curve network that partitions the plane into labeled regions. Our method gives the user the option to specify the desired topology, in terms of number of connected components and genus, for any subset of the labels. The output is a geometrically valid material interface that interpolates the curve networks while meeting the topological requirements (Figure 2.1 (b2-e2)).

Our key contribution is a novel definition, called *interface sets*, that gives rise to not one, but a *space* of topology-varying material interfaces. Our definition mimics the level sets, which is a family of closed manifold surfaces defined by a scalar function and parameterized by a scalar value. Similarly, the interface sets are non-manifold surface networks defined by a vector function and parameterized by a vector value. We analyze the topological events in the multi-variate space of interface sets, which are much more complex than those in the univariate family of level sets, and propose a simple and effective method for sampling distinct topologies of interface sets.

Using the interface sets, we extend the recently introduced topology-controlled algorithm of Zou et al. [166] from two-labeled domains to multiple labels. Our algorithm proceeds in two stages. First, within each cell bounded by the cross-section planes, we define a suitable vector

function and enumerate interface sets with different topologies. Each topology is also given a score that measures its likelihood. Next, we perform combinatorial optimization to select one topology per cell so that the overall reconstruction satisfies the user-given topological constraints while the total score is maximized.

In addition to specifying components and genus, the user can steer the method in interactive ways. The user may browse and select from the list of topologies computed by our method for each cell. If a desired topology does not exist in our computed list, we offer a sketching interface whereby the user can easily create new topologies. These user inputs guide the algorithm towards a more satisfactory reconstruction. We demonstrated our algorithm and tool on both simple synthetic inputs and non-trivial biological data sets (e.g., Figures 2.1, 2.8, 2.9).

Contributions To the best of knowledge, our method is the first for material interface reconstruction that offers topology control without the use of any templates. Our main contributions are:

1. Defining a multi-variate space of material interfaces, analyzing its topological structure in the discrete setting, and developing a topology sampling method (Section 2.3).
2. Extending the topology-controlled reconstruction algorithm of Zou et al. [166] from two to multiple labels (Section 2.4).
3. Developing interactive tools for refining the surface topology (Section 2.5).

While our method is designed for cross-sectional inputs, we believe some of our contributions (particularly the method of interface sets) can benefit topology-aware modeling from other input types, such as point clouds or labeled medical images.

2.2 Related works

We briefly review the three bodies of work that are closed to ours, namely modeling multi-labeled domains, topology control in modeling two-labeled domains. And please refer to Sec 1.1 for a review of reconstruction from cross-sections.

2.2.1 Modeling multi-labeled domains

Typical representations of multi-labeled domains include (regional or global) implicit functions [163, 97, 164, 88, 60, 160, 103, 125] and volume fractions [31, 3, 10, 9]. While implicit function representations are often based on level sets, current works typically utilize a single level as the underlying function evolves (e.g., during a simulation), which creates a univariate family of domains. We are not aware of any work that explores a multi-variate space of material interfaces.

Many reconstruction methods are capable of creating geometrically valid material interfaces. The majority of these methods are based on iso-contouring [81, 20, 53, 162, 10, 69, 60, 123, 160], a few perform mesh surgeries [35, 46], and others further address the quality of elements (e.g., triangles and tetrahedra) using Delaunay meshing [118, 30, 51, 38, 59] or particle diffusion [101]. However, none of these methods offers explicit control over the topology. While topological errors can be avoided by fitting or evolving a template shape with the correct topology [151], these methods are limited to the availability of templates.

2.2.2 Topology-aware modeling of two-labeled domains

Numerous methods have been developed to fix topological errors on a closed manifold surface (see survey [11]). The vast majority of these methods are concerned with the removal of

redundant topological handles, while some also address connected components [108, 83]. Another class of methods directly reconstruct a topologically correct model from raw inputs, such as a point cloud [133, 134, 159], a collection of cross-section curves [166], or a grayscale volume [16, 161]. These methods are guided by prescribed genus [166, 133], interactive inputs [134, 159], or an existing template [16, 161].

It is non-trivial to obtain outputs with desirable topology in multi-labeled domain, and a naive application or extension of existing method might fail to generate a satisfactory result. For the simple input shown in Figure 2.2 (a), which consists of two parallel planes annotated with three labels (red, blue, outside), applying the method of Liu et al. [94] results in two components of the red label (Figure 2.2 (b), see cutaway). This would be an undesirable result if the user wishes to create a single component of the red label that tunnels through the blue label. A naive way to extend Zou’s method to handle multiple labels is by reconstructing each label independently with the desired topology and combining the reconstructed surfaces. This is exemplified in Figure 2.2 (c), where red and blue labels are reconstructed by Zou’s method respectively with genus 0 and 1. However, as seen in the cutaway view, the combination of two reconstructions results in jarring conflicts and intersections. In contrast, our extension of Zou’s work creates a geometrically valid material interface with the desired genus for both labels (Figure 2.2 (d)).

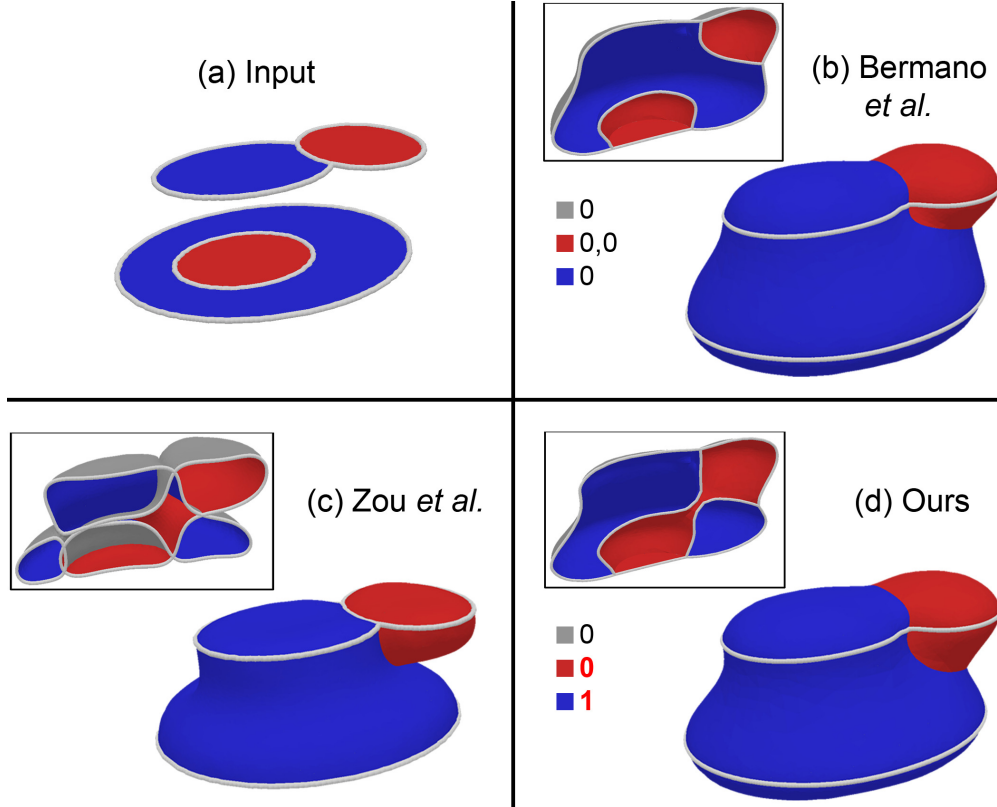


Figure 2.2: Comparing the topology-oblivious multi-labeled method of Bermano et al. [18] and Liu et al. [94] (b), the topology-constrained two-labeled method of Zou et al. [166] (c), and our topology-constrained multi-labeled method (d) on two cross-sections with three labels (a). Cutaway views are shown in inserts. Legends report the per-component genus for each label (red numbers are constrained).

2.3 Interface sets

Level sets have played fundamental roles in existing methods [133, 134, 166] to provide topology control in modeling two-labeled domains. These methods take advantage of several unique features of level sets. First, given a scalar function, any level set is guaranteed to be geometrically valid (i.e., a closed manifold). Second, the collection of all level sets is parameterized along a single “level” axis and can be easily explored. Third, the level sets have a rich topological variety, and extensive studies are available on the topological evolution of the level set with the level [102, 58].

To enable topology control in the context of multi-labeled modeling, we introduce a space of material interfaces that possesses similar features as level sets. Given a vector function, we define a space of *interface sets* such that each interface set is a geometrically valid material interface. The space is parameterized by a vector value (as opposed to a scalar level in level sets) and can be systematically explored. Lastly, this space reduces to the family of level sets in the special case of two labels, and it contains an even richer variety of non-manifold topologies in the case of three or more labels.

We start by defining interface sets and discussing their properties in the continuous setting (Section 3.1). Building upon classical works on level set topology, we then characterize the topological variations of interface sets in a discrete setting (Section 3.2). Finally, we propose a simple and effective scheme for sampling the large variety of interface set topologies (Section 3.3). In the next section, the sampling scheme will be utilized in our reconstruction algorithm to produce candidate local topologies from cross-sectional inputs.

2.3.1 Definition and properties

Our definition builds on an existing implicit definition of material interfaces, which has been used by various researchers [97, 60, 160]. In this definition, a n -labeled domain is represented by a vector-valued function $\vec{f}(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \dots, f_n(\vec{x})\}$, where \vec{x} is a point in d -dimensional space and each f_i is a continuous scalar function. Intuitively, $f_i(\vec{x})$ describes the “prominence” of the i -th label at \vec{x} . Each point is then assigned the *most prominent* label(s), that is,

$$\text{Labels}(\vec{x}) = \arg \max_{i=1, \dots, n} f_i(\vec{x}). \quad (2.1)$$

The material interface consists of all points whose label assignment is not unique (i.e., two or more labels share the greatest prominence). This material interface is guaranteed to be geometrically valid, as it divides the space into regions with unique labels.

To be able to define not just one, but a parameterized set of material interfaces, we introduce an *offset* vector $\vec{c} = \{c_1, \dots, c_n\}$ to the definition described above. This offset vector plays the role of the “level” in defining the level sets in a scalar function. More precisely, \vec{c} is *added* to the vector function \vec{f} before evaluating the labels. That is, the labeling is now *parameterized* by \vec{c} as

$$Labels(\vec{c}, \vec{x}) = \arg \max_{i=1, \dots, n} (f_i(\vec{x}) + c_i). \quad (2.2)$$

The *interface set at offset* \vec{c} consists of all points \vec{x} whose label assignment is not unique, that is, $|Labels(\vec{c}, \vec{x})| \neq 1$.

We can visualize interface sets in $d = 2$ dimensions intuitively as superimposed terrain maps (Figure 2.3). Imagine that each f_i is the height map of a 3D terrain over the 2-dimensional domain, and that each terrain has a unique color (Figure 2.3 (a)). Given an offset vector \vec{c} , we shift each i -th terrain vertically by the amount c_i and superimpose the shifted terrains (Figure 2.3 (b,c,d) middle). The labeling function $Labels(\vec{c}, \vec{x})$ is precisely the picture of the superimposed terrains taken from above, and the interface set is where two or more terrains meet in this picture (Figure 2.3 (b,c,d) bottom).

It is easy to see that every interface set is a geometrically valid material interface, since it divides the domain into regions carrying unique labels (i.e., $Labels(\vec{c}, \vec{x})$). In particular, the interface set at the zero offset $\vec{c} = 0$ is the material interface defined in the first paragraph and used in previous works.

We can also show that interface sets reduce to level sets in the case of $n = 2$ labels. In this case, any level set of a scalar function can be reproduced by some interface set of a vector function, and vice versa. Specifically, the level set of a scalar function f at any level c is identical to the interface set of the vector-valued function $\vec{f}(\vec{x}) = \{f(\vec{x}), 0\}$ at offset $\vec{c} = \{0, c\}$. Conversely, the interface set of a vector-valued function $\vec{f}(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x})\}$ at any offset $\vec{c} = \{c_1, c_2\}$ is the same as the level set of the scalar function $f(\vec{x}) = f_1(\vec{x}) - f_2(\vec{x})$ at level $c = c_2 - c_1$.

Note that an interface set is different from the intersection of n level sets, each defined by a scalar function f_i and a level c_i , as studied in multivariate topological data analysis [57, 40]. The interface set is generally a $(d - 1)$ -dimensional complex, regardless of the number of labels n , since it partitions the d -dimensional space into labelled regions. In contrast, the intersection of n level sets has a dimensionality of $d - n$, which is lower than that of the interface set and decreases as the number of labels increases. In the case of $n = 2$ labels in $d = 3$ dimensions, the intersection of level sets consists of one-dimensional curves, which are known as *fibers* and have found uses in visualization of bivariate data [41, 144].

2.3.2 Discrete topological variations

Topological evolution of the level set, as the level changes, is well-understood [102, 58]. Topological changes are marked by local *topological events*, such as merging, splitting, and destruction or creation of components. These events take place at well-defined locations, known as *critical points*, which are identified by vanishing gradient of the scalar function. The function values at these locations, known as *critical values*, divide the range of levels into one-dimensional intervals, such that the level sets within one interval all share a common topology.

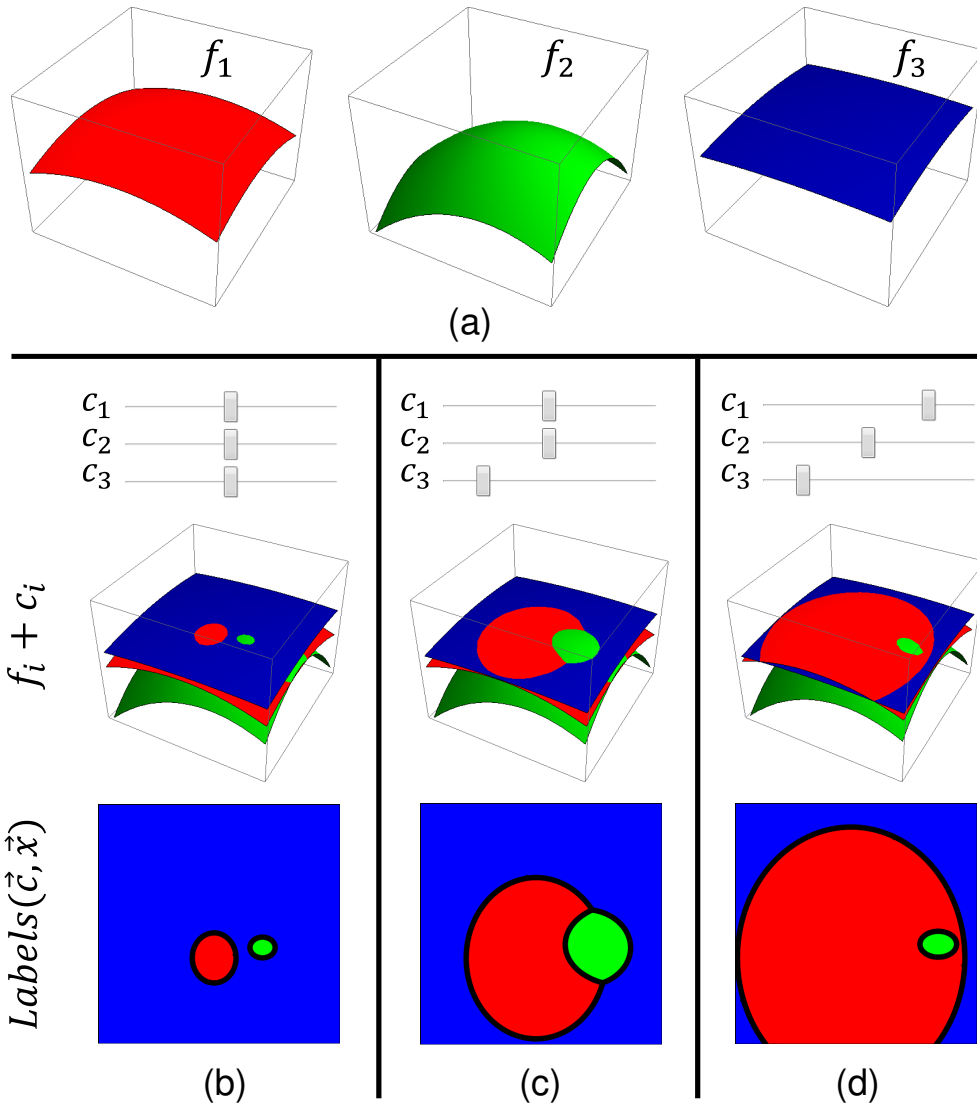


Figure 2.3: Interface sets in 2D: (a) The input vector function $\vec{f} = \{f_1, f_2, f_3\}$, visualized as three height maps. (b,c,d): Three different choices of offsets $\vec{c} = \{c_1, c_2, c_3\}$ (top), superimposed height maps $f_i + c_i$ (middle), and the labeling (as color) and interface sets (as black curves) in the 2D domain (bottom).

In contrast, topological evolution of the interface set, as the offset vector changes, is far more complex. First, there is a greater variety of topological events that involve non-manifold features of the surface (e.g., junction curves and points where more than three sheets meet). Second, the multi-variate nature of our “level” parameter - the offset vector - creates a complex topological landscape in the space of interface sets. Let’s consider the n -dimensional

space of all offset vectors, which we call the *offset space*. This space is made up of disjoint n -dimensional regions (as opposed to one-dimensional intervals in the case of level sets) such that interface sets within one region all share a common topology. We call such regions *topology pockets*, or *pockets* in short. Topological events take place when the offset vector moves from one pocket to an adjacent pocket in the offset space. We call the offset vectors that lie on the boundary of pockets the *critical offsets*. Unlike the critical values in a scalar function, critical offsets in a vector function form continuous, $(n - 1)$ -dimensional complex in the offset space, which we call the *critical complex*¹.

Characterizing the critical offsets is key to understanding the topological evolution of the interface set. However, providing a complete, continuous characterization has proven to be a non-trivial task. As our goal is to develop algorithms for practical, discrete inputs, we perform our analysis in a discrete setting and leave the continuous characterization as a venue for future investigation.

Discretization

We consider the discrete input as a simplicial complex C in \mathbb{R}^d . Each vertex (i.e., 0-cell) v of C is associated with a vector $\vec{f}_v = \{f_{v,1}, \dots, f_{v,n}\}$. A common way to construct a function is by piecewise linear (PL) interpolation within each cell of C . Such interpolation is particularly suited for analyzing level set topologies [58], since topological events of level sets are restricted to the vertices of C . However, we have observed that the topological events of interface sets in a PL vector function are no longer restricted to vertices of C . In fact, these events can take place in arbitrary locations in the domain. An example is shown in Figure 2.4 (a) for

¹The effective dimension of offset space is $n - 1$, since the interface set only depends on the relative difference between components of the offset vector. Similarly, the critical complex is an “extrusion” of a $(n - 2)$ -dimensional complex (Figure 2.5(c)).

a 4-labeled domain in 2D; note that the yellow label disappears inside a triangle after the offset changes.

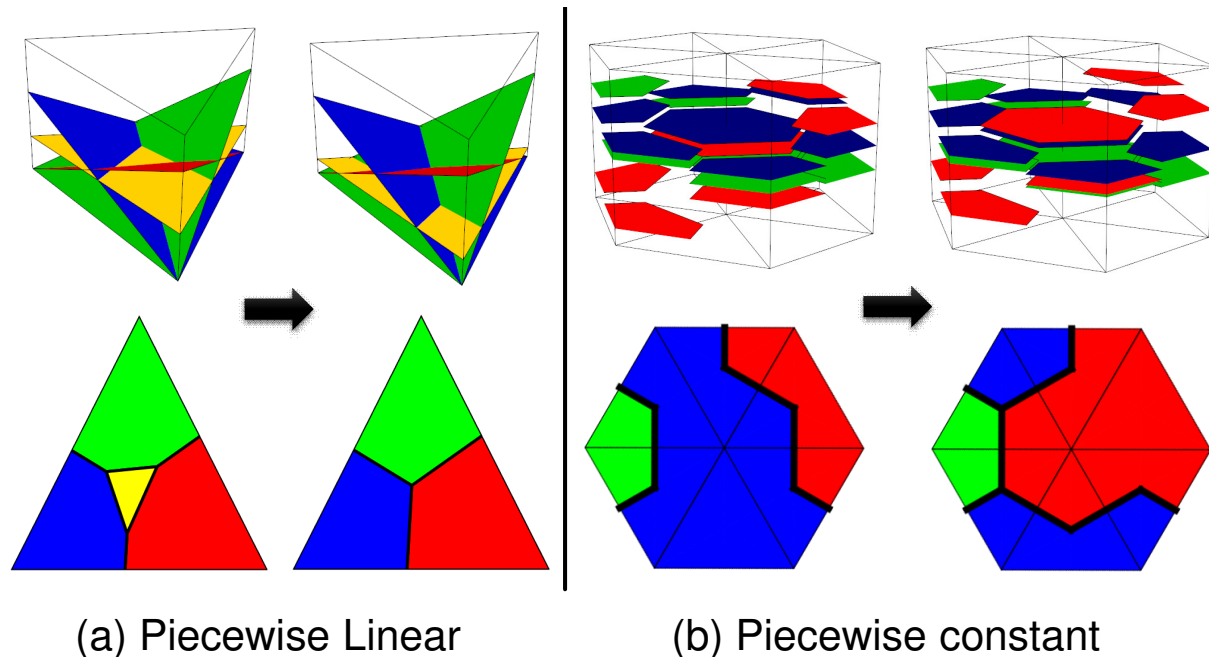


Figure 2.4: (a) A topological change of interface sets in PL interpolation can take place in arbitrary locations: lowering the offset of the yellow label causes the yellow region to disappear inside the triangle. (b): A topological change in our PC interpolation is restricted to the dual of the input complex. The offsetted functions are shown at the top and the interface sets are shown at the bottom.

To make the analysis of topological events simpler, we opt for a piecewise constant (PC) interpolation. We consider the dual complex of C , noted as C^* , which consists of k -cells that are dual to $(d - k)$ -cells of C for $k = 0, \dots, d$. The vertices of C^* lie at the barycenters of their dual d -cells in C (although the exact locations do not affect the topology analysis). We define the vector function \vec{f} so that $\vec{f}(\vec{x}) = \vec{f}_v$ for any point \vec{x} in the interior of a d -cell in C^* that is dual to a primary vertex v . Since \vec{f} is discontinuous, we adjust the definition of the interface sets as *the union of all cells of C^* that are faces of two d -cells of C^* with different labels*. Examples of such interface sets are shown in Figure 2.4 (b). In contrast to

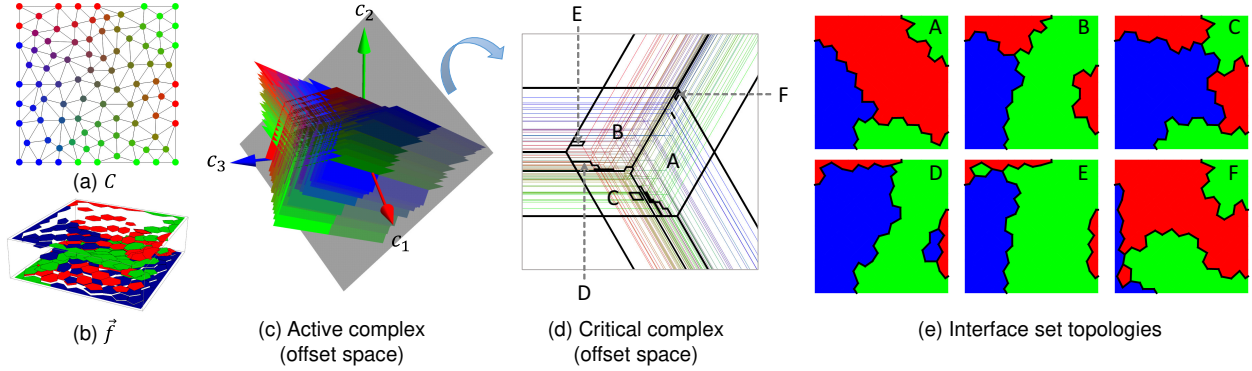


Figure 2.5: Topology analysis of 3-labeled interface sets. (a) A complex C where each vertex is colored by the associated 3-vector. (b) The piecewise constant vector function shown as one height map for each label. (c) The active complex in the 3D offset space consisting of one triple half-plane (“triblade”) for each vertex of C . (d) A cutaway of the offset space (by the gray plane in (c)) showing the active complex (colored lines) and the critical complex (black lines). (e) Example interface sets in different pockets (see pocket labeling in (d)).

PL interpolation, interface sets in PC interpolation are restricted to low-dimensional cells of C^* , which allows for simpler characterization of critical offsets (see below).

Critical offsets

In our PC interpolation, the interface set changes *only* when a vertex of complex C alters its label. We call the offsets that trigger these vertex label-changing events the *active* offsets. An active offset is critical if label-changing causes the topology of the interface set to change as well. We will first characterize the active offsets and then identify the subset that is critical.

Given a vertex v and any pair of labels $i, j \in \{1, \dots, n\}$, there is a collection of offsets \vec{c} at which v may switch its label between i and j . This collection is defined by a set of equality and inequality constraints:

$$\{\vec{c} \mid f_{v,i} + c_i = f_{v,j} + c_j > f_{v,k} + c_k, \forall k \neq i, j\} \quad (2.3)$$

Geometrically, this collection forms a bounded $(n - 1)$ -dimensional hyperplane in the n -dimensional offset space. There are C_n^2 such hyperplanes for each vertex v , and they together partition the offset space into n symmetric regions corresponding to the n possible labelling of v . Combining the hyperplanes over all vertices forms a piecewise linear complex in the offset space is the union of all active offsets. We call this complex the *active complex*.

To see this visually, take $n = 3$. Equation 2.3 defines a 2D half-plane in the 3D offset space. For each vertex v , there are $C_3^2 = 3$ such half-planes, one for each pairing of labels. The three half-planes share a common boundary line in the direction of $\{1, 1, 1\}$ passing through the point $\{-f_{v,1}, -f_{v,2}, -f_{v,3}\}$. Visually, they form a “triblade” around the line. The triblades associated with all vertices intersect to form a honeycomb-shaped active complex (Figure 2.5 (c) and cutaway in (d)).

Since our reconstruction algorithm is concerned with the topology of individual labels, we define a topological event in this paper as when there is a change in either the number of connected components or genus of the surfaces that bound a particular label. To this end, we can use the same criteria for critical points in a PL scalar function [58] to identify topological events in any given label. Specifically, recall that the *star* of a cell σ in a complex consists of all cells that contain σ as a face, and the *link* consists of all faces of cells in the star that are disjoint from σ . Given a labeling of the vertices by an offset vector, we define the *i-link* of vertex v as the union of cells in v 's link that contain only vertices with label i . Switching the label of v between i and another label triggers a topological change of label i if the i -link of v is not contractible to a single point.

As an example, we use the criteria to analyze the label-changing event in Figure 2.4 (b). Since the label of the center vertex changes from blue to red, we focus on the blue-link and the red-link of that vertex. The former consists of one edge (at the bottom) and a vertex (at

top-left), which is not contractible, while the latter consists of a single edge (at top-right), which is contractible. Hence the blue label experiences a topological change (a splitting).

An active offset \vec{c} satisfying Equation 2.3 is critical if *either the i -link or j -link of v is not contractible*. Geometrically, the critical complex, made up of the union of all critical offsets, forms a sub-complex of the active complex. As the active complex is piecewise linear, so is the critical complex, which divides the offset space into polyhedral pockets. An example of the critical complex for $n = 3$ labels is shown in Figure 2.5 (d) on a cross-section of the offset space, and (e) shows topologically distinct interface sets in different pockets.

Note that the per-label topological events that we detect do not cover all possible ways in which the topology of the interface set can change. For example, as we will show in Section 2.7, the connectivity of the non-manifold junction curves can change without altering the topology of any individual label. It would be interesting in the future to define other types of topological events, which would lead to a more refined critical complex. This would potentially allow a reconstruction algorithm to have finer control over the non-manifold topology of the material interface.

Topology sampling

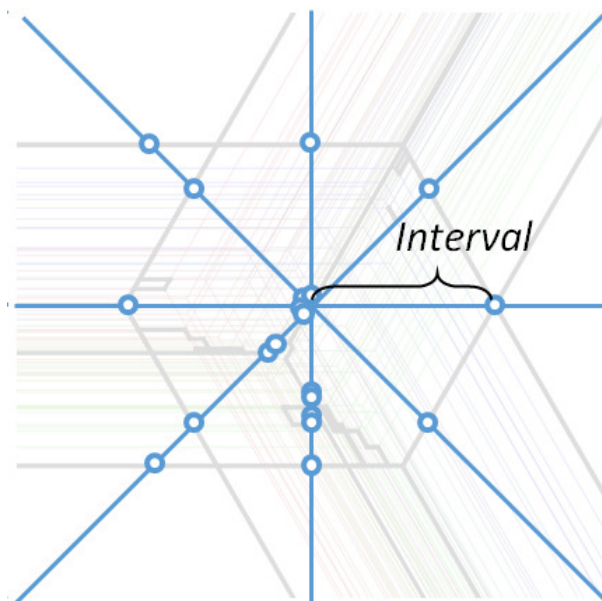
A direct way to enumerate distinct topologies of interface sets is to explicitly construct the critical complex in the offset space. However, this can be computationally expensive. Our calculations, confirmed by experiments, show that the active complex has a complexity of $O(V^{n-1})$ where V is the number of vertices in the input domain C and n is the number of labels. A brute-force algorithm that first constructs the active complex and then prunes non-critical parts would be impractical.

To tame the complexity, we opt for an approximate, sampling-based approach. The motivating observation is that the more transient topologies tend to correspond to smaller pockets in the offset space. For example, tiny pockets (D,E,F) in the offset space of Figure 2.5 (d) correspond to topologies that contain small isolated components, as shown in (e). We therefore argue that a regular sampling scheme in the offset space would have a greater chance of finding the more stable topologies, because they are more likely to be captured by larger pockets.

A naive point sampling scheme is to use regular lattice points in the offset space. However, to form a good coverage, more than a few points would be needed for each dimension of the space, and the total number of samples can still be significant as the dimension of the offset space (i.e., the number of labels) grows.

To reduce the sample count without sacrificing the coverage, we use 1-dimensional *rays* as samples. By intersecting a ray with the critical complex, we can compute *intervals* along each ray within which the interface sets share a common topology, much in the same way as how the level set topologies are enumerated. Unlike point samples, each ray effectively represents an infinite number of point samples (in one direction), and hence a relatively small number of rays are needed.

As our reconstruction algorithm is mostly interested in material interfaces at offsets close to zero, we shoot rays in a radial pattern from the origin of the offset space (see insert). Accounting for the one redundant dimension of the offset space, the rays all lie in hyperplane orthogonal to the 1-vector $\{1, \dots, 1\}$. To create a pseudo-uniform distribution, we form rays



connecting the origin with points on a regular lattice in that hyperplane centered at the origin with $(2b + 1)$ points on each of its side, where b is a user-specified small integer. This creates up to $(2b + 1)^{n-1}$ rays. In our tests, we found that $b = 1$ offers a reasonable sampling of topologies while keeping the overall execution time low even for large n (e.g., 6-8).

Intersecting a ray with the critical complex can be implemented easily and executed efficiently (i.e., in polynomial time). We start by computing the intersection between a parametric equation of the ray and a hyperplane of the active complex defined by Equation 2.3, which involves solving a linear equation with one variable and checking the solution against a set of linear inequalities. The intersecting offset, if found, is further checked for criticality using the link-based criteria described earlier, which operates in the local neighborhood of a vertex in the input complex C . Repeating these computations for all hyperplanes and sorting the resulting critical offsets produces the desired intervals. The total complexity of the algorithm is $O(H \log H + L * n * H)$ where L is the maximum number of cells in the link of a vertex and $H = VC_n^2$ is the number of hyperplanes. For a fixed dimensionality of the input (3 in our case), this complexity is polynomial in both the number of vertices (V) as well as in the number of labels (n).

2.4 Reconstruction algorithm

We now describe our algorithm for reconstructing material interfaces from cross-section inputs. The input to our method consists of a collection of possibly non-parallel planes in 3D, each partitioned into labeled regions by a network of curves. Without additional input,

our algorithm produces a geometrically valid material interface that interpolates the curve networks.

The user have the option to specify the desired topology of the subject. For complex subjects made up of many labels (e.g., Figures 2.1,2.8,2.9), the user may not have the knowledge of the precise topology of all labels. Also, the topology of some labels may not matter in downstream applications. To be able to handle these practical situations, we let the user choose *any (possibly empty) subset* of the labels whose topology need to be constrained. For each constrained label, the user specifies the desired number of connected surface components that bound that label as well as the genus for each component. Note that a connected 3D region with an interior cavity (“bubble”) counts as two separate surface components.

Our algorithm adopts the classical divide-and-conquer paradigm for cross-section-based reconstruction. We consider the partitioning of the 3D space into convex polyhedral cells by the input planes (known as the *arrangement* in computational geometry). The reconstruction problem is reduced to creating a surface within each cell that interpolates the curves on the boundary of the cell. What differentiates our algorithm from the majority of existing methods is that we create not one, but a collection of surfaces within each cell that differ in topology. These surfaces give rise to a space of topologically different overall reconstructions, among which one that matches the user-specified topology is chosen. Our method proceeds in two stages, which are illustrated on a simple example in Figure 2.6:

1. **Enumeration:** For each cell of the arrangement, compute a set of topologically distinct material interfaces that all interpolate the curve network on the cell’s boundary. Assign a score to each material interface that measures its likelihood. (Figure 2.6 (b), Section 2.4.1)

2. **Selection:** Select one material interface per cell so that the overall reconstruction matches the user-given topology constraints while the sum of the scores is maximized. (Figure 2.6 (c), Section 2.4.2)

Our method generalizes the same two-stage framework of Zou et al. [166] from closed, manifold surfaces to multi-labeled material interfaces. Besides using the newly developed interface sets for topology enumeration (Section 2.3), we made several extensions in their framework to address the challenges associated with multiple labels. In the enumeration stage, Zou designed a scalar *indicator* function whose level sets interpolate the curve loops on the cell’s boundary. We extend it to a vector function whose interface sets interpolate the boundary curve networks. In the selection stage, Zou uses a region-growing dynamic programming algorithm, which we extend to simultaneously track the topology of multiple labels.

We next detail the two stages while highlighting our extensions over Zou’s work. The result of these two stages is a topologically correct reconstruction made up of interface sets within the arrangement cells. Since our interface sets are defined on the dual of a tetrahedral complex, they have jagged appearances. We improve the geometry of the reconstruction in a post-process using the method in [94] which creates a refined and fair material interface that still interpolates the input curve networks.

2.4.1 Enumeration

We consider a polyhedral cell Ω in the arrangement whose boundary $\partial\Omega$ is partitioned by a curve network U into regions with up to n labels. To enumerate material interfaces within Ω , we define a vector function \vec{f} over Ω and use the techniques developed in the previous section to sample topologically distinct interface sets of \vec{f} .

The function \vec{f} , as well as the range of the offsets, need to be carefully chosen so that the interface sets interpolate U . In addition, the interface sets should depict a natural extension of U into the interior of Ω . In the special case of two labels (outside/inside), Zou et al. [166] defines a *harmonic* indicator function that evaluates to 1 (resp. 0) at any point on $\partial\Omega$ that is labeled outside (resp. inside). Harmonic function offers a natural interpolation of the boundary values. This particular function also has the desirable property that any level set at a level in the range $(0, 1)$ interpolates U on $\partial\Omega$.

Extending Zou’s scalar function to $n > 2$ labels, we define a vector indicator function $\vec{f} = \{f_1, \dots, f_n\}$ such that each f_i is a harmonic function and, for any point \vec{x} on $\partial\Omega$, $f_i(\vec{x}) = 1$ if \vec{x} has label i and $f_i(\vec{x}) = 0$ otherwise. It is easy to verify that the interface set of \vec{f} interpolates the curve network U at an offset $\vec{c} = \{c_1, \dots, c_n\}$ where each c_i lies in the range $[0, 1)$. Figure 2.5 (a) is an example of such a function in a triangulated 2D cell. As seen in Figure 2.5 (e), interface sets at different offset vectors within the $[0, 1)$ range (which projects to the center hexagonal region in Figure 2.5 (d)) touch the cell’s boundary at the same locations.

We compute \vec{f} and enumerate its interface sets on a tetrahedralization of Ω . To ensure consistence among neighboring cells, tetrahedral meshing is performed once over the entire 3D domain, constrained by the planes as well as vertices and edges of the curve networks (we use Tetgen [137]). The harmonic functions are computed on the edge graph of the tetrahedral mesh inside Ω , as in [166], which results in a vector \vec{f}_v associated with each vertex v . We then invoke the ray-sampling algorithm to compute intervals of offsets within the range $[0, 1)$. For each interval that does not contain the zero offset, we extract the interface set at the offset vector in the midpoint of that interval.

We also extend the scoring method in [166] to assess the likelihood of an interface set. The key idea is to treat each harmonic function f_i as the probability distribution of label i . Given an offset vector \vec{c} , which gives rise to the labeling $Labels(\vec{c}, \vec{x})$ for any point $\vec{x} \in \Omega$, we consider the joint probability of all labeled points,

$$h(\vec{c}) = \sum_v w(v) \log(f_{v, Labels(\vec{c}, v)}) \quad (2.4)$$

where the summation is over all interior vertices v in the tetrahedralization of Ω , and $w(v)$ measures the total volume of the tetrahedra incident on v .

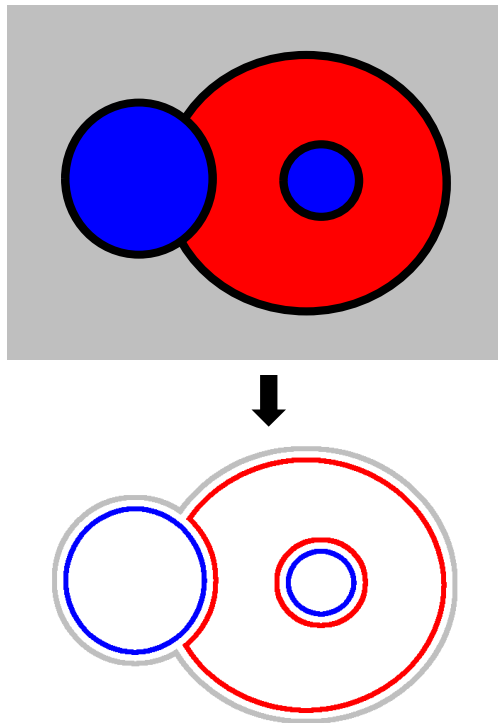
Note that many interface sets may have the same topology. This can be caused by the same pocket in the offset space being sampled by multiple rays or even multiple times by the same ray. Also, different pockets may correspond to the same interface set topology. For each distinct topology (in terms of the number of connected components and genus for each label), we keep only the interface set with the highest score and remove the rest.

2.4.2 Selection

We first briefly review the combinatorial optimization method of Zou et al. [166]. In the context of two-labeled modeling, their algorithm aims to find a closed surface with a user-specified genus from enumerated topologies within each cell. The algorithm is *optimal*, in that the output is guaranteed to have the highest total score among all possible combinations of cell topologies that match the target genus. The basic idea is to grow a *known volume* (KV), which is a union of a subset of the cells, while keeping track of the top-scored solution (i.e., a choice of topology per cell) for each possible surface topology within the KV. The KV is grown by merging with one adjacent cell at a time. Each merging computes the solutions of the new KV from those in the old KV as well as the enumerated topologies in the merged

cell. When the KV is grown to the entire domain, the algorithm outputs the top-scored solution that matches the target genus.

We present a simple extension of the algorithm to handle multiple labels. In a nutshell, we treat the problem of creating a n -labeled material interface as creating n overlapping closed surfaces. We encode the curve network as a set of overlapping closed loops, in which each input curve segment is duplicated (see insert). Accordingly, we encode the topology of an interface set in a cell as the topology of a collection of manifold surfaces whose boundaries are these loops. Feeding this representation into Zou’s algorithm allows simultaneous tracking of the topology of all labels as the KV is grown.



Although optimal, the algorithm can have a high complexity due to the possibly large number of topologies being tracked, which may grow significantly with the number of labels. We adopt three strategies to curb the space of topologies. The first two strategies follow those in [166], while the last one is unique to our multi-labeled context. First, we remove any interface set from the enumeration stage if its topology is deemed too complex, such as containing some surface with non-zero genus in the cell. Second, we remove any intermediate solutions during KV growing that already have higher genus or number of components than the given topology constraints. Third, if the user only constrains a subset of the labels, we only keep intermediate solutions that differ in the topology of those constrained labels. This last strategy effectively makes the complexity of the algorithm depend only on the number of constrained labels.

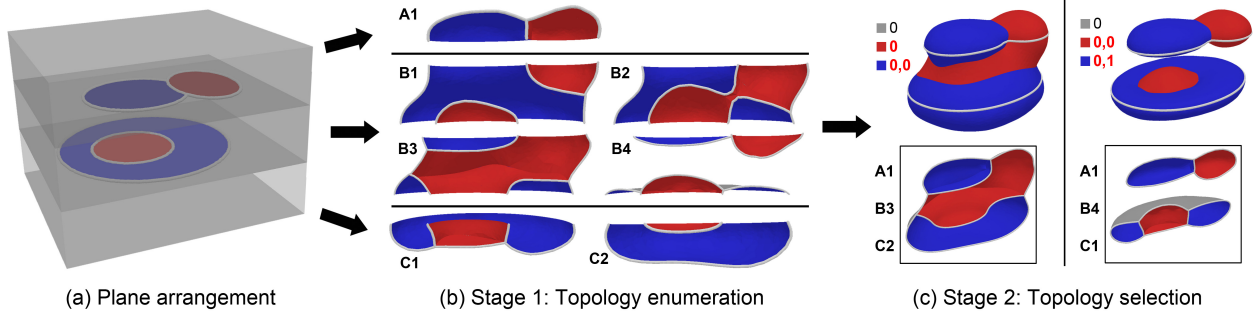


Figure 2.6: Reconstruction algorithm: starting from the plane arrangement (a, showing 3 cells divided by the two cross-section planes), our algorithm first enumerates and scores topologies of interface sets within each cell (b, interface sets in each cell are ordered by decreasing scores), then one topology is selected per cell to achieve the topological constraints while maximizing the total score (c, showing solutions under two sets of constraints marked in red; letters by the cutaway views at the bottom are choices of cell topologies).

2.5 User interaction

Besides specifying topological constraints, we offer two ways for a user to interact with our algorithm and refine the solution. First, our method produces a ranked list of possible topologies within each cell (see Section 2.4.1). The user can browse through the list and pick any favorable topology. The user-selected topology will be treated as hard constraint during optimization. This interaction can be useful when the algorithm creates a solution that meets the topological constraints but exhibits undesirable local connectivity. For example, given the input in Figure 2.7 (a), the automatic solution under genus-0 constraint for the red label places the branching of the red label in the upper cell (c). The user decides that the branching should take place in the lower cell, and she selects the corresponding topology from our ranked list in that cell (d, left). Incorporating this information, our algorithm then produces another genus-0 solution with the desired branching location (d, right).

To deal with the case that the desired topology is not found in the computed list, we developed a sketching tool whereby the user can create arbitrarily complex topologies within a cell. As

shown in Figure 2.7 (e, left), the user is presented with a cutaway view of the cell on a plane that she can manipulate, and she can scribble on that plane where a particular label should be present. The labeled scribbles are incorporated by our algorithm to update the vector function in that cell, by treating the scribble points as fixed label constraints (similar to points on the cell boundary). A new set of interface set topologies are then enumerated (the top-scored one is shown in (e, middle)) and used for optimization (e, right). Sketching can be particularly useful when our algorithm fails to find a solution satisfying the topological constraints, due to the limited set of topologies explored by the algorithm.

2.6 Results

We test our algorithm and tool on several non-trivial biological examples. These examples contain a large number of labels (6 or more) that interact with each other in complex ways. Two of the examples (mouse brain and liver) are also demonstrated in the accompanying video.

In the mouse brain example in Figure 2.1, reconstruction without any topology constraints leads to errors (e.g., extraneous components and tunnels) in 3 of the 7 labels (b1-e1). Constraining the topology of these labels results in a satisfactory solution and no new topological errors were introduced to the un-constrained labels (b2-e2).

A more complex scenario is shown on a liver example in Figure 2.8. Unconstrained reconstruction produces several obvious errors, including two extra components for the green label and an extra tunnel for the outside label (see arrows in (b)). Based on prior knowledge, we also know that the turquoise label should form a “shell” that wraps around the blue and purple labels. In the unconstrained reconstruction, however, both blue and red labels are exposed to the outside, and together they create a tunnel for the turquoise label (see the dotted line in

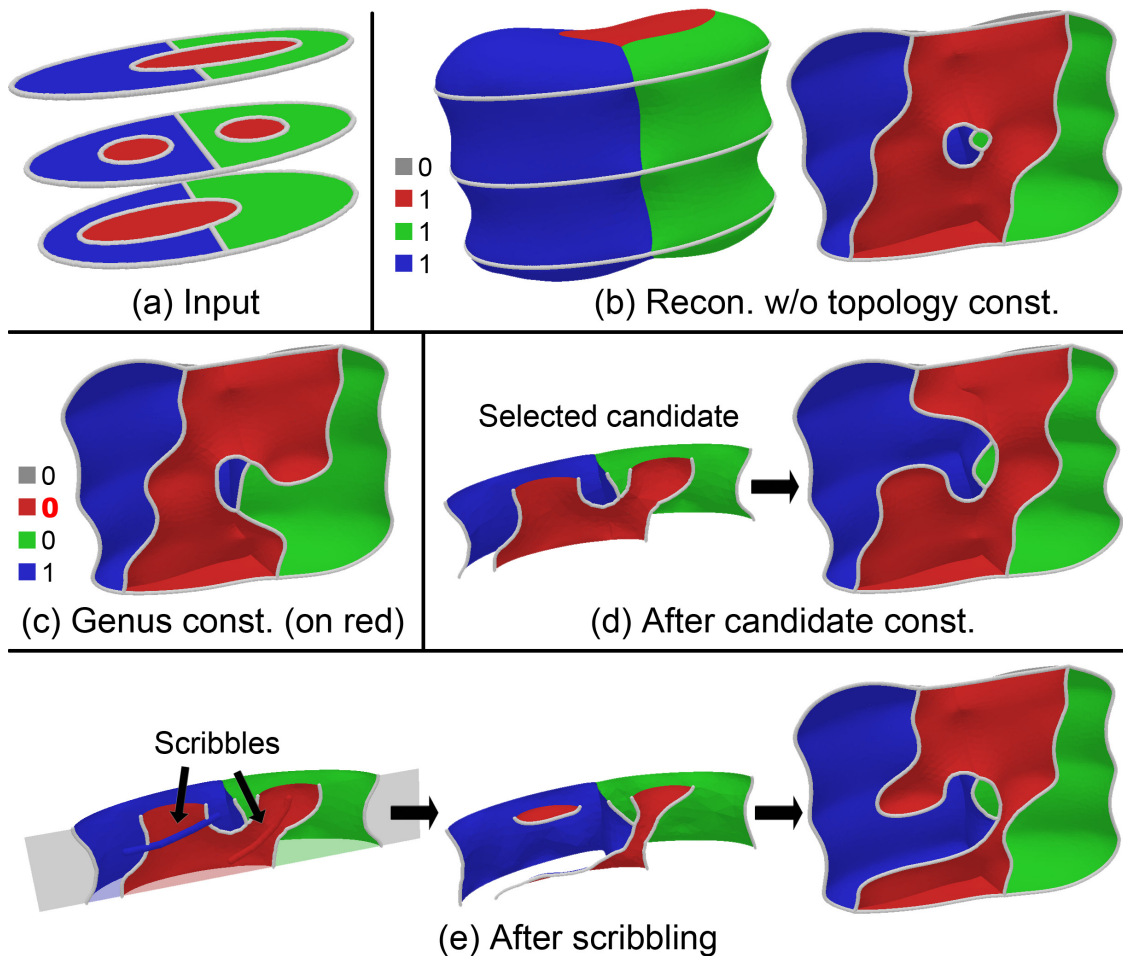


Figure 2.7: User interactions: a 4-labeled input (a) and reconstructions without topological constraints (b), with genus-0 constraint on the red label (c), after the user picks a different topology in one of the cells (d), and after the user adds scribbles in that cell (e). All reconstructions are shown in cutaway views as their exterior shapes are similar to that in (b).

the insert of (b)). After running our algorithm with topology constraints on green, turquoise and outside labels, a new error occurs for the unconstrained blue label - it breaks into two components (see arrows in the insert of (c)). A topologically correct reconstruction is created after adding constraints for both blue and red labels (d). Finally, we added scribbles in two cells to create a more natural branching structure for the green label while keeping the same topological constraints (e) (the same effect can be achieved if we select an alternative topology in one of the cells and scribble in the other cell; see the accompanying video).

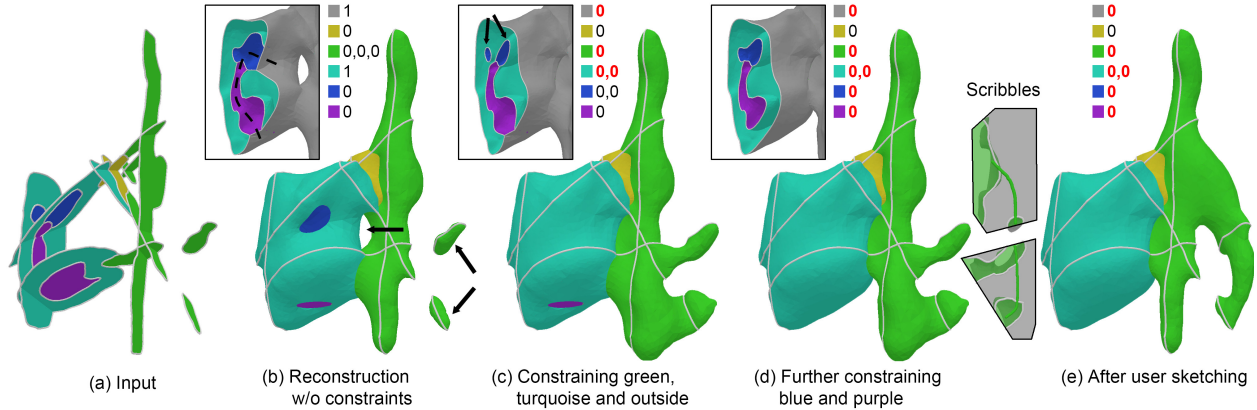


Figure 2.8: The Liver data set (a) and reconstructions without topological constraints (b), with some (c) and more (d) constraints, and after applying scribbles (e). Cutaway views are shown in the inserts, and the legends report the per-component genus for each label (constrained genus are in red). Arrows point to topological issues, where the solution does not meet users expectation. See detailed explanations in Section 2.5.

Lastly, we demonstrate our algorithm on a chicken heart data set made up of 13 parallel slices containing 8 labels. As shown in Figure 2.9 (a), this input is particularly challenging as some labels (e.g., light-green) weave through others. Without topological constraints, the reconstruction contains numerous errors, which are all resolved after adding constraints on 5 labels. Figure 2.9 (d) examines two of these labels, showing the removal of an extra tunnel for the orange label (d1-d3) and the connection of two components for the light-green label (d4-d6) as a result of adding the constraints.

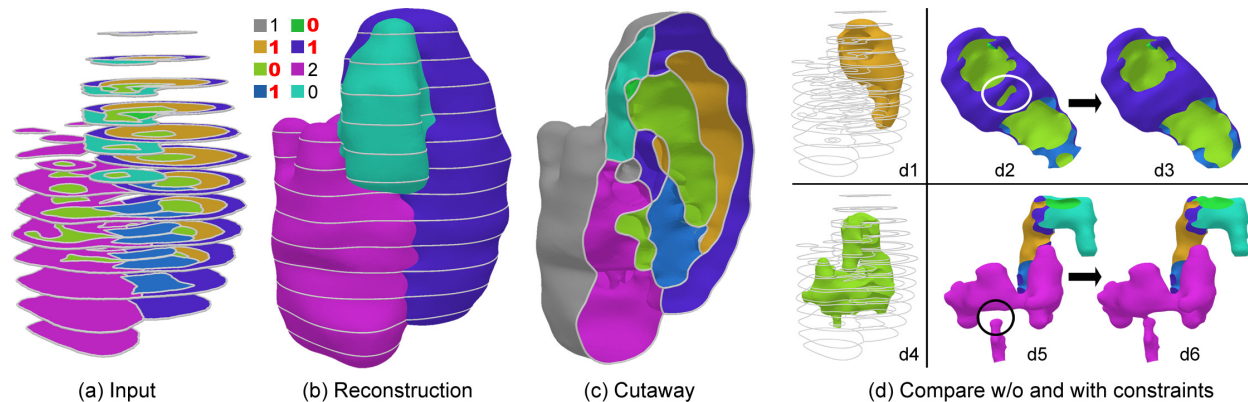


Figure 2.9: The chicken heart data set (a) and reconstruction with topological constraints on 5 labels (b, cutaway view in c). (d) compares the reconstruction of orange (top) and light-green (bottom) labels without and with topological constraints. The surfaces in (d2,d3,d5,d6) are colored by labels of adjacent sub-domains to reveal the intertwining of labels.

2.6.1 Performance

The performance of our method depends on many aspects of the input. Besides the number of labels, slices and constraints, the amount of topological ambiguity (as manifested by the number of enumerated topologies) within each cell can also significantly affect the performance. The examples in the paper (Figures 2.1, 2.8, 2.9) exhibit a range of characteristics along these axes, as shown in Table 2.1.

Table 2.1 reports the timings of the two stages of our algorithm on these examples. Our tool was implemented in C++ and run on a MacBook Pro with 2.5 GHz Intel Core i7 and 16GB RAM. The first stage, topology enumeration, is by far the most time-consuming stage. In practice, we run this stage only once and invoke the selection stage repeatedly to optimize for different topological constraints. The only exception is that, after a sketching interaction is performed in one of the cells, the enumeration needs to be re-run for that cell. Timing of the enumeration stage is in fact dominated not by ray-shooting, but by processing the

intervals returned by ray-shooting (e.g., extracting interface sets, computing their genus and connected components, and scoring). Timing of the selection stage is sensitive to the number of topologies enumerated in each cell.

	#Slices	#Labels (const.)	#Tets.	Max #Topo. per cell	Stage 1 time	Stage 2 time
Fig 2.1	6	7(3)	91793	8	259s	11ms
Fig 2.8	5	6(5)	69819	18	212s	83ms
Fig 2.9	13	8(5)	188401	89	2702s	56s

Table 2.1: Running time of the two stages of our algorithm on the mouse brain (Fig 2.1), liver (Fig 2.8), and chicken heart (Fig 2.9). Also showing the number of constrained labels, number of tetrahedra, and maximum number of per-cell topologies.

2.7 Conclusion and discussion

We introduce an algorithm for reconstructing multi-labeled material interfaces that allows the user to explicitly prescribe the topology of individual labels. Our key contribution is defining a novel space of material interfaces (as interface sets) that has a rich variety of topologies and allows for systematic exploration. Combined with interactive tools, our method was shown to be effective on non-trivial real-world data in the form of cross-sectional slices.

2.7.1 Limitations

Our method uses a number of approximating schemes to tame the complexity of topology enumeration, including using piecewise constant interpolation and ray-sampling. As a result, some topologies could be missed. We would like to explore the practicality of a complete construction of the pockets in the offset space in the piecewise linear setting, perhaps limited to a small region around the origin of the offset space (i.e., the zero offset vector). Extensions

of topology filtering methods [68, 63] from scalar functions to vector functions could also be useful for removing small pockets prior to the construction, therefore improving the efficiency.

While our current work focuses on optimizing topology, the geometry of our reconstruction can be further improved in a number of ways. As in [166], if an underlying image volume is available, one can create a reconstruction that is aligned to intensity edges in the volume by replacing the harmonic function in each cell with image-based random-walk probabilities. Some other ideas include using more sophisticated scoring function of interface set topologies that favor smoother geometry, and higher-order harmonic functions to improve the continuity of surface across cell boundaries.

Chapter 3

Repairing Inconsistent Curve

Networks on Non-parallel

Cross-sections

3.1 Introduction

One of the primary applications of surface reconstruction from cross-section curves is in interactive image segmentation, particularly for 3D volumes that arise from biomedical imaging (e.g., MRI or CT scans). In a typical session, an expert user would delineate boundaries of a region of interest (e.g., an organ) on selected 2D slices of the 3D volume, and the computer would reconstruct a surface that interpolates those cross-sectional curves. Even with the advances in automatic segmentation methods, interactive segmentation remains a standard practice since it is difficult for existing automated methods to perform accurately and consistently on real-world imaging data.

The majority of reconstruction algorithms are specialized for parallel cross-sections, in part due to the natural choice of axial planes of 3D volumes for boundary delineation. A key limitation of using parallel slices, however, is the restriction of the slice orientation. Allowing a user to choose planes whose orientations are adapted to the 3D shape has the potential to lower the number of planes needed to segment the shape, and thereby reducing human delineation time. This hypothesis has motivated the development of several reconstruction algorithms that are capable of handling arbitrarily oriented cross-sections [28, 94, 15, 18, 70, 166, 71, 77]. Some of these algorithms are further capable of reconstructing material interfaces given multi-labelled slices [94, 15, 18, 77].

Despite the availability of these algorithms for surfacing non-parallel slices, a critical but overlooked challenge in deploying these algorithms in practice is making sure that the input slices to these algorithms are *consistent*. A set of slices is said to be consistent if, for any two slices that intersect at a line l , the labelling induced by the curve networks on each slice agrees on l (see Figure 3.1). The majority of surfacing algorithms *require* a consistent input. Unfortunately, many applications that produce slices for surfacing do not guarantee consistency among the slices. For example, in most off-the-shelf software for interactive 3D volume segmentation, an expert delineates boundaries on each slice independently, and there is no mechanism in the software to enforce consistency among what the expert draws on intersecting slices.

For inputs consisting of only few slices with simple shapes (such as the one in Figure 3.1), manual corrections may be possible to restore the consistency. However, the task can become intractable even for modest size data sets (see Figures 3.6, 3.8, 3.10). We therefore propose an algorithm to automatically restore consistency. Our algorithm fills the gap between real-world data (which are often inconsistent) and existing surfacing algorithms (which typically require consistency), hence allowing these algorithms to be more easily adopted in practice.

Problem statement Restoring label consistency can be stated as a constrained curve deformation problem. We are given a set of planes, each divided by a curve network into regions equipped with labels $1, \dots, n$ ($n \geq 2$). We wish to deform the curve network on each plane in a minimal way to guarantee that the labels on any pair of intersecting planes are consistent along their intersection line (Figure 3.1).

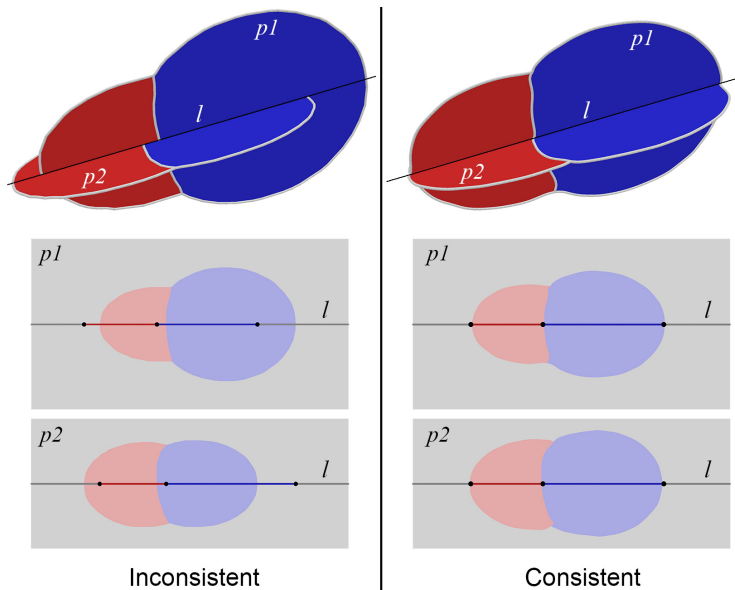


Figure 3.1: Curve networks on two intersecting planes (p_1, p_2) with inconsistent (left) and consistent (right) labeling. The pictures at the bottom show the labeling on each plane as well as the labeling from the other plane on the intersection line (l).

Contributions We present the first algorithm for restoring consistency to non-parallel cross-sections. Our algorithm would allow surfacing algorithm for non-parallel cross-sections to be able to process a much wider range of inputs that are often generated by practical applications. Technically, we make two main contributions:

1. We formulate restoration of label consistency as a constrained non-linear optimization problem using a multi-labelled implicit representation.

2. We propose a novel solution strategy of this challenging optimization problem. The solution is shown to be efficient and effective on real world data sets.

3.2 Relate work

We briefly review the work of curve deformations that is related to ours. And please refer to Sec 1.1 for a review of reconstruction from cross-sections.

Geometric deformations Deformation of geometry, being 2D curves or 3D surfaces, is a major research topic in computer graphics and related domains. Broadly speaking, deformation methods can be classified into explicit or implicit ones based on how the deformation is represented. *Explicit* methods either deform the geometry itself, by computing displacements of points on the geometry [32], or deform the embedding space around the geometry, using some control structure such as lattices [131], cages [82], skeletons [98], and points [126]. On the other hand, *implicit* methods represent the geometry as the level set of an implicit function in 2D (for curves) or 3D (for surfaces) and indirectly deform the curve by modifying the values of the functions [24]. Implicit representations are particularly attractive for interactive editing due to the ease in manipulating the functions [25, 128, 129]. Other popular ways for manipulating the functions include interpolating between given functions (as in shape metamorphosis [78, 44, 37, 150]), associating the function with a particle system [47], evolving the function with a user-specified speed function (as in the powerful level-sets method [132, 145]), and interpolating functional or positional constraints using a variational framework [148, 84, 70, 160].

Deformation methods aim at fulfilling well-defined objectives specified by the user or by the application. Examples of such objectives are handles or sketches (as in interactive editing), a target shape (as in shape metamorphosis or registration), or intensity features in an image

(as in image segmentation). We are not aware of any deformation method that is designed for consistency objectives like those in our problem. Our method falls into the class of variational implicit methods [148, 84, 70, 160]. Unlike existing methods that formulate deterministic constraints as simple equalities or inequalities, our method uses additional integer variables to encode the uncertainty of labelling along intersection lines, which results in a more challenging optimization problem.

3.3 Problem formulation

A perhaps tempting strategy for restoring consistency is to directly deform the input curve networks. Such strategy would find some minimum-energy displacement of the curve vertices constrained by the requirement that, for any two slices intersecting at line l , the curve networks on both slices should intersect with l at the same set of locations. However, formulating such constraint is difficult, since we do not know the number or the location of these intersection points on l . Furthermore, the topology of the curve networks remains fixed in this strategy, which reduces the flexibility of the deformation.

We adopt a different strategy that makes it easy to formulate the consistency constraints and also permits topological changes of the curve networks. The curve network on each plane is implicitly represented by functions over the plane, and curve deformation is indirectly achieved by modifying the functions. In this representation, consistency among slices can be formulated by linear inequality constraints along the intersection lines after introducing additional integer variables (i.e., the labels of vertices on the intersection lines). By expressing the amount of curve deformation as an energy term on the functions, we can cast the task of restoring consistency as a constrained mixed-integer optimization problem.

We first discuss the implicit representation of curve networks on each plane in Section 3.3.1. After introducing our deformation energy using this representation in Section 3.3.2, we present our optimization formulation in Section 3.3.3.

3.3.1 Implicit representation

Curves that partition the plane into two labels can be implicitly represented as the level set of a scalar function. To represent a curve network that partitions the plane into $n > 2$ labels, we consider a commonly used implicit representation that utilizes a vector function [97, 60, 160, 77]. We first briefly review such representation, and then discuss our choice of the initial vector function and how it is discretized on input cross-sections.

Implicit representation To represent an n -labelled domain, we consider a vector function $\vec{f}(\vec{x}) = \{f_1(\vec{x}), \dots, f_n(\vec{x})\}$ where each f_i is a continuous scalar function and \vec{x} is a point in the domain. The value of $f_i(\vec{x})$ can be intuitively understood as the “strength” of label i at \vec{x} . We assign each point \vec{x} the label that has the maximal strength,

$$Label(\vec{x}) = \arg \max_{i=1, \dots, n} f_i(\vec{x}). \quad (3.1)$$

We are interested in the boundary between regions of different labels, known as the *interface set* [77]. More precisely, \vec{x} is on the interface set if $Label(\vec{x})$ contains more than one label. Interface sets are natural generalizations of level sets, since the interface set for $n = 2$ labels are equivalent to the level set of some scalar function [77]. For $n > 2$ labels, an interface set in the 2D domain can be made up of curve segments meeting at non-manifold junctions (where three or more labelled regions meet).

Initial vector function We seek a vector function \vec{f} whose interface set reproduces an input curve network. This is equivalent to asking that the labelling defined by \vec{f} (Equation

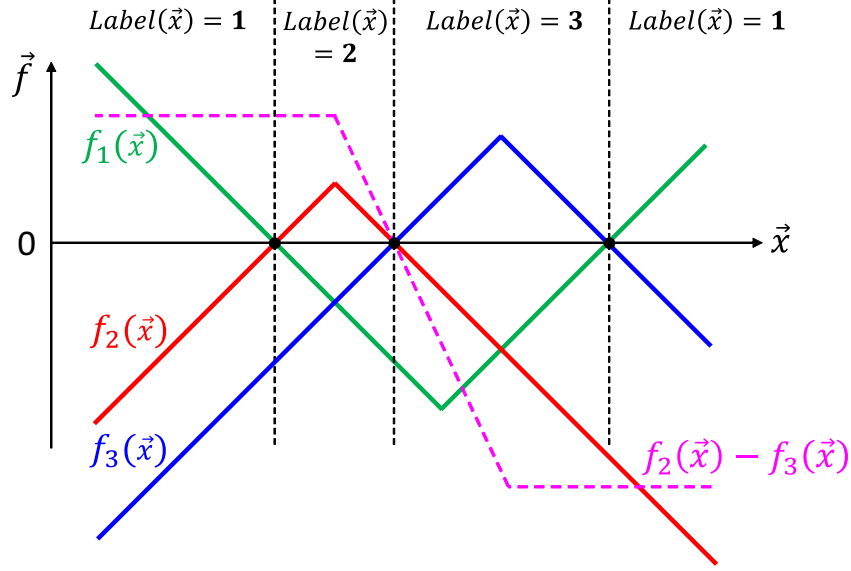


Figure 3.2: Vector function $\vec{f} = \{f_1, f_2, f_3\}$ defined as signed distance functions over a three-labelled 1D domain. Note that the difference function $f_2 - f_3$ (magenta dotted graph) is a distance-like function locally at the interface between labels 2 and 3.

3.1) coincides with the labelling of regions partitioned by the network. Our definition of \vec{f} is based on signed distance functions. Specifically, consider a point \vec{x} lying inside a region with input label i . We set $f_j(\vec{x})$ to be positive only if $j = i$ and negative for all other labels j . The magnitude of $f_j(\vec{x})$ is set as the Euclidean distance from \vec{x} to the nearest curves that bound regions with label j . If label j is absent from the plane, we set $f_j(\vec{x}) = -\infty$. It is easy to see that this definition of \vec{f} ensures that $Label(\vec{x}) = i$. Figure 3.2 illustrates the vector function over a three-labelled 1D domain.

While one could define the vector function in more sophisticated ways (e.g., using higher degree polynomials [160]), we chose signed distance functions not only due to its simplicity but also because of the convenience it offers for formulating the deformation energy (Section 3.3.2). A key observation is that a Euclidean distance function has a constant gradient magnitude, hence a small change to the function values leads to a bounded spatial displacement of its level set. Also, adding a constant function to a distance function leads to level sets with

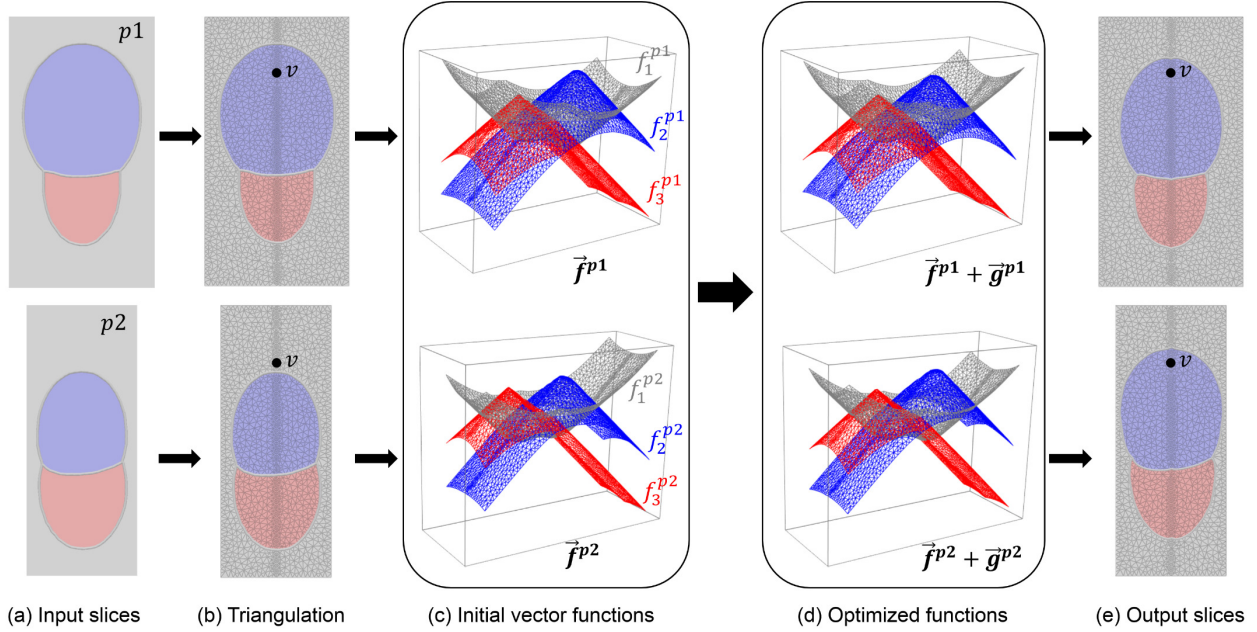


Figure 3.3: Overview of our algorithm. Each input slice (a) is first triangulated (b), and a vector function is computed per slice to reproduce the input labels (c). Then functions on all slices are optimized together to enforce label consistency while minimizing deformations (d). Finally, the output curve networks are extracted as the interface sets of the optimized functions (e).

similar shapes (i.e., offset curves). As a result, the magnitude and variation of the change to a distance function correlates with the amount of deformation of the level set. This is not true for general smooth functions. Note that the interface set between two labels i, j coincides with the zero-level set of their difference function, $f_i - f_j$. In our signed-distance-based definition of \vec{f} above, $f_i - f_j$ is similar to a signed distance function (with a constant gradient magnitude of 2) in the vicinity of the interface set between labels i and j (see Figure 3.2). As a result, we can approximately measure the deformation of the interface set between labels i, j by the magnitude and variation in the change to the difference function $f_i - f_j$, as we shall elaborate in Section 3.3.2.

Discretization We use piecewise linear vector functions encoded by values on vertices of a triangulation. A cross-section plane is discretized by a constrained Delaunay triangulation

where the constrained set includes edges and vertices of the curve network as well as the intersection lines with other planes. For consistency between cross-sections, all planes that share one intersection line l use a common set of vertices and edges on that line. This set is created by first uniformly sampling l and then adding new vertices where l intersects with the curve networks or with other intersection lines (e.g., where three or more planes meet at a point). The sampling density along l is chosen to be sufficiently high to capture the inconsistency among the planes. We then use Shewchuk’s Triangle package [136] to compute the triangulation. Figure 3.3 (b) shows the triangulation of the two slices on the left of Figure 3.1.

After triangulation, we compute a vector $\vec{f}(v) = \{f_1(v), \dots, f_n(v)\}$ at each vertex v as defined above. To obtain the region label at each vertex, and assuming that each input curve is equipped with labels on its two sides, we use a flooding process over the triangulation to obtain the labels of vertices that are not on the curve network. To avoid numerical difficulties, for each vertex v on the input curve network that bounds regions with labels $\Phi \subseteq \{1, \dots, n\}$, we assign v the label with the lowest index in Φ and set the magnitude $\|f_i(v)\|$ for all $i \in \Phi$ to be a small positive constant ϵ . Figure 3.3 (c) visualizes, for each slice in (b), the three scalar functions corresponding to the three labels.

Interface set reconstruction Given the initial vector functions, the core of our method (Section 3.4) is an optimization process that modifies these functions to ensure label consistency between slices. After optimization, we need to extract the interface set of the modified vector function on each slice as the output curve network. To do so, we use a dual scheme akin to that in previous works [60] but over a triangulation. We create two types of points on the output curve network, either on triangle edges (called edge points) or inside triangle faces (called face points). For each triangle edge between two vertices with different labels i, j , we locate the edge point as the zero-crossing of the function $f_i - f_j$ along that edge. For each

triangle whose vertices do not have a common label, we locate the face point as the centroid of the edge points. Finally, to create the network, we connect each face point inside a triangle to the edge points on the triangle’s edges. To produce smoother results while conforming to vertex labelling, we fair the curve networks using the non-shrinking centroid-averaging scheme of Taubin [142] while constraining each edge (resp. face) point to lie on the respective triangle edge (resp. face). Figure 3.3 (e) shows the interface sets from the modified functions in (d).

3.3.2 Deformation energy

While there are many ways to modify the input to achieve consistency, we are looking for a way that best preserves both the location and shape of the curve networks. As mentioned earlier, our implicit representation makes it possible to capture the amount of curve deformation by the magnitude and variation in the changes to (the difference of) the implicit functions.

Consider an initial vector function \vec{f} in any domain, and let the modified function be $\vec{f} + \vec{g}$ where \vec{g} is another vector function. In the continuous setting, our deformation energy has the integral form

$$E(\vec{g}) = \int \lambda \sum_{i \in R} \|g_i(\vec{x})\|^2 + \sum_{i,j \in R} \|\nabla(g_i(\vec{x}) - g_j(\vec{x}))\|^2 d\vec{x} \quad (3.2)$$

where $R \in \{1, \dots, n\}$ is the set of labels where $f_i \neq -\infty$ for any $i \in R$. The first term inside the integral measures the magnitude of the change to each scalar function, and the second term measures the variation (as Dirichlet energy) of the change to the difference between scalar functions. The two terms are balanced by a user-specified constant λ . Note that penalizing the first term has the indirect effect of penalizing the change to the difference

between scalar functions, but this term additionally acts as a regularizer to prevent spurious solutions of \vec{g} that differ by a constant function.

For a piecewise linear vector function \vec{g} over a triangulation of the plane, the integral energy in Equation 3.2 can be equivalently expressed in the following matrix form

$$E(\mathbf{g}) = \mathbf{g}^T M \mathbf{g} \quad (3.3)$$

where \mathbf{g} is a flattened list of values $g_i(v)$ for all vertices v in the triangulation and labels $i \in R$. The matrix M encodes the geometric structure of the triangulation and can be derived from Equation 3.2 using integrals of barycentric coordinates over triangles. Specifically, let $M_{a,b}$ denote the $|R| \times |R|$ submatrix whose top-left position in M is $\{(a-1)|R|+1, (b-1)|R|+1\}$. The submatrix is non-zero only if the a -th and b -th vertices share a triangle edge, in which case it has the form

$$M_{a,b} = \begin{cases} \lambda \sum_{t \in T_a} \sigma_t I_{|R|} + \sum_{c \in V_a} \omega_{a,c} H_{|R|}, & \text{if } a = b \\ \frac{\lambda}{2} \sum_{t \in T_{a,b}} \sigma_t I_{|R|} - \omega_{a,b} H_{|R|}, & \text{if } a \neq b \end{cases} \quad (3.4)$$

where T_a, V_a are the list of triangles and vertices in the 1-ring neighborhood of the a -th vertex, $T_{a,b}$ is the list of triangles containing both the a -th and b -th vertices, σ_t is the area of triangle t , $\omega_{a,b}$ is the cotangent weights [117] for the edge between the a -th and b -th vertices, I_m is the identity matrix of size m , and H_m is the Laplacian matrix of a complete graph with m nodes.

3.3.3 Optimization formulation

We wish to modify the implicit functions on all planes to achieve two goals. First, any vertex v shared by multiple planes should have the same label on those planes. That is, on each of those planes, the corresponding scalar function of that label should be no smaller than the

function of any other label at v . Second, the sum of the deformation energy (E in Equation 3.3) over all planes should be minimized. Hence we have an optimization problem with a quadratic objective function (the second goal) and linear inequality constraints (the first goal). Note that formulating the inequality constraints requires the knowledge of the final label of each vertex on the intersection lines between slices. As a result, we formulate a mixed integer problem that solves for both function values at all vertices (which are real-valued) and labels at those vertices on intersection lines (which are integers).

We now detail the formulation, starting with some notations. Let P be the set of triangulated planes and \vec{f}^p be the initial vector function on each plane $p \in P$. Let I be the set of all vertices on the intersection lines between planes, and denote by P_v the planes on which v lies. Note that we need to store at a vertex $v \in I$ multiple vectors $\vec{f}^p(v)$, one for each plane $p \in P_v$. Since not all labels are present on every plane, we denote by R_p the set of labels present on the plane p and R_v the set of labels present on all planes $p \in P_v$. In the example of Figure 3.3, the input planes are $P = \{p1, p2\}$, and $P_v = P$ for any vertex v on the intersection line. As both planes contain three labels, we have $R_{p1} = R_{p2} = R_v = \{1, 2, 3\}$.

We solve for the change in the vector function \vec{f}^p on each plane p , denoted by the vector function \vec{g}^p (see Figure 3.3 (c,d)), as well as one integer label $L(v) \in R_v$ at each vertex $v \in I$. Rewriting \vec{g}^p as a flat list \mathbf{g}_p that consists of $g_i^p(v)$ for each vertex v on p and each label $i \in R_p$, the optimization objectives are:

$$\text{Minimize: } \sum_{p \in P} \mathbf{g}_p^T M_p \mathbf{g}_p \quad (3.5)$$

$$\text{Subject to: } g_{L(v)}^p(v) + f_{L(v)}^p(v) \geq g_i^p(v) + f_i^p(v) + \epsilon, \quad (3.6)$$

$$\forall v \in I, p \in P_v, i \in R_p, i \neq L(v)$$

This formulation minimizes the sum of deformation energy over all planes (Equation 3.5) while enforcing consistency of labelling over all vertices on the plane intersections (Equation 3.6). Here, M_p is the matrix used in Equation 3.3 over the triangulated plane p .

As the number of vertices on all planes can be large (typically thousands), solving the optimization problem as formulated above can be prohibitively expensive. To reduce the problem size, we make two observations. First, the linear inequalities in (3.6) only involve function values at vertices on the intersection lines (I). Second, since the deformation energy is quadratic, the minimal energy after fixing the values at a subset of the vertices (e.g., I) can be expressed as a quadratic function of these values. As a result, we can re-formulate the optimization problem to solve for both function values and labels *only* at vertices in I .

Specifically, let $\mathbf{g}_p = \{\mathbf{g}_{p,U}, \mathbf{g}_{p,I}\}$, where $\mathbf{g}_{p,I}$ are values at vertices on the intersection lines between p and other planes, and $\mathbf{g}_{p,U}$ are values at the remaining vertices on p . In this ordering, the matrix M_p has the form

$$M_p = \begin{vmatrix} A & B \\ B^T & C \end{vmatrix} \quad (3.7)$$

where A is a square matrix of size $|\mathbf{g}_{p,U}| \times |\mathbf{g}_{p,U}|$. Let $N_p = C - B^{-1}A^{-1}B$ (i.e., the Schur complement of the block A of matrix M_p), the minimal energy of (3.5) can be re-written using only variables on I as

$$\text{Minimize: } \sum_{p \in P} \mathbf{g}_{p,I}^T N_p \mathbf{g}_{p,I} \quad (3.8)$$

Note that, unlike the sparse matrix M_p , N_p is dense. However, we have observed in our experiments that the number of vertices on the intersections ($|I|$) are usually 1 to 2 orders of magnitude fewer than the total number of vertices, and hence using the objective of Equation 3.8 still yields a significant speed-up over the original form in Equation 3.5

To summarize, the reduced formulation solves for values $\bar{g}^p(v)$ and labels $L(v)$ only for vertices $v \in I$, with the objective of (3.8) and constraints of (3.6).

3.4 Optimization

The optimization problem can be understood intuitively as searching for the minimum of a convex energy (Equation 3.8) over a set of disjoint polytopes in the solution space. To see this, consider a higher dimensional space \mathcal{G} where each point $\mathbf{g} \in \mathcal{G}$ represents the vector of real variables over all planes, that is, $\mathbf{g} = \cup_{p \in P} \mathbf{g}_{p,I}$. Each set of labels $L(v)$ for all vertices $v \in I$ yields a set of linear inequalities in (3.6), which in turn defines a convex polytope of feasible region in \mathcal{G} . The goal is to find a label set L such that the minimal energy within the corresponding polytope in \mathcal{G} is smallest among all polytopes.

A standard trick to solve such problems is to cast it as a mixed-integer program (MIP). To do so, we first replace the integer variable $L(v)$ by an array of binary variables $m_i(v)$, one for each label $i \in R_v$, so that $m_i(v) = 1$ only if $i = L(v)$. Keeping the energy goal as in (3.8), an equivalent mixed-integer linear program (MILP) can be constructed by replacing the linear equalities in (3.6) with

$$(1 - m_i(v))C + g_i^p(v) + f_i^p(v) \geq g_j^p(v) + f_j^p(v) + \epsilon, \tag{3.9}$$

$$\forall v \in I, p \in P_v, i, j \in R_p, i \neq j,$$

where C is a large constant, and by adding new constraints including $m_i(v) \geq 0$ for all i and $\sum_{i \in R_v} m_i(v) = 1$. Alternatively, we can construct a mixed-integer non-linear program (MINLP) by replacing (3.9) with

$$m_i(v)(g_i^p(v) + f_i^p(v) - g_j^p(v) - f_j^p(v) - \epsilon) \geq 0, \tag{3.10}$$

$$\forall v \in I, p \in P_v, i, j \in R_p, i \neq j,$$

However, these MIP formulations have their own drawbacks. The large constant C used in the MILP formulation may lead to weak linear programming relaxation and numerical issues, whereas the constraints in the MINLP formulation are non-convex. These limitations result in low efficiency in the solution process. When testing on our data sets, where the number of variables can be on the order of hundreds, we found that state-of-the-art MIP solvers (e.g., Gurobi) fail to return a solution even after running for hours.

We propose an efficient method for solving our optimization problem without converting it to MIP. The key observation is that, given a label set L , minimizing the energy within the polytope of L is a quadratic programming (QP) problem, which can be solved much more efficiently than MIP. Using the QP as a building block, we follow a greedy strategy to search for the optimal label set. It starts with a initial labeling obtained from the signed distance functions along the intersection lines. It then incrementally changes the labelling, one vertex at a time, to decrease the QP energy. These two stages are detailed next.

3.4.1 Initial labels

A straight-forward scheme to initialize the label of a vertex v on an intersection line is to average the signed distance vectors over all planes containing v and take the label with

the maximum value in the averaged vector. Specifically, recall that P_v is the set of planes containing v and R_v is the set of labels present on all these planes, this scheme initializes label $L(v)$ for all $v \in I$ as:

$$L(v) = \arg \max_{i \in R_v} \sum_{p \in P_v} f_i^p(v) / |P_v|. \quad (3.11)$$

This simple scheme, however, may produce “jumps” in the labels along an intersection line. In particular, the labelling along an intersection line l between two planes p_1, p_2 may suddenly change at a vertex v where l meets another intersection line l' between planes p_1, p_3 . The jump is caused by the fact that $L(v)$ considers the function values on all three planes p_1, p_2, p_3 while the labels at the remaining vertices of l consider function values on only two planes p_1, p_2 .

To create a smoother set of labels, we proceed in two steps. In the first step, we use Equation 3.11 to determine labels at those vertices that lie on multiple intersection lines. We call these vertices *junctions*, denoted by J . In the second step, we modify the function along each intersection line to match the labels at the junctions while maintaining the smoothness of the original function. The modified functions are then used to obtain the labels of the non-junction vertices using the simple averaging scheme above.

Specifically, for the second step, we represent the change of the original function \vec{f}^p along an intersection line l on a plane p as another vector function $\vec{h}^{p,l}$. Note that there will be one function $\vec{h}^{p,l}$ for each plane p that contains l . We wish to find $\vec{h}^{p,l}$ such that the modified function $\vec{f}^p + \vec{h}^{p,l}$ along l is as similar to \vec{f}^p as possible while matching the label at each junction vertex on l . To measure similarity, we consider the same energy as Equation 3.2 but over a 1-dimensional line, which penalizes the integral of the squared magnitude and gradient of $\vec{h}^{p,l}$ along the line. Using this energy, we need to solve a quadratic program for each plane

p and intersection line l . The variables are $h_i^{p,l}(v)$ for each vertex $v \in l$ (including junction vertices) and each label $i \in R_p$ (labels present on p). The objective and constraints are:

$$\text{Minimize: } \mathbf{h}_{p,l}^T M_l \mathbf{h}_{p,l} \quad (3.12)$$

$$\text{Subject to: } h_{L(v)}^{p,l}(v) + f_{L(v)}^p(v) \geq h_i^{p,l}(v) + f_i^p(v) + \epsilon, \quad (3.13)$$

$$\forall v \in J \cap l, i \in R_p, i \neq L(v),$$

where $\mathbf{h}_{p,l}$ is a flattened list of $h_i^{p,l}(v)$. Matrix M_l has a similar structure as M in Equation 3.3. Specifically, the $|R_p| \times |R_p|$ submatrix of M_l whose top-left position is $\{(a-1)|R_p| + 1, (b-1)|R_p| + 1\}$ is non-zero only if the a -th and b -th vertices share a common edge on l . Let the edge between the a -th and b -th vertices be e , each non-zero submatrix has the same form as Equation 3.4 except that now T_a is the list of edges sharing the a -th vertex, $T_{a,b} = \{e\}$, σ_t is the length of edge t , and $\omega_{a,b} = 1/\sigma_e$.

After solving for $\vec{h}^{p,l}$ for all planes p containing an intersection line l , and denoting this set of planes by P_l , we obtain the label for all non-junction vertices v on l by averaging functions over these planes:

$$L(v) = \arg \max_{i \in R_v} \sum_{p \in P_l} (f_i^p(v) + h_i^{p,l}(v)) / |P_l|. \quad (3.14)$$

3.4.2 Updating labels

As mentioned before, fixing the vertex labels L reduces our optimization problem (3.8,3.6) to a quadratic program (QP). We consider the minimal energy of this QP as a function of L , denoted by $E(L)$. Starting from an initial label set L_0 , we will create a sequence of label sets L_1, L_2, \dots such that $E(L_{k+1}) < E(L_k)$ for $k \geq 0$.

Our approach is guided by the observation that $E(L_k)$ is always achieved by some point on the boundary of the polytope of L_k . To see this, observe that the convex energy E has a unique local minimum in the solution space \mathcal{G} that corresponds to no change to the initial implicit functions. Assuming the input is inconsistent, the minimal-energy solution does not lie inside any feasible regions. As a result, the minimizer in the polytope of L_k has to lie on one or more facets of the polytope. Intuitively, the polytopes that are “on the other side” of these facets are likely to have even lower energy. We therefore enumerate these polytopes and pick one with the lowest energy as our next label set L_{k+1} .

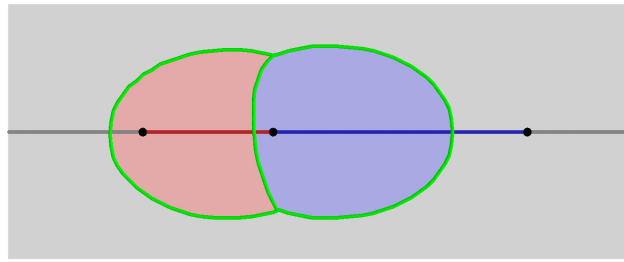
More specifically, each facet of the polytope of L_k corresponds to an equality in the constraint set (3.6), or

$$g_{L_k(v)}^p(v) + f_{L_k(v)}^p(v) = g_i^p(v) + f_i^p(v) + \epsilon$$

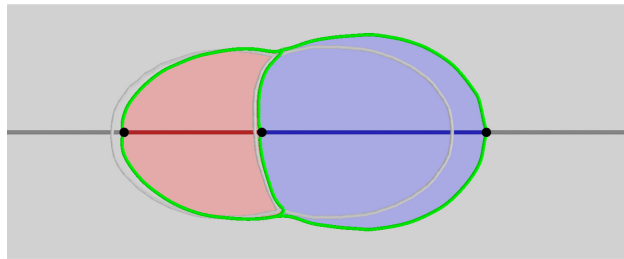
for some vertex v , plane p , and label i . The polytope “on the other side” of this facet corresponds to values of $g_{L_k(v)}^p(v), f_{L_k(v)}^p(v)$ that make the left-hand side smaller than the right-hand side. With ϵ being a small constant, the inequality would change the label of v from $L_k(v)$ to i . This leads to the following simple algorithm. First, we identify all vertex-label pairs $\{v^*, i^*\}$ that satisfy the above equality on some plane. For each such pair, we create a new label set L^* such that $L^*(v) = L(v)$ for all $v \neq v^*$ and $L^*(v^*) = i^*$, and then compute $E(L^*)$ by solving QP (3.8,3.6). We then choose the next label set L_{k+1} to be the L^* with minimal $E(L^*)$, if such minimal energy is smaller than $E(L_k)$. Otherwise, the process terminates and outputs L_k as the solution.

The optimization process is illustrated in Figure 3.4 on the simple example from 3.1. Observe that the vertices whose labels change during the updates are located close to the interface set. In practice, we have observed that the initial labels obtained by our method are usually

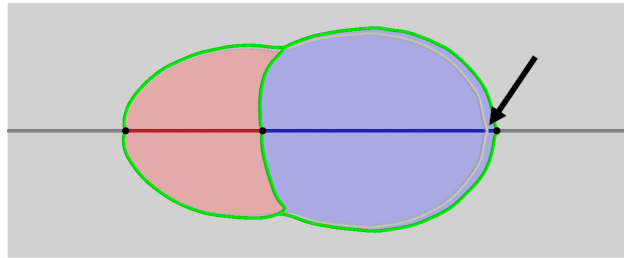
fairly close to the final labels (see next section). As a result, iterative updates can converge quickly to a locally optimal solution.



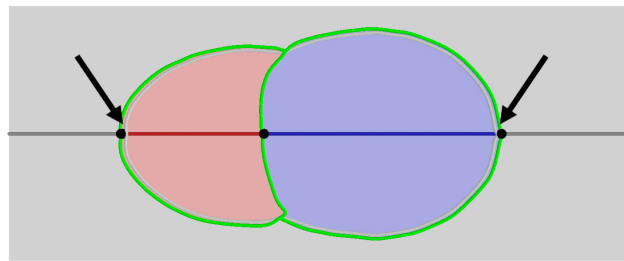
(a) Input



(b) Initial labels



(c) Update (1st iteration)



(d) Update (3rd iteration & final)

Figure 3.4: Optimization process on the input in Figure 3.1 (left, plane p_2), showing the labeling on the plane (as red, blue, gray colors) and interface set (green curves) in the input (a), after initializing the labels on the intersection lines (b) (see Section 4.1), and after the first (c) and final (d) iterations of label updates (see Section 4.2). Interface sets in previous steps are shown in white curves in subsequent steps in (b,c,d) for comparison, and locations where vertices change labels are indicated by arrows.

3.5 Experimental results

We test our algorithm on simple synthetic inputs as well as a few non-trivial examples describing anatomical structures. Our implementation uses Gurobi for solving the quadratic program (3.8,3.6) given a fixed label set L .

Choice of parameter We first evaluate the effect of the parameter λ in our deformation energy (Equation 3.2). Figure 3.5 shows the results of our method on the same input in Figure 3.1 (left) for different values of λ . Observe that λ controls the trade off between two competing goals: maintaining the location of the input curves and preserving their shape. If the value is too large (e.g., $\lambda = 100$), most of the input curve network is kept in place but at the cost of severe shape distortion around the intersection lines (highlighted by the boxes) to satisfy consistency. If the value is too small (e.g., $\lambda = 0.0005$), our method will strive to maintain the overall shape of the curves but may produce a significant amount of scaling. Nevertheless, we found that there is a reasonably large range of values of λ (e.g., between 0.001 and 0.1) for which our method produces plausible results for our test examples. These parameters are reported in Table 3.1.

Complex examples We test our method on several non-trivial biological data sets containing multiple planes (5 or more) that intersect with each other in complex ways (Figures 3.6, 3.7, 3.8, 3.9, 3.10). All of the examples exhibit a large number of inconsistencies. In the case of multi-labelled data (Figures 3.8, 3.9, 3.10), observe that the interaction between multiple labels would make it very difficult to rectify manually while simultaneously preserving the shape of the input contour. Our method is capable of restoring consistency on all planes in each data. Note that in some cases the topology of the curve network changes in the output (Figure 3.8, plane p2, cyan region). The flexibility of allowing topological changes without additional effort is another benefit of using an implicit representation.

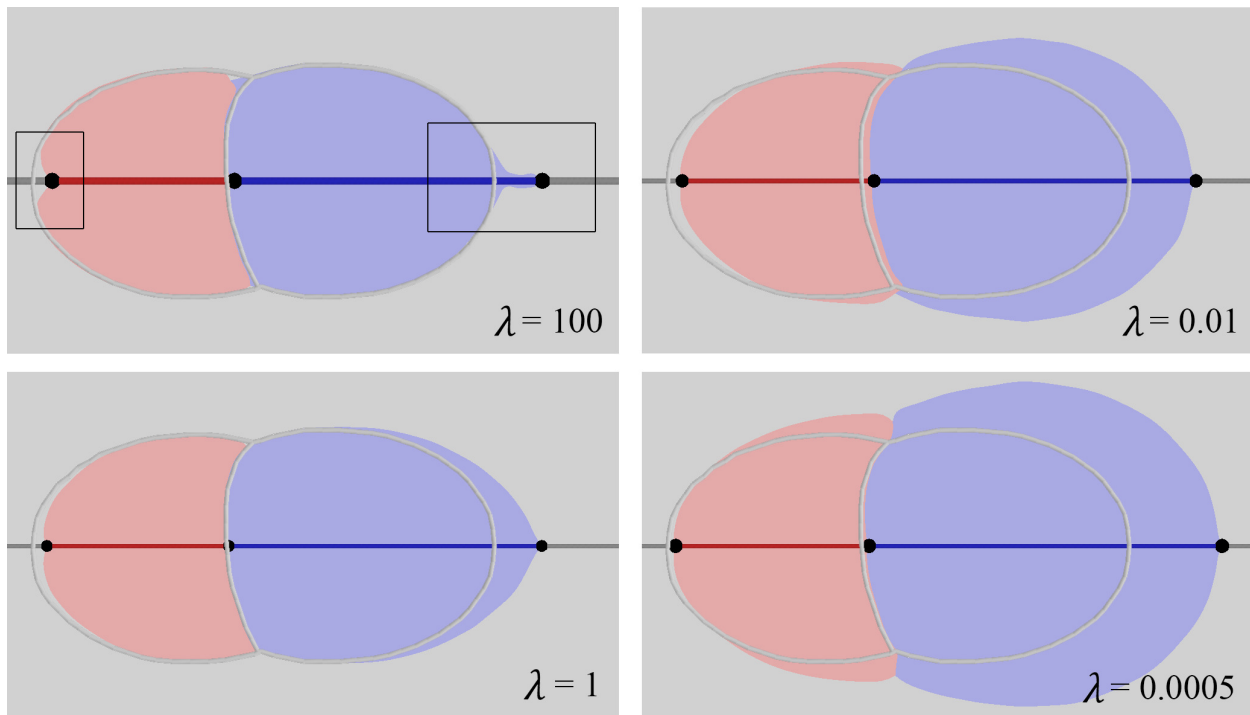


Figure 3.5: Results of our method (on input in Figure 3.1 left, showing plane p2) for different values of λ in Equation 3.2. The labeling is shown as colored regions, and the input curve network is shown as gray curves for reference.

The output of our algorithm can be utilized by any existing method for surfacing non-parallel cross-sections. While the surfacing method of [18] can be applied to an inconsistent input, the surface often contains artifacts near inconsistency between the cross-sections. An example is shown in the bottom-left of Figure 3.6 using the inconsistent input in the top-left (the artifacts are highlighted). Applying the same surfacing method to the consistent cross-sections produced by our algorithm results in an artifact-free surface (Figure 3.6 bottom-right).

To make the solution process more efficient for these complex examples, we further simplify the problem size by reducing the set of vertices I whose values and labels that we solve for in the optimization formulation (3.8, 3.6). We observed that vertices that change labels in the optimization process are either inconsistent to start with (i.e., having different labels on different planes in the input) or close to these inconsistent vertices on the intersection lines.

We therefore restrict the set I to inconsistent vertices plus a fraction η of all vertices on the intersection lines ranked in descending order by their distances to inconsistency vertices. We use $\eta = 0.1$ in all three examples. Theoretically, it is possible for some vertices that are on the intersection lines but excluded from I to become inconsistent, since there are no label constraints imposed on these vertices. In this case, one could re-run the optimization again by including the newly inconsistent vertices in I , and repeat the process until no more inconsistency is present. However, we have not had any need to run optimization more than once on our data set.

Performance The core of our algorithm solves a non-linear constrained optimization problem which can also be formulated as a mixed integer program (MIP). Such problems are notoriously challenging to solve even using carefully engineered general solvers. For example, we tried to use Gurobi to solve the MILP formulation as described in Section 3.4, and it failed to converge even after hours of running on all our complex examples (even after we restrict the set I). Our proposed approach is made efficient by our careful choice of initialization which places the starting guess close to the final solution (Section 3.4.1), and leveraging the efficiency of solving smaller quadratic programs (QP) (Section 3.4.2).

To better understand the performance of our method, we divide our processing time into three stages: a preprocessing stage that discretizes the planes and prepares data structures for optimization, initialization of labels, and iterative updates of labels. The preprocessing stage is dominated by the matrix inverse operation for obtaining the coefficient matrix N_p in the reduced energy objective (Equation 3.8). The inversion is done per plane p and the complexity depends on both the total number of vertices on p and the number of labels on that plane. The complexity of both the second and third stages depends on the size of the QP, which scales with the number of vertices and labels in the intersection set I , and the number of times QP is solved. For label initialization, QP is solved only twice per

	# Planes (Labels)	# Total vertices	$ I $	λ	Pre-proc time	Initial time	Update time	Gurobi time
Atrium	5 (2)	5740	109	0.01	0.6398	0.0258	0.448619	13.784
Ferret Brain	10 (2)	13131	300	0.01	3.1514	0.703	62.945	-
Liver (Fig 3.8)	5 (4)	8222	95	0.1	10.943	0.5324	13.4681	-
Liver (Fig 3.9)	6 (4)	20799	125	0.005	60.7131	0.628	29.1373	-
Mouse Brain	6 (7)	14159	168	0.025	127.661	2.394	291.436	-

Table 3.1: Data size and running time for the examples in Figures 3.6, 3.7, 3.8, 3.9, 3.10, showing the number of planes, number of labels, total number of vertices in the triangulations, number of vertices in the reduced intersection set I , λ value, and timing (in seconds) for each of the three stages of our method. ('-' indicates the solver fails to return within 2 hours)

intersection line, one for each plane containing the line, and hence is usually very efficient. For label updates, QP needs to be solved for possibly many times depending on the number of iterations required to terminate and the number of vertices that need to be checked in each iteration.

We report the performance of the stages of our algorithm in Table 3.1. Our algorithm was implemented in C++ and runs on a laptop with 2.5 GHz Intel Core i7 and 16GB RAM. Overall, our method finishes within minutes for all examples. The dominating stage is updating labels, followed by the preprocessing stage.

Optimality To evaluate the optimality of our solution, we compare our results to those obtained by Gurobi using the MILP formulation. Since Gurobi fails to run on any of our complex examples, we design the following experiment. We take a subset of k planes from the ferret brain data (Figure 3.7), for $k = 2, \dots, 5$, and run both Gurobi (solving MILP) and our method (Sections 3.4.1,3.4.2) for each k . We stopped at $k = 5$, beyond which Gurobi could not return an answer after running for two hours. We report the energy of the solution found by both methods in Table 3.2, as well as the running time of each method. Observe that our method is able to achieve the same energy as the general MIP solver for all experiments, yet

in significantly less time. Also note that our label initialization stage achieves close-to-optimal energy levels, which is an important factor for the fast convergence of our method.

k	Initialization energy	Final energy	Gurobi energy	$ I $	Our time	Gurobi time	$ I $	Our time	Gurobi time
2	16.97	16.65	16.65	16	0.274	0.135	61	0.826	1.05
3	26.49	24.95	24.95	32	0.354	0.421	121	1.253	11.28
4	26.46	25.02	25.03	47	0.531	0.719	181	3.024	33.16
5	36.14	29.55	29.55	93	1.471	25.26	430	33.218	619.91
6	53.03	46.74	46.74	181	9.923	1011.8	951	342.426	-

Table 3.2: Comparing our optimization method and the MIP solver in Gurobi on a subset of k planes in the ferret brain data, in terms of minimal energy and time (in seconds). Column 5, 6, 7 show the time on reduced set, and column 8, 9, 10 show the time on original set on intersection lines.

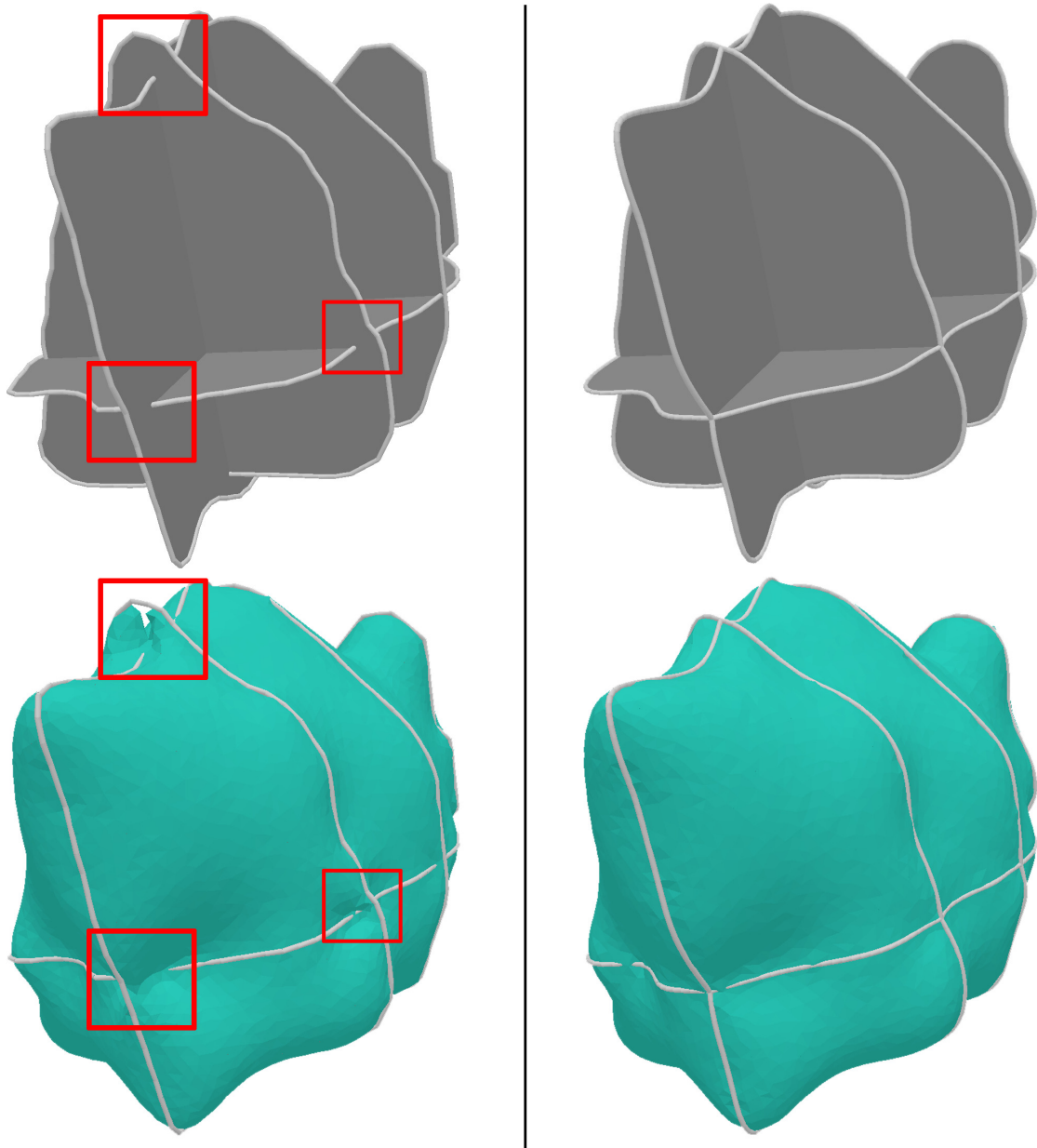


Figure 3.6: The result (top-right) of repairing an inconsistent two-labelled Atrium data set (top-left, several inconsistencies are highlighted), and surfaces reconstructed from these two sets of slices using [18] (bottom; observe the artifacts in bottom-left).

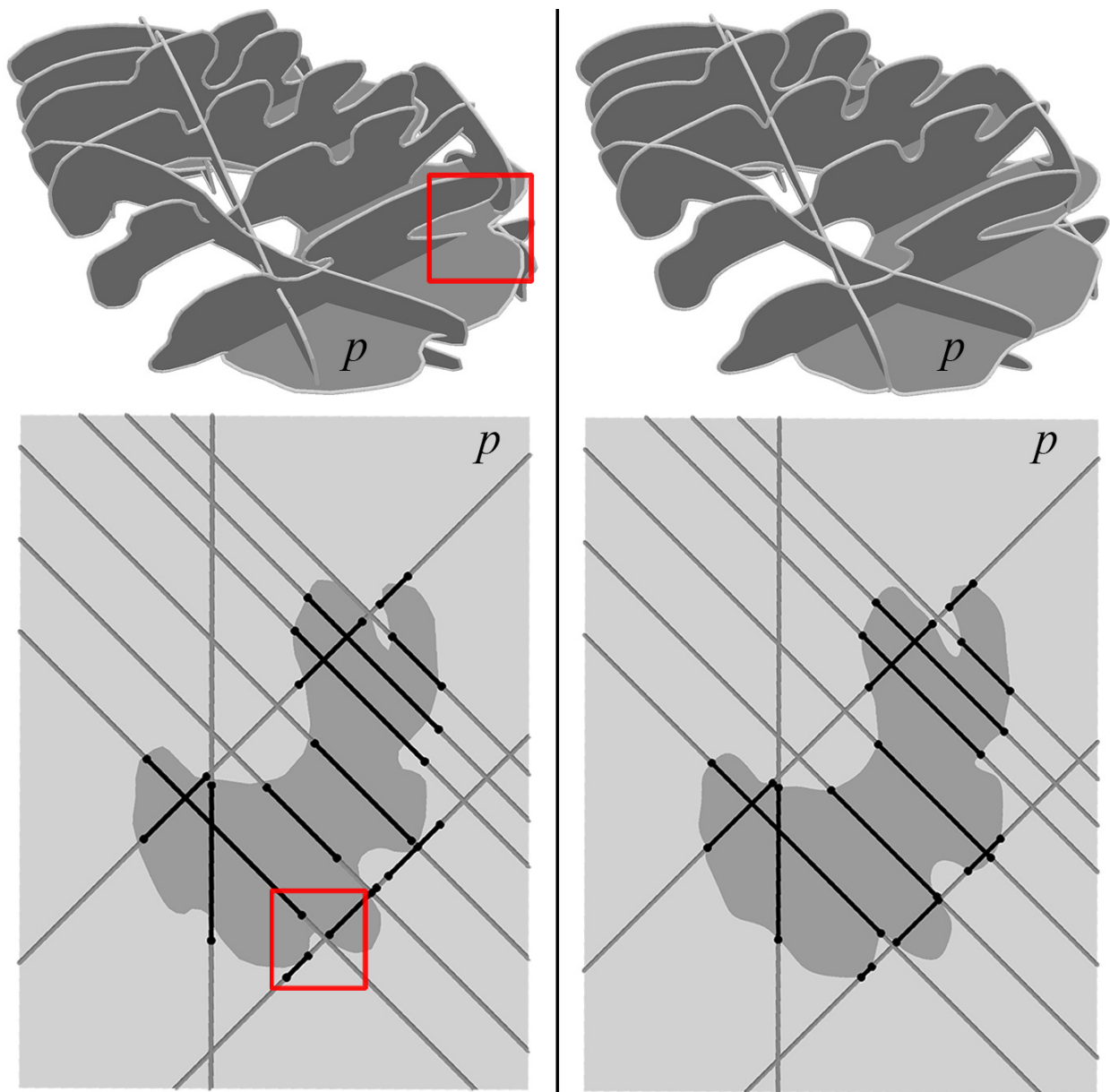


Figure 3.7: The result (right) of repairing a two-labelled ferret brain data set that is highly inconsistent (left, one inconsistency is highlighted). The bottom pictures show the labeling on one of the planes (p) as well as labelling from other planes on intersection lines.

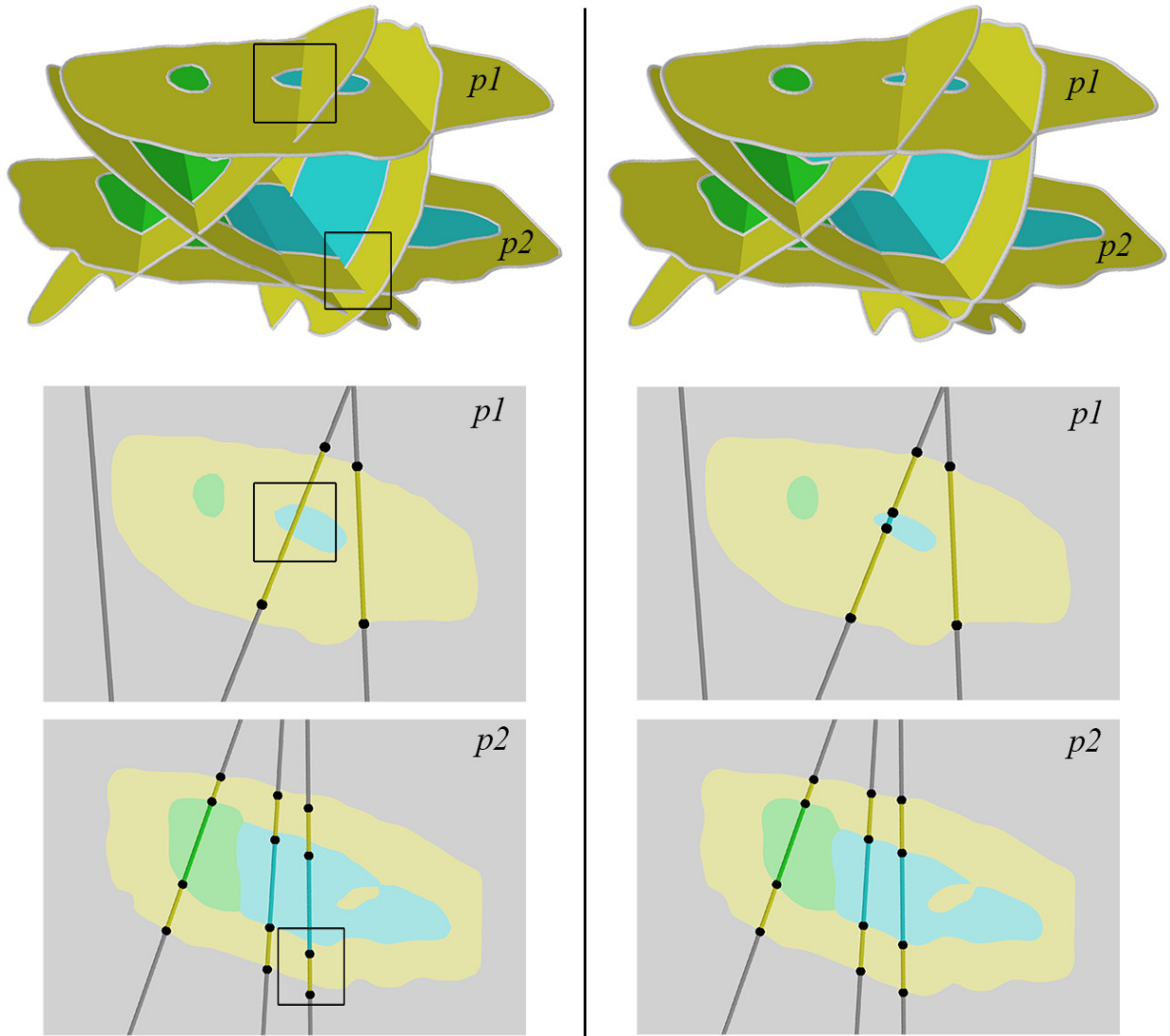


Figure 3.8: The result (right) of repairing a 4-labelled liver data set (left, two inconsistencies are highlighted), showing the labeling on two planes ($p1, p2$) at the bottom.

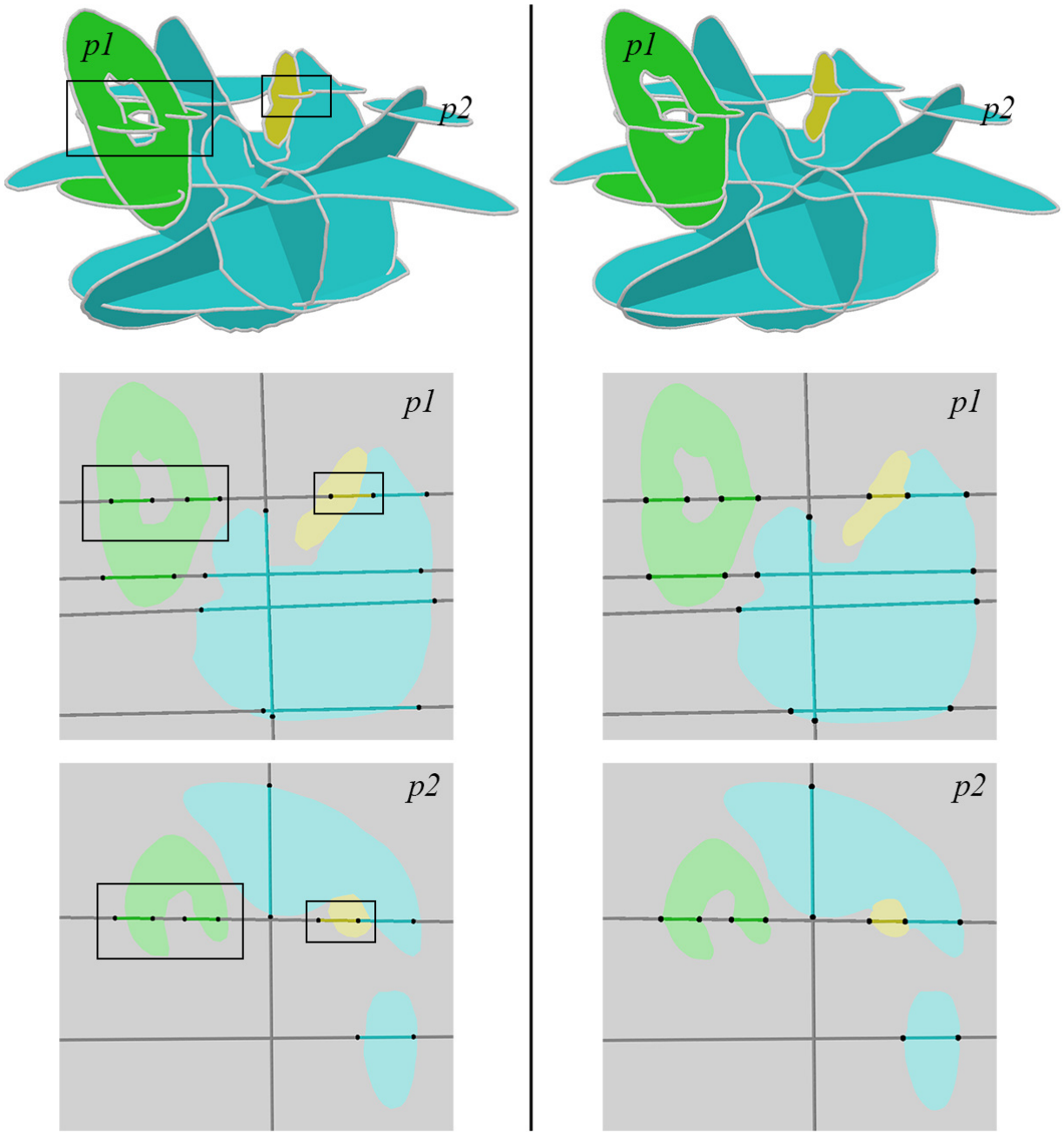


Figure 3.9: The result (right) of repairing another 4-labelled liver data set (left, two inconsistencies are highlighted), showing labelling on two planes ($p1, p2$) at the bottom.

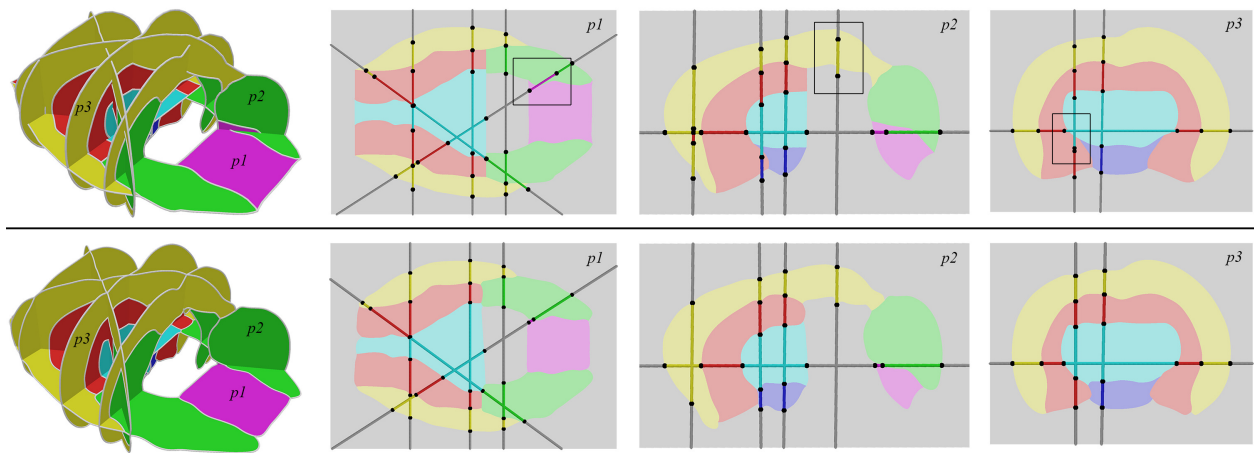


Figure 3.10: Result (bottom) of repairing an inconsistent 7-labelled mouse brain set (top), showing the labeling on three planes (p_1, p_2, p_3) and labeling on other planes along the intersection lines. A few inconsistencies are highlighted in black boxes.

3.6 Conclusion and discussion

In this paper, we consider the problem of solving label inconsistencies given contour networks on multi-labeled domains consisting of planar slices. We formulated the solution as a constrained optimization problem using an implicit representation where we carefully construct the energy function to preserve the shape of the contour while eliminating inconsistencies. We presented a targeted solver which exceeds the performance of tuned general solvers for this same problem. Our algorithm solves a critical step in the reconstruction pipeline from cross-sections, and it is our hope that this method will pave the way for existing surfacing algorithms to reach a wide spread use in the scientific, medical, and design communities.

Limitations and future work Our work can be improved and extended in several ways. First, the deformation energy used in our optimization formulation captures the distortion to the input curve network in terms of its location and tangents. However, it does not explicitly preserve the smoothness of, or any sharp features on, the input curves. Augmenting the energy with higher-order terms has the potential to more faithfully retain the curve shape. Second, while label consistency is sufficient for reconstructing a continuous surface, reconstructing a *smooth* surface places stronger requirement on the input curve network, such as the differential properties where curves on different planes intersect. The precise consistency condition for smooth reconstruction, and how to enforce them, invite further investigation. Lastly, we would like to explore strategies to further speed up the optimization process, so that it may be used within an interactive volume segmentation program to give immediate feedback to the user as she delineates the boundary curves.

Chapter 4

Variational Implicit Point Set Surfaces

4.1 Introduction

Constructing a curve or surface that interpolates or approximates a given set of 2D or 3D points is one of the fundamental problems in geometric modeling. A common representation of the reconstructed output is the zero-level set of some smooth implicit function. This representation naturally ensures a smooth and closed manifold. In addition, an implicit function enables a range of applications such as boolean operations and collision detection.

While extensively studied [17], implicit modeling from points remains a difficult problem. A fundamental challenge is that the constant zero function, although meaningless for reconstruction, perfectly meets our goal: the function is smooth and its zero-level set (which includes the entire space) interpolates any input points. A common approach to avoid this trivial solution is to introduce additional constraints, such as normals at the input points or additional spatial locations with inside/outside labels. If such constraints are not available as

part of the input or provided by the user, they will need to be inferred from the input data *prior to* reconstruction.

This two-stage paradigm - constraint estimation followed by reconstruction - has a number of drawbacks. The use of multiple disparate methods, each carrying its own set of parameters, makes parameter-tuning trickier and complicates the analysis (e.g., how the output changes with the input). More importantly, methods for constraint estimation are completely unaware of the quality of the reconstructed surface. Lacking any better guidance, current estimation methods (e.g., for normals) rely on local point neighborhoods, which are often unreliable when the points are sparse or non-uniformly distributed. Errors in constraint estimation, in turn, lead to poorly reconstructed surfaces (e.g., Figure 4.1 (e,f)).

We propose a direct definition of an implicit function from an un-oriented point set. Unlike the above-mentioned two-stage paradigm, our definition integrates constraint estimation and surface reconstruction within a single variational formulation. Specifically, *we seek a smooth implicit function whose zero-level set is close to the input points and whose gradient at each input point has unit magnitude*. The unit-gradient constraint avoids the trivial solution of zero but does not need to be estimated in a separate process. By a judicious choice of the smoothness energy [56], we show that our definition can be expressed as a standard constrained quadratic optimization problem with closed-form coefficients. The variational problem can be solved using off-the-shelf optimization packages without the need to discretize the domain.

Our method has a number of advantages over the previous two-stage paradigm. First, as an explicit definition, we can analyze properties of the reconstruction as a function of the input points. In this paper, we show that the surface given by our definition reproduces linear geometry and commutes with similarity transformations (translation, rotation, uniform

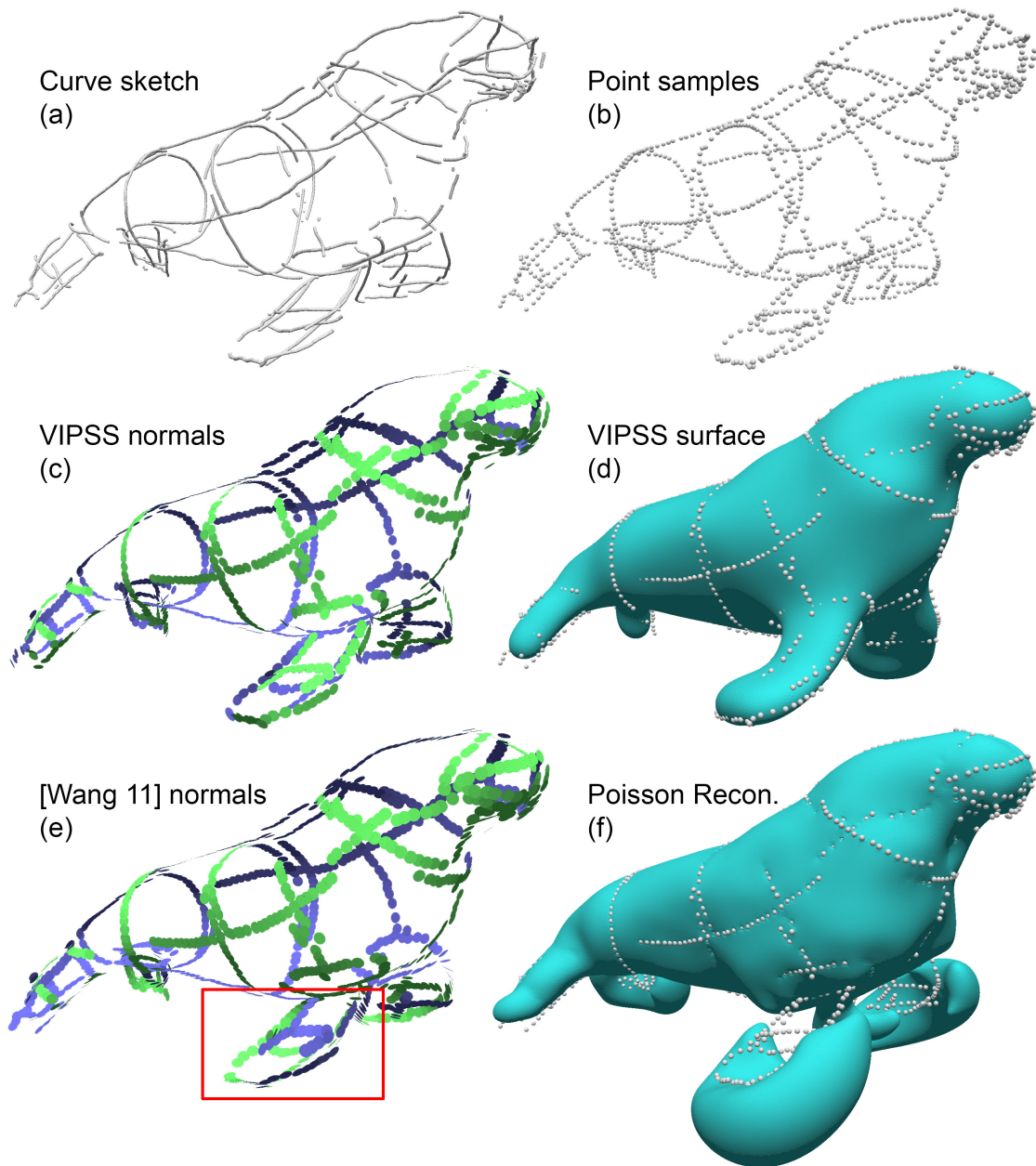


Figure 4.1: Given sparse, non-uniform, noisy and un-oriented points (b) sampled from a set of unstructured 3D curves (a), our variational definition (VIPSS with $\lambda = 0.003$) simultaneously produces oriented normals (c) and a smooth approximating surface (d). The input is challenging for state-of-the-art normal estimation methods such as [154], which fails around sparsely sampled thin features (the flippers) (e). Incorrect normals lead to poor reconstructions using existing implicit methods such as Screened Poisson [86] (f, fitting weight $\alpha = 0.5$).

scaling). Second, our definition involves only a single parameter (λ) that controls the accuracy of approximation. If exact interpolation is desired, reconstruction is completely parameter-free for any point set by setting $\lambda = 0$. Third, we observed that the surfaces produced by our definition are much more resilient to sparse or non-uniform samples than existing methods. We owe such robustness to our integrated formulation and the chosen smoothness energy [56], which considers the global shape of the reconstructed implicit function instead of local neighborhood of points.

The main limitation of our method is its computational complexity, which scales cubically with the number of points. While we are actively exploring means to improve scalability, we demonstrate the usefulness of our method in one application (surfacing unstructured 3D sketches) which result in sparse, non-uniform, un-oriented samples that are challenging for existing reconstruction methods (Figure 4.1).

Contributions We make the following technical contributions:

- We introduce a variational definition of an implicit surface directly from un-oriented points. The definition has a single parameter, applies to any dimensions, and does not need domain discretization or numerical integration (Section 4.3).
- We show several theoretical properties of the definition, including exact interpolation, reproducing linear geometry, and commuting with similarity transformations (Appendix A).
- We propose an effective strategy for initializing the optimization (Section 4.4.1).

4.2 Related Works

4.2.1 Surface reconstruction from points

We review the most relevant methods for surface reconstruction from 3D point sets. For more in-depth and comprehensive discussions of these and other methods, we refer readers to the latest survey [17].

Combinatorial methods

One class of reconstruction algorithms directly produce a triangulated surface whose vertices are the input points. These methods are typically based on the Delaunay triangulation of the points or its dual, the Voronoi Diagram (see survey [48]). Compared with implicit methods, the smoothness of the surfaces created by combinatorial methods is limited by the input sampling density. Also, a water-tight surface is not always guaranteed. Furthermore, as these methods usually work under the assumption that the input samples are dense enough with respect to the local shape, sparse or non-uniform sampling may lead to significantly degraded reconstruction quality (see Figure 4.12).

Implicit methods

We broadly classify implicit reconstruction methods into three types: ones that require “signed” input (e.g., oriented normals or additional labelled points), ones that perform signing in a post-process, and ones that do neither.

Requiring signed input Most implicit methods require the input points to be equipped with oriented normals. The Poisson reconstruction method [85] and its variants [99, 143, 115, 86] seek an “indicator function” that is 1 (resp. 0) in the interior (resp. exterior) of the shape and whose gradient near the shape’s boundary agrees with the given normals. The implicit moving least squares (IMLS) method defines a local polynomial, such as a constant [135, 52, 90, 112] or an algebraic sphere [66], at each spatial location that fits nearby samples and normals. Note that these methods are different from another group of methods (also called moving-least squares) that employ a projection operator [4, 92, 61], whose result is generally not a level set of an implicit function [8]. IMLS using a constant polynomial effectively blends linear functions defined by the tangent planes at the input points, which is also used in [72, 27]. More generally, the partition-of-unity method [109] blends polynomials that are fitted to groups of points.

Another tool for implicit reconstruction from signed input is radial basis functions (RBF). RBF interpolants for scattered data have been extensively studied in the literature [39, 155]. Unlike IMLS and partition-of-unity, an RBF interpolant is a linear combination of *fixed* radial basis kernels (typically centered at the input points) blended with a *fixed* set of weights chosen so that the given values at the input points are interpolated. To avoid the trivial interpolant of constant zero, most reconstruction methods that use RBF introduce additional spatial locations equipped with signed values [147, 106, 42, 149, 54, 110, 140, 124, 153]. These locations are often created by offsetting along a sample’s normal. However, care must be taken in determining the offset amount, particularly near thin features. Alternatively, some authors [36, 65, 79, 95] apply a Hermite variant of the RBF interpolant [56] directly to the input points and normals.

All methods in this class interpolate or closely approximate the normals at the input points. As a result, their reconstruction quality relies heavily on the accuracy of normal estimation.

In this work, we use the Hermite RBF interpolant of [56] (which we call Duchon’s interpolant and will discuss in details in Section 4.3.2). However, our variational formulation removes the need for estimating the normals.

Signing in a post-process A number of methods first create an un-signed distance function from the input points and then determine the sign afterwards. Various strategies were used for signing, including graph-cut [73], ray-shooting [107], watershed [119], and energy minimization [64]. While these methods can be applied directly to un-oriented points, the heuristic nature of the signing step makes it difficult to analyze the results of these methods. Also, the two-stage pipeline involves several parameters that need to be carefully tuned.

Variational methods Like our method, several other methods directly reconstruct a signed function from un-oriented points using some variational formulation. After computing un-oriented normals at the samples, Alliez et al. [5] seek an implicit function whose gradients best align with the un-oriented normals under the constraint that the weighted sum of biharmonic energy and fitting error has unit norm. Both Scholkopf et al. [130] and Walder et al. [152] propose un-constrained formulations using scalar (non-Hermite) RBF interpolants. Scholkopf et al. minimize a weighted sum of smoothness energy, fitting error, and the negative constant component of the interpolant. Walder et al. point out that Scholkopf’s formulation leads to ill-shaped functions. They replace the last component of Scholkopf’s objective by another two terms, the negative integral of the function values and of the gradient magnitudes over the domain.

The above formulations all involve multiple parameters for balancing the various energy terms and/or constraints. Also, solving for these variational problems may require domain discretization [5] or numerical integration [152], which introduces additional parameters as well as dependencies on the discretization structure or resolution. In contrast, our definition

has a single parameter (which is fixed in the interpolation mode), and no discretization is needed for optimization. We also observed that our formulation tends to behave more robustly under sparse sampling than [5] (see Figure 4.12).

4.2.2 Normal estimation

As mentioned above, normal estimation is required by many implicit reconstruction methods. Existing estimation methods can be classified into two types, ones that separately estimate the direction and orientation (i.e., forward or backward) of normals, and ones that estimate both at once.

Estimation and orientation of un-oriented normals A common strategy for estimating un-oriented normal directions is by fitting the local neighborhood of a point with a function. Linear functions are most common [72] (also known as the PCA method), but higher-order polynomials have also been used [43, 66]. It is well-known that the choice of the point neighborhood is crucial to the accuracy of estimation, and various proposals were made to deal robustly with noisy samples [116, 104] and sharp features [93, 33, 96]. Another strategy is based on analyzing the shape of individual [6, 50] or a group of [5, 100] Voronoi cells. While many of these methods can successfully handle noisy samples, they tend to fail when the sampling rate is low or the pattern is not uniform (see Figure 4.9).

Orienting the un-oriented normals can be treated as a combinatorial optimization problem. The choice of energy varies from simple consistency among neighboring normals [72] to more sophisticated ones that better handle thin features [157, 91, 74]. The energy can be minimized by a minimal spanning tree [72] or a global solver [127]. However, if the un-oriented normal has a wrong direction (e.g., orthogonal to the true normal), it cannot be corrected by the orientation step.

Direct estimation of oriented normals Wang et al. [154] proposed a variational method that estimates both the direction and orientation of normals in a single step. They formulate a quadratic optimization problem that minimizes the weighted sum of two energy terms on each pair of nearby normals, a consistency energy similar to [72] and an orthogonality energy. In our experiments, we found this method outperforms the two-step methods mentioned above for sparse and non-uniform samples, but tuning its parameters can be challenging (see Section 4.5.2 and Figure 4.10). While our definition solves a similar quadratic optimization problem, our objective captures the global regularity of the implicit function, which leads to more robust results while removing the need for parameter tuning.

Recently, deep learning has been employed to infer oriented normals from point clouds [34, 67]. These methods can produce impressive results on densely sampled points contaminated with noise. However, we found that they are less successful on sparse samples (see Figure 4.10).

4.3 Definition

We introduce our definition of the point set surface in this section. We first give a general definition for any choice of smoothness energy, and then specialize it to a particular energy that leads to a simple and computable definition.

Notations In this paper, scalar values are italicized (e.g., x), vector values are bold (e.g., \mathbf{x}), and matrices are capitalized (e.g., M). All vectors are assumed to be column vectors. We use D^i to denote taking the i -th derivative (or gradient, if the variable is a vector), and $D = D^1$. For a two variable function $f(\cdot, \cdot)$, we use $D^{i;j}$ to denote taking the i -th partial derivative of the first variable and j -th partial derivative of the second variable. $\|\cdot\|$ denotes the L_2 norm.

4.3.1 A general definition

Given a set of points $\mathbf{x}_i \in \mathbb{R}^d$ ($i = 1, \dots, n$), we wish to define a smooth implicit function $f(\mathbf{x})$ whose zero-level set is as close to \mathbf{x}_i as possible. As mentioned earlier, to avoid the trivial solution $f \equiv 0$, we need an additional constraint. To this end, we note that existing implicit modeling methods either look for a true signed distance function, or an “indicator function” that is close to a signed distance function near the shape boundary. In the same spirit, we require that the gradient of f should have *unit magnitude* at each input point. Unlike signed constraints, such as normals or labelled points, the unit-gradient constraint does not need to be provided or estimated *a priori*. Note that the same constraint has been used previously for fitting implicit functions [121].

We arrive at the following variational definition of an implicit point set surface:

Definition 1. *The variational implicit point set surface (VIPSS) of points $\mathbf{x}_i \in \mathbb{R}^d$, for a chosen energy E and approximation parameter λ , is the zero-level set of a function $f \in \mathbb{R}^d$ that*

$$\begin{aligned} \text{Minimizes:} \quad & \sum_i f(\mathbf{x}_i)^2 + \lambda E(f) \\ \text{Subject to:} \quad & \|Df(\mathbf{x}_i)\| = 1, \quad \forall i \end{aligned} \tag{4.1}$$

Here, λ balances the two goals of data fitting and smoothness. A larger λ leads to a smoother surface at the cost of a less accurate approximation of the input points.

A key ingredient of this definition is the energy E , which needs to be appropriately chosen for the definition to be meaningful and practical. On one hand, lower E values should correspond to a smoother zero-level set of f . This requires an energy of at least 2nd-order, so that linear functions (whose level sets are hyperplanes, which are perfectly smooth) have zero energy.

On the other hand, the energy should make the variational problem computable, and ideally not requiring any discretization of the domain.

We describe one class of energy E that makes Definition 1 computation-friendly without discretization. Suppose we are given an additional set of Hermite data $\mathbf{s} = \{s_i\}$ and $\mathbf{g} = \{\mathbf{g}_i\}$, so that each point \mathbf{x}_i is equipped with a scalar s_i and a (possibly un-normalized) vector \mathbf{g}_i . Let $f_{\mathbf{s},\mathbf{g}}$ be the interpolating function with minimal energy,

$$f_{\mathbf{s},\mathbf{g}} = \arg \min_f \{E(f) \mid f(\mathbf{x}_i) = s_i, Df(\mathbf{x}_i) = \mathbf{g}_i, \forall i\}$$

The solution to the constrained optimization problem of (4.1) is therefore the interpolant $f_{\mathbf{s},\mathbf{g}}$ for the Hermite data $\{\mathbf{s}, \mathbf{g}\}$ that

$$\begin{aligned} \text{Minimizes:} \quad & \mathbf{s}^T \mathbf{s} + \lambda E(f_{\mathbf{s},\mathbf{g}}) \\ \text{Subject to:} \quad & \mathbf{g}_i^T \mathbf{g}_i = 1, \quad \forall i \end{aligned} \tag{4.2}$$

Conceptually, while $f_{\mathbf{s},\mathbf{g}}$ smoothly interpolates a *given* Hermite data $\{\mathbf{s}, \mathbf{g}\}$, we look for the *best* Hermite data, with unit vectors \mathbf{g} , that results in the smoothest interpolant. If energy E is chosen such that the energy-minimizing Hermite interpolant $f_{\mathbf{s},\mathbf{g}}$, as well as its energy $E(f_{\mathbf{s},\mathbf{g}})$, can be expressed in closed-form by \mathbf{s} and \mathbf{g} , then (4.2) becomes a constrained optimization problem on a finite variable set (\mathbf{s}, \mathbf{g}) .

Next, we review one choice of E that has the above characteristics, namely being 2nd-order and the energy-minimizing Hermite interpolant has closed form (Section 4.3.2). With this choice, we will show that Definition 1 becomes a constrained quadratic optimization problem (Section 4.3.3).

4.3.2 Duchon’s energy

Duchon [56] studied a family of semi-norms whose minimizers, subject to interpolatory conditions, have simple and closed forms. In the context of this work, we are interested in the following 2nd-order member of the family ²:

$$E(f) = \int_{\mathbb{R}^d} \|\tau\|^{d-1} \|\mathfrak{F}D^2 f(\tau)\|^2 d\tau \quad (4.3)$$

where \mathfrak{F} denotes the Fourier transform. The energy is a generalization of the “thin-plate” or “bending” energy in one dimension ($d = 1$), where

$$E(f) = \int_{\mathbb{R}} \|\mathfrak{F}D^2 f(\tau)\|^2 d\tau = \int_{\mathbb{R}} \|D^2 f(x)\|^2 dx$$

Duchon showed that the function that interpolates Hermite data $\{s_i, \mathbf{g}_i\}$ at each input point \mathbf{x}_i with the minimal energy has the following form:

$$f_{\mathbf{s}, \mathbf{g}}(\mathbf{x}) = \sum_i a_i \phi(\mathbf{x}, \mathbf{x}_i) + \sum_i \mathbf{b}_i^T D^{0,1} \phi(\mathbf{x}, \mathbf{x}_i) + \mathbf{c}^T \mathbf{x} + d \quad (4.4)$$

where ϕ is the triharmonic radial basis kernel ($\phi(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^3$), and $a_i \in \mathbb{R}$, $\mathbf{b}_i \in \mathbb{R}^d$, $\mathbf{c} \in \mathbb{R}^d$, $d \in \mathbb{R}$ are constants determined from the input data. Specifically, these constants need to satisfy the interpolatory conditions ($f_{\mathbf{s}, \mathbf{g}}(\mathbf{x}_i) = s_i$, $Df_{\mathbf{s}, \mathbf{g}}(\mathbf{x}_i) = \mathbf{g}_i$ for all i) and additional orthogonality conditions including $\sum_i a_i = 0$ and $\sum_i a_i \mathbf{x}_i + \sum_i \mathbf{b}_i = 0$ to ensure the existence

²Duchon’s original semi-norms take an additional square-root, which we drop in this work

of a unique solution. All of these conditions can be expressed by a system of linear equations

$$A \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ d \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \\ \mathbf{0} \\ 0 \end{pmatrix} \quad (4.5)$$

where $\mathbf{a} = \{a_i\}$, \mathbf{b} is the flattened array of $\{\mathbf{b}_i\}$ of length $d \times n$, and \mathbf{g} is the flattened array of \mathbf{g}_i of length $d \times n$. The coefficient matrix A of this system (often called the *interpolation matrix*) has the form

$$A = \begin{pmatrix} M & N \\ N^T & \mathbf{0} \end{pmatrix}, \quad M = \begin{pmatrix} M_{00} & M_{01} \\ M_{01}^T & M_{11} \end{pmatrix}, \quad N = \begin{pmatrix} N_0 & \mathbf{1} \\ N_1 & \mathbf{0} \end{pmatrix} \quad (4.6)$$

Here, the matrices M_{00} , M_{01} , M_{11} have dimensions $n \times n$, $n \times dn$, and $dn \times dn$ respectively. Each (i, j) -th entry (or block) of matrix $M_{\alpha\beta}$ where $\alpha, \beta \in \{0, 1\}$ is the differential $D^{\alpha,\beta}\phi(\mathbf{x}_i, \mathbf{x}_j)$. Note that both M_{00} , M_{11} are symmetric matrices. N_0 has dimension $n \times d$ and its i -th row is \mathbf{x}_i^T . N_1 has dimension $dn \times d$ and consists of n identity matrices of dimension $d \times d$.

Assuming that the points \mathbf{x}_i are pairwise disjoint, the matrix A is always invertible and hence the constants $\mathbf{a}, \mathbf{b}, \mathbf{c}, d$ satisfying (4.5) uniquely exist [155]. In one dimension ($d = 1$), the resulting interpolant $f_{\mathbf{s}, \mathbf{g}}$ coincides with the ordinary piecewise cubic Hermite interpolation. In this sense, Duchon's interpolant can be considered as a generalization of cubic Hermite interpolation to arbitrary dimensions. Figure 4.2 (a) gives examples of the interpolant in 1D and 2D (with $s_i = 0$ and \mathbf{g}_i set to a constant for all i).

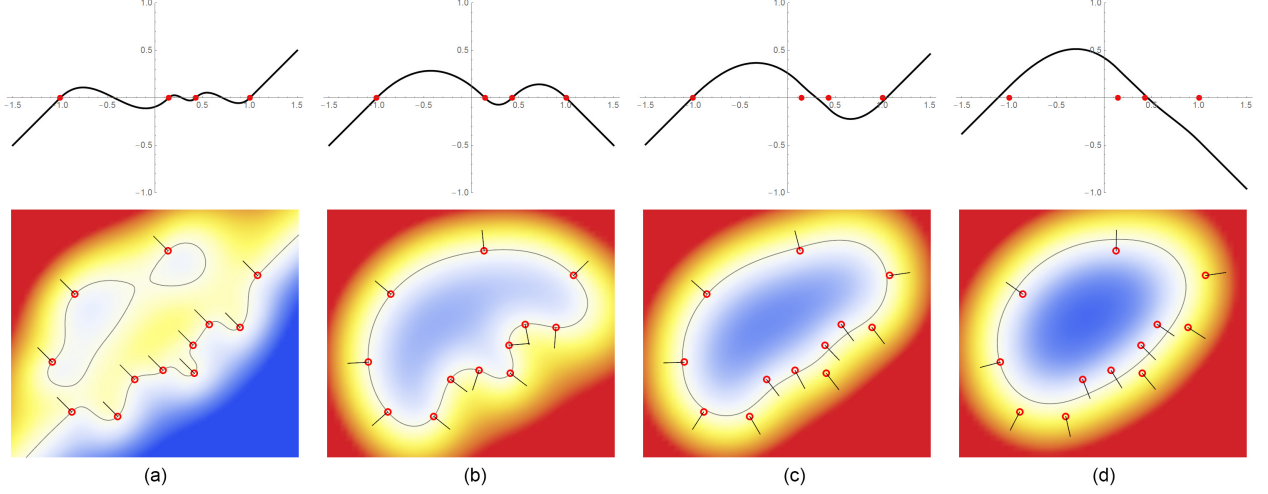


Figure 4.2: Examples of Duchon's interpolants $f_{\mathbf{s},\mathbf{g}}$ that interpolate scattered points in 1D (top, red dots) and 2D (bottom, red circles) for different choices of the Hermite data $\{\mathbf{s}, \mathbf{g}\}$. In (a), $s_i = 0$ and \mathbf{g}_i is a constant vector at each point. In (b,c,d), $\{\mathbf{s}, \mathbf{g}\}$ are obtained by our variational formulation (4.9) with $\lambda = 0, 0.1, 1.0$ respectively. The zero-level set in (b,c,d) (black curves) is the VIPSS at the respective λ .

The energy of Duchon's interpolant can also be written in a simple closed form. Since E is the semi-norm of a semi-Hilbert space whose reproducing kernel is the triharmonic radial basis ϕ , the semi-norm of $f_{\mathbf{s},\mathbf{g}}$ can be expressed as

$$E(f_{\mathbf{s},\mathbf{g}}) = \begin{pmatrix} \mathbf{a}^T & \mathbf{b}^T \end{pmatrix} M \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \quad (4.7)$$

4.3.3 Definition using Duchon's energy

We now specialize the VIPSS definition (1) to the case where E is chosen as Duchon's energy (4.3). In essence, we will look for the scalars \mathbf{s} and unit vectors \mathbf{g} such that the Duchon's interpolant $f_{\mathbf{s},\mathbf{g}}$ minimizes the energy objective in (4.2).

We first write the inverse of the interpolation matrix A as

$$A^{-1} = \begin{pmatrix} J & K \\ K^T & L \end{pmatrix}, \quad J = \begin{pmatrix} J_{00} & J_{01} \\ J_{01}^T & J_{11} \end{pmatrix} \quad (4.8)$$

where matrices $J, K, J_{00}, J_{01}, J_{11}$ have the respective sizes as $M, N, M_{00}, M_{01}, M_{11}$ in (4.6), and J, L, J_{00}, J_{11} are symmetric. We now present our main result:

Proposition 1. *The VIPSS of points \mathbf{x}_i , where E is Duchon's energy defined in (4.3), is the zero-level set of Duchon's interpolant $f_{\mathbf{s}, \mathbf{g}}$ defined in (4.4) for Hermite data $\{\mathbf{s}, \mathbf{g}\}$, such that \mathbf{g}*

$$\text{Minimizes:} \quad \mathbf{g}^T H \mathbf{g} \quad (4.9)$$

$$\text{Subject to:} \quad \mathbf{g}_i^T \mathbf{g}_i = 1, \quad \forall i$$

where

$$H = J_{11} - \lambda J_{01}^T (I + \lambda J_{00})^{-1} J_{01} \quad (4.10)$$

and \mathbf{s} is obtained from \mathbf{g} by

$$\mathbf{s} = -\lambda (I + \lambda J_{00})^{-1} J_{01} \mathbf{g} \quad (4.11)$$

Proof. We start by re-writing the energy of Duchon's interpolant $E(f_{\mathbf{s}, \mathbf{g}})$ as a function of the Hermite data $\{\mathbf{s}, \mathbf{g}\}$. Using notations in (4.8),

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = J \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \end{pmatrix} \quad (4.12)$$

Substituting into (4.7) yields

$$E(f_{\mathbf{s},\mathbf{g}}) = \begin{pmatrix} \mathbf{s}^T & \mathbf{g}^T \end{pmatrix} J M J \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \end{pmatrix} \quad (4.13)$$

To simplify this expression further, note that by $A^{-1}A = I$ we have $J M + K N^T = I$ and $J N = \mathbf{0}$. Therefore,

$$\begin{aligned} J M J &= (I - K N^T) J \\ &= J - K N^T J \\ &= J - K (J N)^T \\ &= J \end{aligned}$$

which leads to

$$E(f_{\mathbf{s},\mathbf{g}}) = \begin{pmatrix} \mathbf{s}^T & \mathbf{g}^T \end{pmatrix} J \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \end{pmatrix} \quad (4.14)$$

Using the expression of $E(f_{\mathbf{s},\mathbf{g}})$ above, the minimization objective in (4.2) becomes a quadratic function on \mathbf{s} and \mathbf{g} :

$$\mathbf{s}^T \mathbf{s} + \lambda \begin{pmatrix} \mathbf{s}^T & \mathbf{g}^T \end{pmatrix} J \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \end{pmatrix} \quad (4.15)$$

For a given \mathbf{g} , (4.15) is minimized when \mathbf{s} is given by (4.11), and the minimum has the form $\lambda \mathbf{g}^T H \mathbf{g}$ where H is defined in (4.10). By dropping the constant λ , we have proven the proposition. \square

We add two technical notes here. First, matrix $I + \lambda J_{00}$ is invertible for generic values of λ , as long as $-1/\lambda$ is not an eigenvalue of J_{00} . Second, H is a positive semi-definite matrix,

since $\lambda \mathbf{g}^T H \mathbf{g}$ is the sum of $\mathbf{s}^T \mathbf{s}$ and Duchon's energy of $f_{\mathbf{s}, \mathbf{g}}$, both of which are non-negative for any choice of \mathbf{g} .

One may observe that Proposition 1 transforms the problem of finding an implicit function (4.1) to that of finding unit vectors (normals) \mathbf{g} at the input points (4.9). From this perspective, our definition offers another way of estimating normals from un-oriented points. The key distinction between our formulation and existing normal estimation approaches is that, while the latter is guided by the local shape of point samples, ours is guided by the global shape of the reconstructed implicit function (via minimizing Duchon's energy). As we shall see in the experimental results, the coupling of normal estimation with reconstruction allows our method to better deal with imperfect sampling than previous methods.

Figure 4.2 (b,c,d) give examples of the VIPSS specialized to Duchon's energy with varying values of λ (0, 0.1, 1) in both 1D and 2D. Observe that the VIPSS in 2D is able to interpolate or approximate sparse and non-uniformly distributed points. Increasing λ results in smoother curves that deviate further from the input. At $\lambda = 0$, the VIPSS exactly interpolates the points (a property that will be discussed in the next section).

4.4 Implementation

Reconstructing the VIPSS from a point set involves four steps:

1. Computing matrix H . This involves constructing the interpolation matrix A by (4.5), computing its inverse A^{-1} , and computing H from sub-matrices of A^{-1} by (4.10).
2. Optimizing vectors \mathbf{g} by (4.9).

3. Constructing Duchon’s interpolant $f_{\mathbf{s},\mathbf{g}}$. This involves recovering interpolated values \mathbf{s} from \mathbf{g} by (4.11) and then the constants $\mathbf{a}, \mathbf{b}, \mathbf{c}, d$ in $f_{\mathbf{s},\mathbf{g}}$ by $A^{-1} \cdot \{\mathbf{s}^T, \mathbf{g}^T, \mathbf{0}^T, 0\}^T$.
4. Surfacing the zero-level set of $f_{\mathbf{s},\mathbf{g}}$.

Steps (1,3) can be done using standard linear algebra packages (we use Armadillo). To solve the constrained optimization problem of (4.9), we first convert it into an unconstrained problem by representing each \mathbf{g}_i using two spherical angles and then solve it by L-BFGS. For step (4), one may use any available method that polygonalizes level sets of implicit functions. Since the input points are usually close to the zero-level surface, we use a tracing-based marching cubes method [23] at a fixed grid resolution (100^3 to 200^3 in our experiments) that starts from a data point. More advanced meshing methods, such as [29], can be applied to produce surfaces with better triangle shapes.

As with many non-linear optimization problems, the quality of the solution depends heavily on the quality of the initialization. In the following (Section 4.4.1), we describe a practically effective method for initializing our optimization problem. We end this section with a complexity analysis (Section 4.4.2).

4.4.1 Initializing the optimization

Optimization formulations like ours (4.9) commonly appear in the literature on computing direction fields [154, 89, 80, 76]. A typical initialization strategy is relaxing the per-vector unit-norm constraint ($\mathbf{g}_i^T \mathbf{g}_i = 1$) to constraining the total norm of all vectors to be one ($\mathbf{g}^T \mathbf{g} = 1$). By the Rayleigh Quotient Theorem, the minimal value of $\mathbf{g}^T H \mathbf{g}$ under the relaxed constraint is achieved when \mathbf{g} is the eigenvector of H with the smallest eigenvalue (denoted

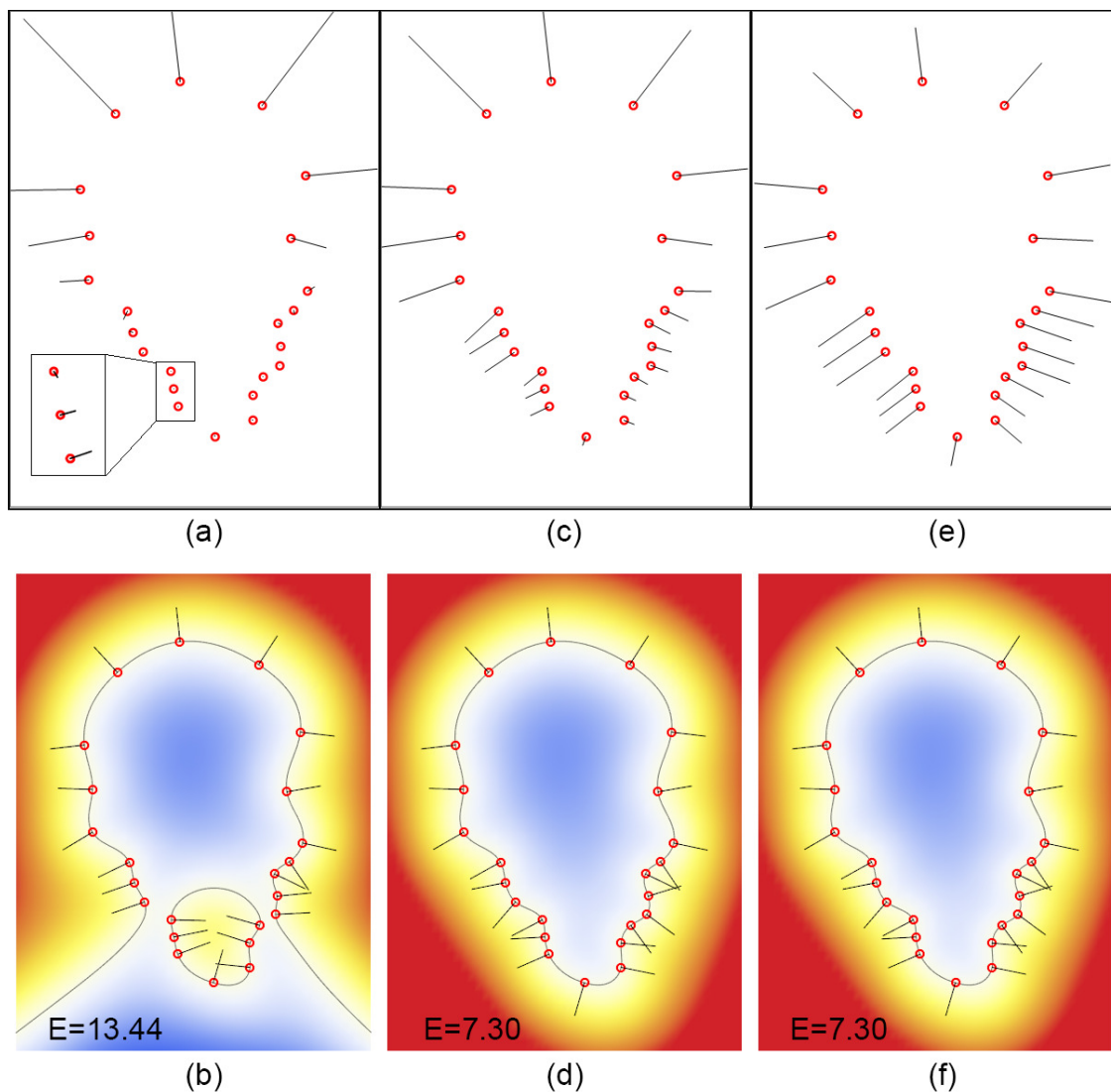


Figure 4.3: Initial vectors generated using the spectral method with $\lambda = 0$ (a) contains vectors with very small magnitudes and flipped orientations (see insert, vectors shown with 20x scaling), which leads to a high-energy result after optimization (b). Initial vectors generated with $\lambda = 0.01$ (c) and 0.1 (e) are more uniform, and they lead to the same low-energy result (d,f) after optimization with $\lambda = 0$.

by e_H). This eigenvector, after normalization, is then used as the initial solution to start the optimization under the original, per-vector unit-norm constraint.

However, the spectral initialization strategy often fails for our problem. In these failure cases, there is a significant variation among the norms of individual vectors $\|\mathbf{g}_i\|$ derived from e_H .

Those vectors with extremely small norms tend to be less accurate, and such vectors often form clusters with “flipped” orientations that are difficult to be corrected by optimization. We illustrate such a case in 2D in Figure 4.3 (a,b). Note that the initial vectors associated with the lower points in (a) have very small lengths, and their orientations are opposite to those vectors associated with the upper points (see zoom-in). Optimizing from this initialization leads to a high-energy local minimum shown in (b).

To find more stable initial vectors, we observed that the norms of individual vectors $\|\mathbf{g}_i\|$ derived from e_H tend to be more uniformly distributed, both in length and in direction, as λ increases. This can be conceptually explained by the fact that a larger λ leads to a smoother interpolant, whose gradients at the input points are more similar with each other. We can also provide a more rigorous argument in the limiting case of $\lambda \rightarrow \infty$:

Proposition 2. *As $\lambda \rightarrow \infty$, any $\mathbf{g} = \{\mathbf{g}_i\}$ where $\mathbf{g}_i = \mathbf{g}_j$ for all pairs i, j is an eigenvector of H with zero eigenvalue.*

Proof. By $JN = \mathbf{0}$, where J is defined in (4.8) and N in (4.6), we have

$$J_{00} N_0 + J_{01} N_1 = \mathbf{0}$$

$$J_{01}^T N_0 + J_{11} N_1 = \mathbf{0}$$

We then have

$$\begin{aligned} H N_1 &= J_{11} N_1 - J_{01}^T (I/\lambda + J_{00})^{-1} J_{01} N_1 \\ &= -J_{01}^T N_0 + J_{01}^T (I/\lambda + J_{00})^{-1} J_{00} N_0 \end{aligned}$$

As $\lambda \rightarrow \infty$, $(I/\lambda + J_{00})^{-1} J_{00} \rightarrow I$. Therefore

$$H N_1 \rightarrow -J_{01}^T N_0 + J_{01}^T N_0 = 0$$

As a result, the columns of N_1 are eigenvectors of H with zero eigenvalues as $\lambda \rightarrow \infty$. Since the \mathbf{g} in the proposition can be expressed as a linear combination of the columns of N_1 , it is also an eigenvector of H with zero eigenvalue. \square

In other words, the output of spectral initialization tends toward a constant vector field as $\lambda \rightarrow \infty$. Hence it can be expected that, as λ increases, vectors produced by spectral initialization become increasingly more uniform. We illustrate this behavior in Figure 4.3 (c,e) for two different values of λ (0.01, 0.1) on the same point set as (a). Using either set of vectors as the initial solution, optimizing (4.9) with $\lambda = 0$ successfully reaches the same low-energy solution shown in (d,f).

Guided by the observation, we propose to compute not one, but multiple candidate initializations corresponding to different values of λ . We pre-define a set of “offset” values $\{\lambda_1, \dots, \lambda_k\}$. Let λ_0 be the parameter chosen by the user for the VIPSS. For each λ_i , we construct the matrix H using $\lambda = \lambda_0 + \lambda_i$ and compute its eigenvector e_H . This gives us k initial solutions. We then optimize (with $\lambda = \lambda_0$) for k times, each time starting from one of the initial solutions, and take the optimized result with the least energy. We use the offset values $\{0, 0.001, 0.01, 0.1, 1\}$ in our tests, assuming the data is scaled to fit in a $2 \times 2 \times 2$ cube.

4.4.2 Complexity analysis

The asymptotic complexity of running time, with respect to the dimensionality d and number of input points n , of each step is as follows:

1. $O(d^3n^3)$: $O(d^3n^2)$ for constructing A , $O(d^3n^3)$ for inverting A , and $O(n^3 + d^2n^3)$ for constructing H .

2. $O(d^3n^3)$: $O(d^3n^3)$ for computing the eigenvector of H with the smallest eigenvalue, and $O(ld^2n^2)$ for gradient-descent optimization, where l is the number of descent iterations (which is typically much smaller compared to n).
3. $O(d^2n^2)$: $O(dn^2)$ for computing \mathbf{s} , and $O(d^2n^2)$ for recovering the constants $\mathbf{a}, \mathbf{b}, \mathbf{c}, d$.
4. $O(mdn)$ where m is the number of points at which the surfacing algorithm evaluates the interpolant.

It is clear from the analysis that the bottlenecks of the algorithm are the inversion of matrix A and finding the eigenvector of matrix H , both taking $O(d^3n^3)$ time. Note that A is a dense matrix due to the global nature of the triharmonic basis. While we have observed that H often contains a large amount of close-to-zero entries, particularly for uniformly sampled points and a small value of λ , the sparsity tends to decrease with the increase of non-uniformity in sampling and noise level (which necessitates larger values of λ).

4.5 Experiments

We show experimental results of our method in 3D under varying sampling conditions and compare with relevant methods for normal estimation and surface reconstruction. We end this section with a performance report and an application. In our examples, uniform sampling from existing surfaces is generated using the Poisson-disk sampling method [45] implemented in MeshLab.

4.5.1 Results

We first evaluate our method under varying sampling densities and patterns using a synthetic Torus surface (level set of a degree-4 polynomial). We fix $\lambda = 0$ to perform exact interpolation.

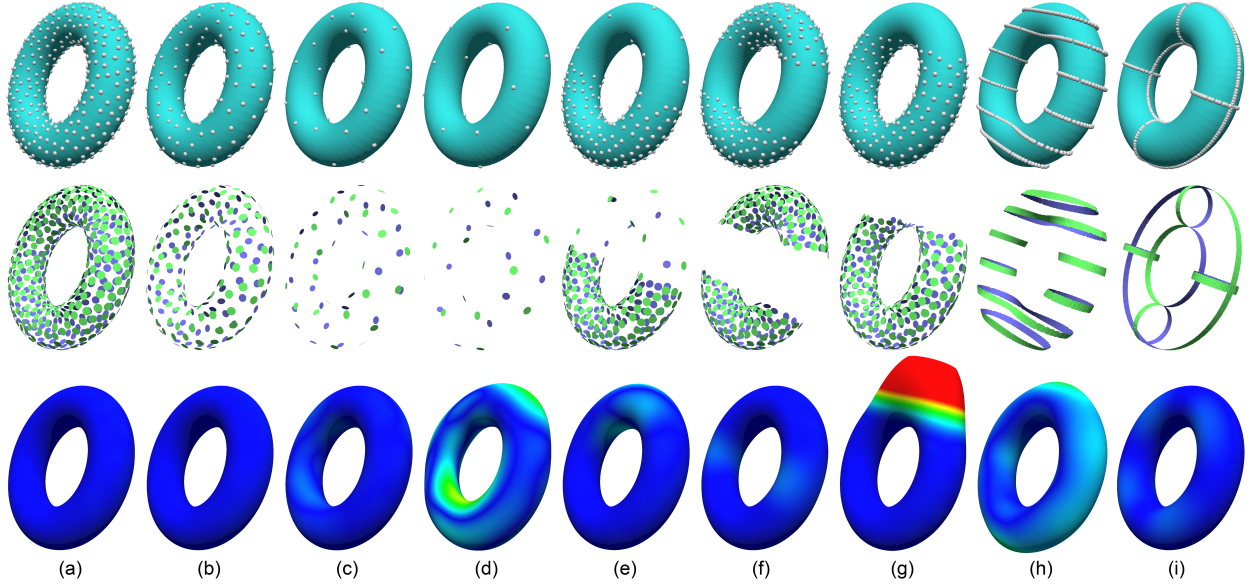


Figure 4.4: Top row: sampling a torus surface with decreasing density (a,b,c,d with 500, 200, 50, 25 points respectively), varying sampling density (e), missing samples (f,g), and along 1-dimensional curves (h,i). Middle row: optimized vectors \mathbf{g} visualized as oriented disks (green/blue: front/back side). Bottom row: the VIPSS ($\lambda = 0$) colored by distance from the original torus surface (blue/red: small/large distance).

As seen in Figure 4.4, our method can reconstruct an almost perfect torus from as few as 50 points (c), and a close approximation from 25 points (d). Also, our method is robust under different types of non-uniform sampling patterns, such as varying density (e), missing samples (f,g), and highly anisotropic sampling along curves (h,i), unless the samples are too ambiguous for inferring the shape (e.g., in (g)).

We next test our method on samples from more complex 3D surfaces (Figures 4.5, 4.6) and wireframes (Figure 4.7). Again, we fix $\lambda = 0$. Observe in Figure 4.5 that the quality of VIPSS drops gracefully under decreasing sampling density. Despite the sparsity and anisotropy of sampling in Figures 4.6 and 4.7, our method is able to faithfully reconstruct various shape features, such as the fingers of the Hand, protrusions of the Vertebra, ears of the Dog, etc.

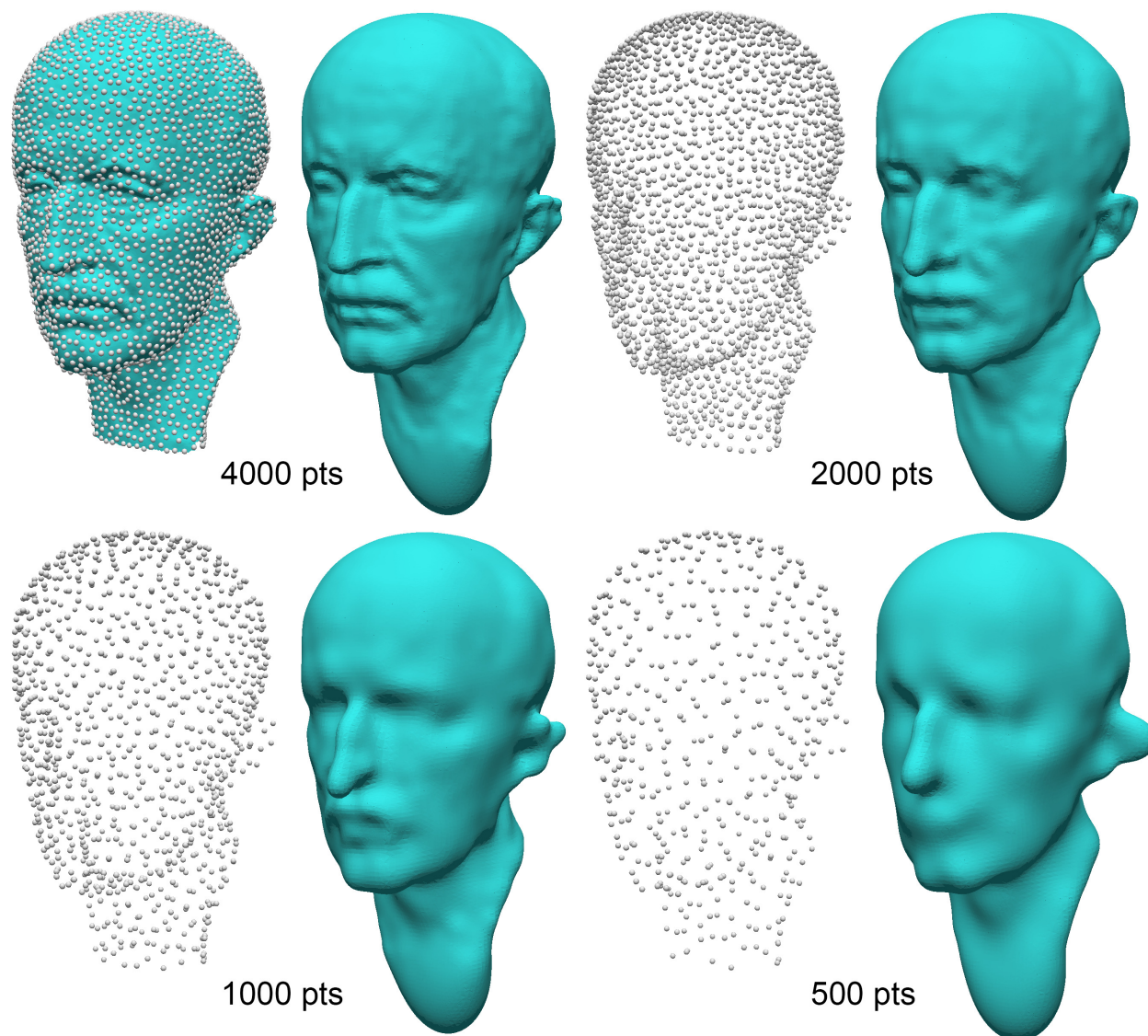


Figure 4.5: VIPSS ($\lambda = 0$) for samples from Max Planck at different densities.

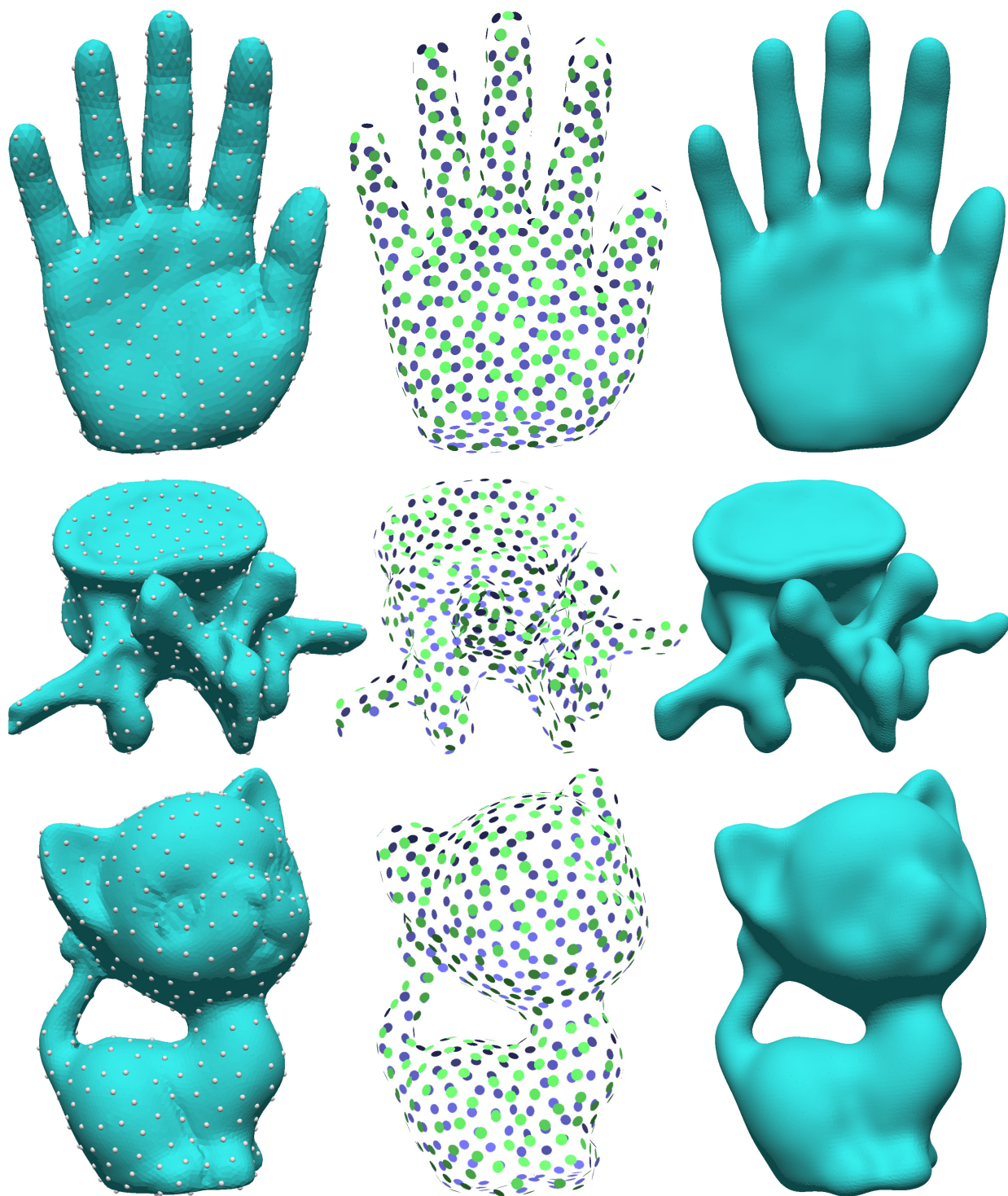


Figure 4.6: Samples from Hand, Vertebra, Kitten (left, each containing 500 points), optimized vectors \mathbf{g} (middle), and VIPSS (right, $\lambda = 0$).

Lastly, we test our method on noisy samples in Figure 4.8. Consistent with our observations earlier in 1D and 2D (Figure 4.2), larger λ values lead to smoother and less approximating surfaces, which are suited for higher noise levels.

4.5.2 Comparisons

Normal estimation methods We compare the oriented vectors \mathbf{g} resulted by our optimization (4.9) with those produced by existing normal estimation methods.

We first consider methods that separately estimate directions and orientations. Since errors in the un-oriented directions persist after orientation, we will focus on issues in the direction estimation step. We considered two prevalent methods for direction estimation, the plane-fitting method of [72] (referred to as PCA) and the Voronoi-based method of [100] (referred to as VCM). Both methods are based on analysis of local point neighborhood, whose size is controlled by the number of nearest neighbors (k in PCA) or a sphere radius (r in VCM). We found that they both tend to fail on non-uniformly distributed samples, where oftentimes a suitable neighborhood size cannot be found. As shown in Figure 4.9, taking the Dog wireframe samples from Figure 4.7 as input, PCA requires a fairly large neighborhood ($k = 30$ in (a)) to get a reasonable direction estimate for most points, while still failing on some (see the red and blue boxes). Increasing the neighborhood size ($k = 50$ in (b)) improves the directions at some points (the red box), but makes others worse (the blue box), due to the interference with nearby features (the dog ear). Similarly, VCM at a small neighborhood size ($r = 0.1$ in (c)) produces many incorrect directions (red box). A larger neighborhood ($r = 0.25$ in (d)) improves some directions but results in over-smoothing at small shape features (blue box).

Next we compare with the variational method of Wang et al. [Wang2011AVM], which estimates oriented normals in a single step by solving a quadratic optimization problem

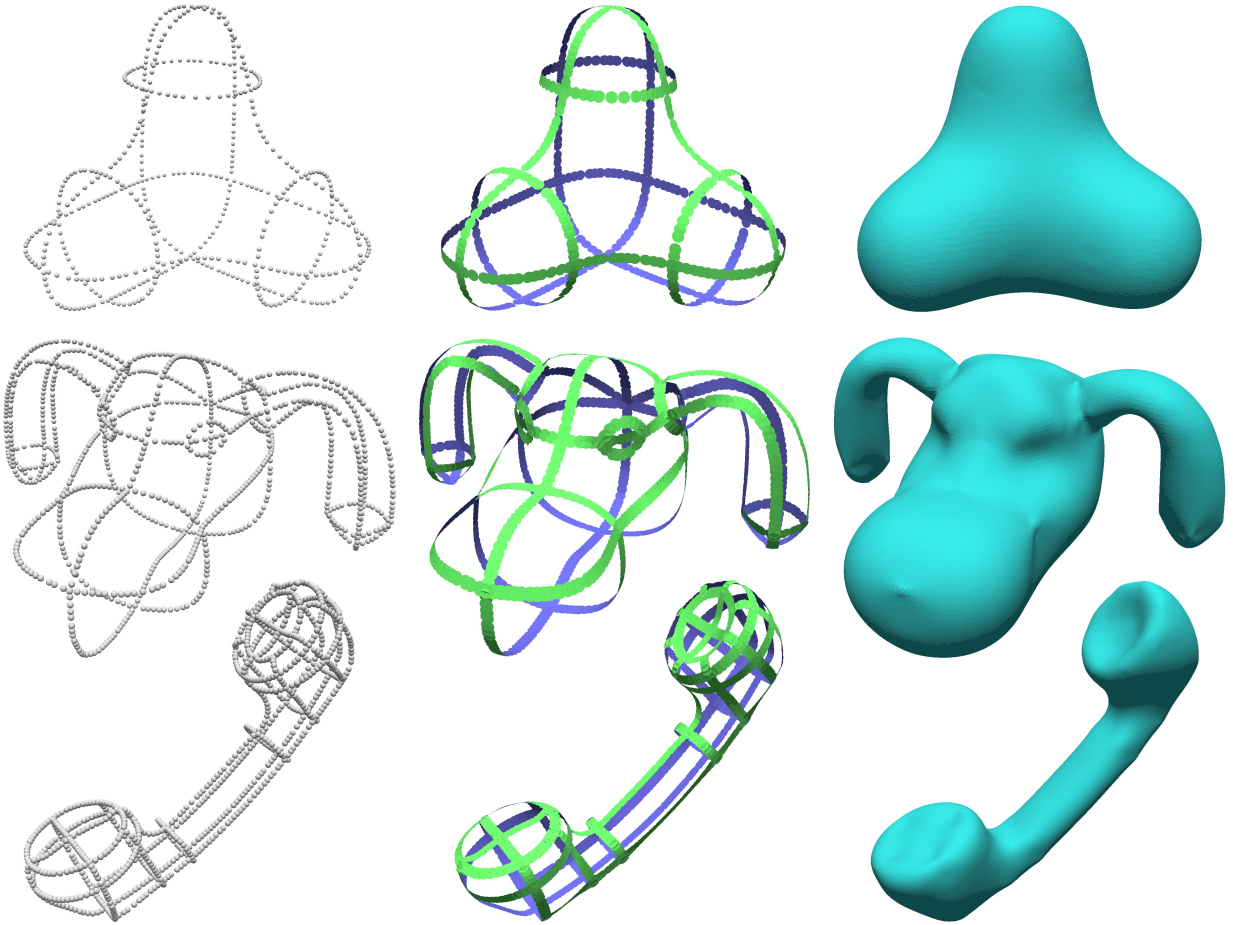


Figure 4.7: Samples from wireframes Trebol, Dog, Phone (left, containing 500, 1000, 1000 points), optimized vectors \mathbf{g} (middle), and VIPSS (right, $\lambda = 0$).

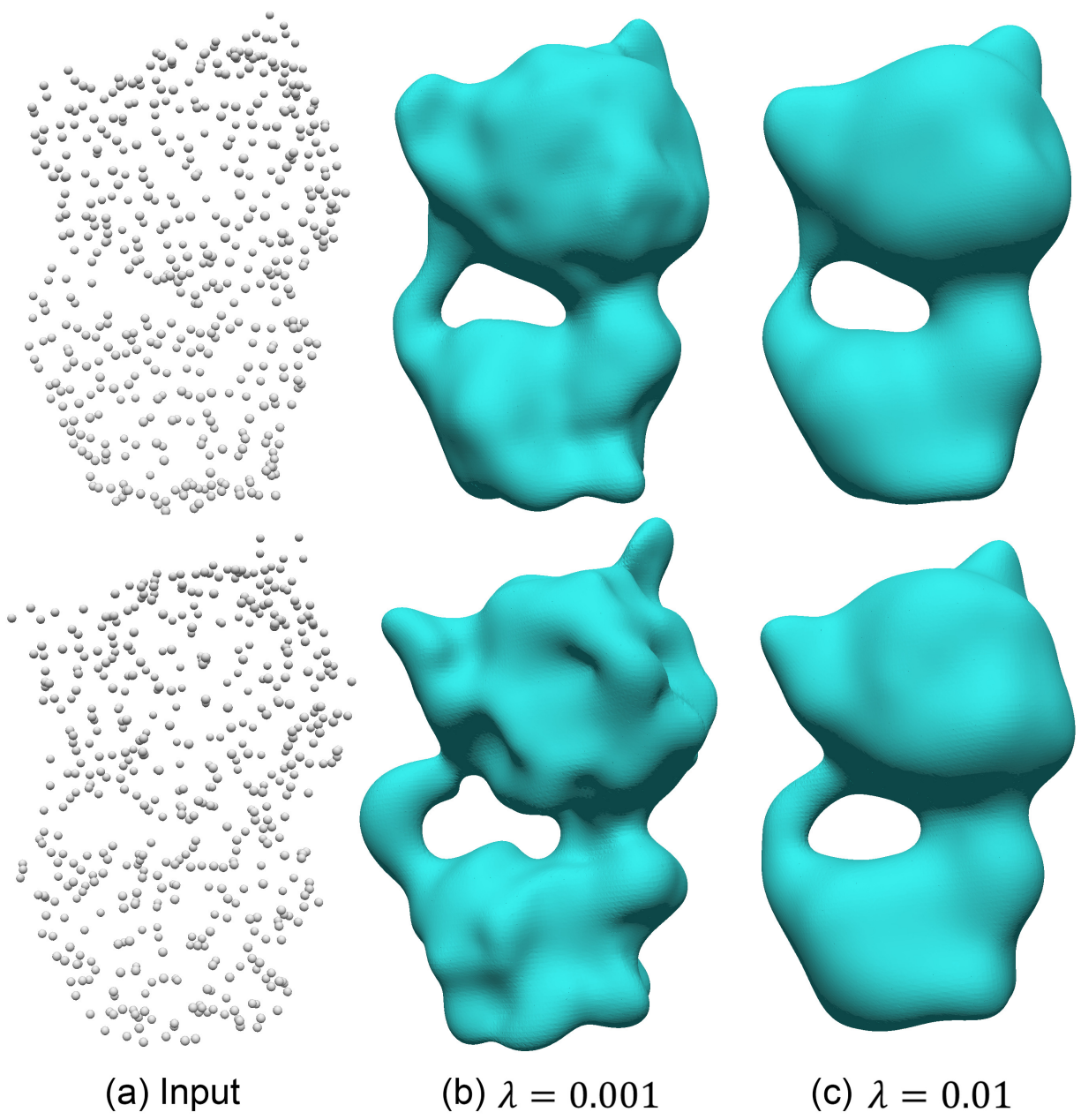


Figure 4.8: (a): Two sampling of the Kitten (500 points each) at low (top, 1%) and high (bottom, 5%) noise rate. (b,c): VIPSS with $\lambda = 0.001$ and 0.01 .

similar to ours (4.9). While our matrix H is derived from the global smoothness of the implicit function, theirs is based on two hand-crafted energy terms that measure correlation between nearby normals. We have observed that, while Wang’s method produces more stable results than two-step methods for sparse and non-uniform inputs, careful tuning of its parameters (neighborhood size and energy weighting) is required for individual inputs. Even with our best effort in tuning, Wang’s method can still fail for some inputs, such as the Walrus sketch (Figure 4.1 (e)) and the Bathtub (Figure 4.10, third row), particularly near sparsely sampled thin shape features (e.g., Walrus’ flippers and Bathtub’s curved wall). These errors, in turn, lead to poorly reconstructed surfaces. Observe that our method produces a plausible set of normals for both inputs, and in turn better reconstructions.

We also compared with the deep-learning-based normal estimation method, PCPNet [67] on the same Bathtub example (Figure 4.10, bottom row). This method was unable to give any reasonable normals for the original 800-point sample, so we showed their result on a denser sampling (3000 points). Even at this density, PCPNet produces incorrect normal orientations for a significant portion of the points, which leads to a poor reconstruction.

Surface reconstruction methods As reviewed earlier, most implicit reconstruction methods require oriented points as input. Hence improved normal estimation (e.g., our optimized vectors \mathbf{g}) would benefit these existing methods.

In our setting, using Duchon’s interpolant (or so-called Hermite RBF) has several advantages over other implicit reconstruction methods. First, the interpolant $f_{\mathbf{s},\mathbf{g}}$ can fully utilize the optimized Hermite data \mathbf{s},\mathbf{g} , not just the vectors \mathbf{g} , in the case of approximation with a non-zero λ (and hence non-zero \mathbf{s}). Second, by Proposition 1, Duchon’s interpolant using optimized Hermite data is theoretically optimal in terms of the objective (4.1). Third, and in practice, we observed that Duchon’s interpolant outperforms existing methods for

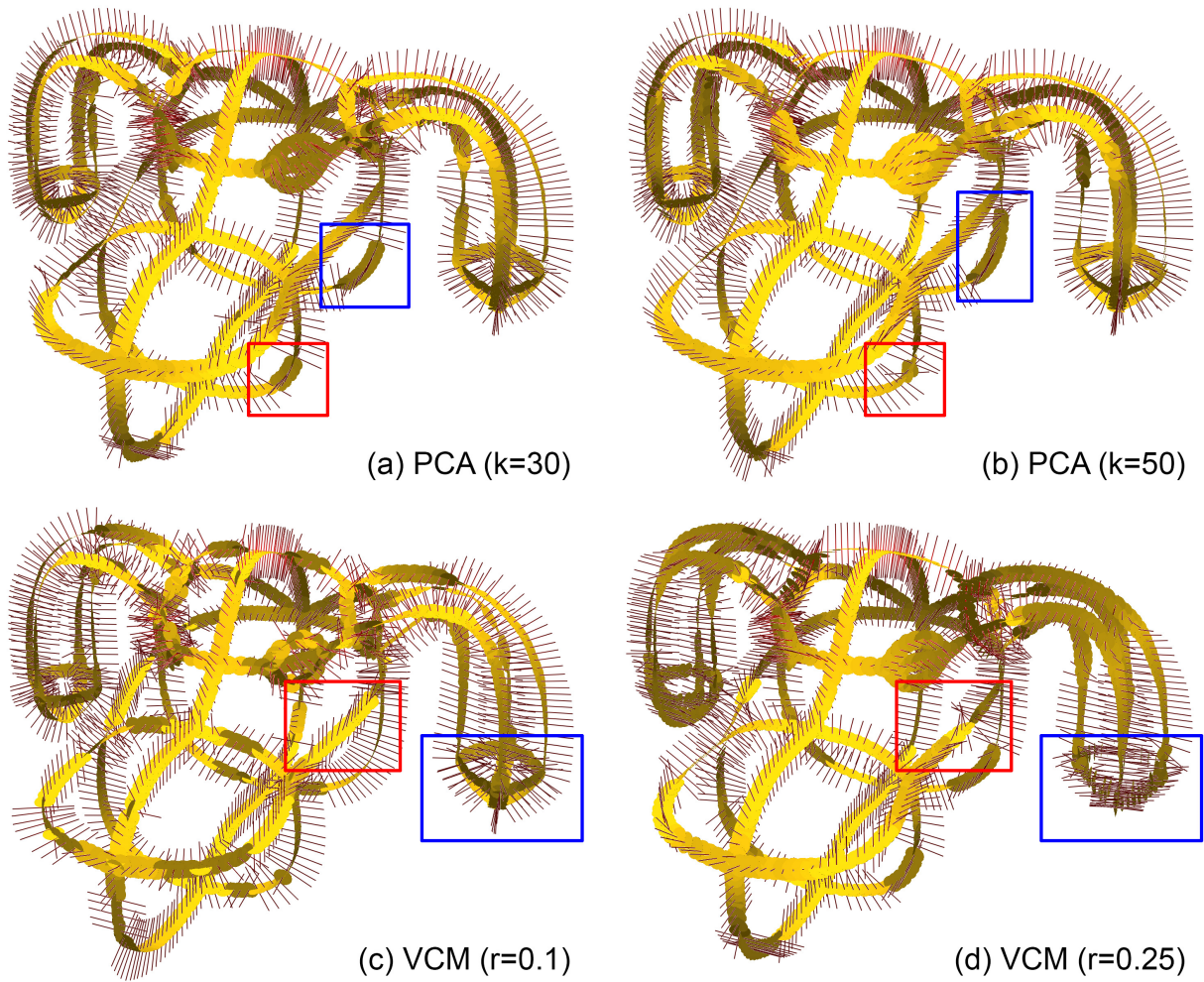


Figure 4.9: Comparing direction estimation on samples from a 3D wireframe using PCA [72] and VCM [100] with different parameters. Each un-oriented direction is shown by a yellow tangent disk and a line segment.

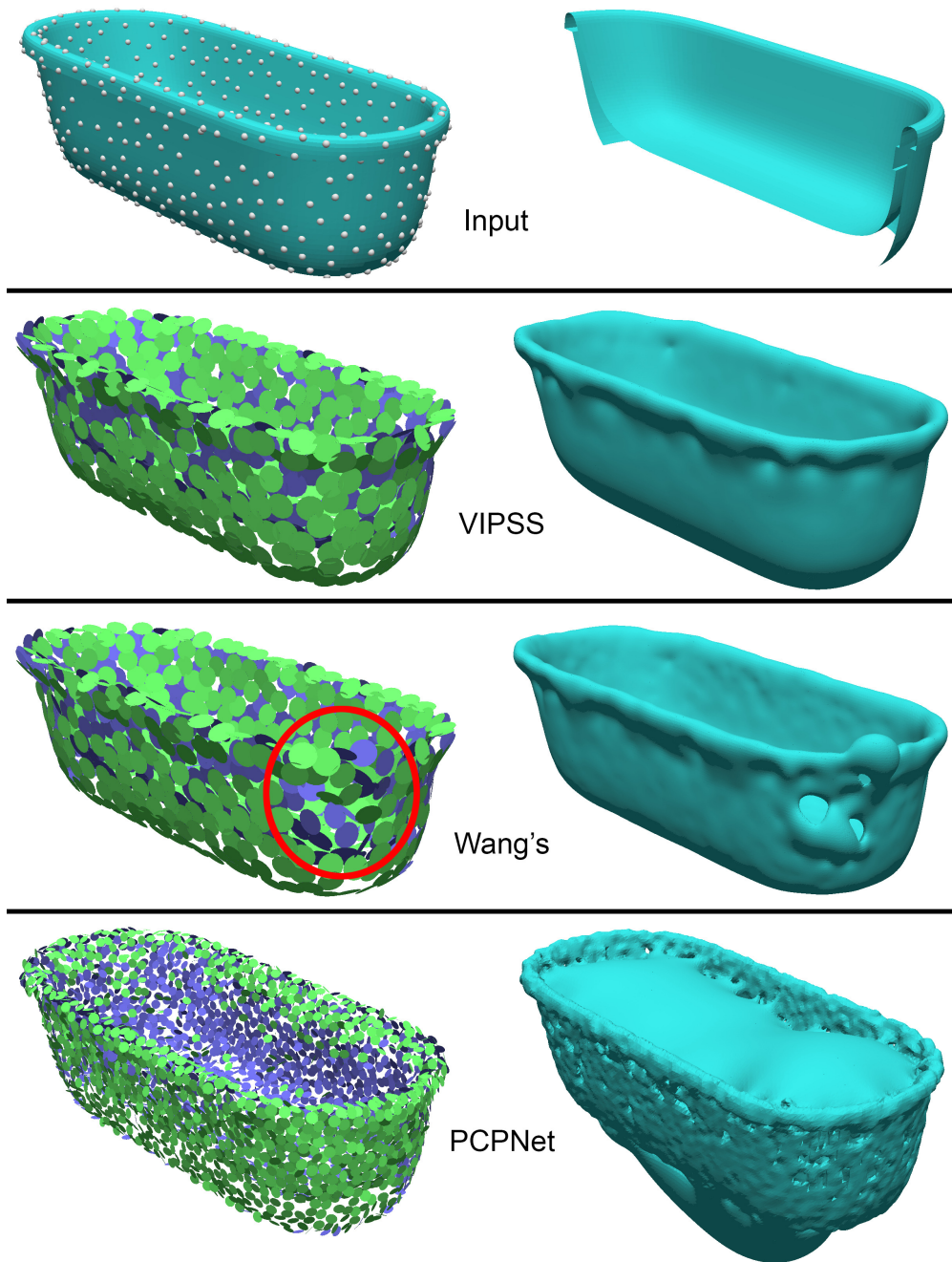


Figure 4.10: Comparing normal estimation from a 800-point sample from the Bathtub (cross-section shown in top-right) using VIPSS ($\lambda = 0$), variational method of [154], and PCPNet [67] (on a 3000-point sampling). The surfaces for these methods are generated using Hermite RBF interpolation (i.e., zero-level set of $f_{\mathbf{s},\mathbf{g}}$ where $\mathbf{s} = 0$ and \mathbf{g} are the estimated normals).

interpolating sparse, oriented points. In Figure 4.11, we compared Duchon’s interpolant with Screened Poisson reconstruction [86] and Algebraic Point Set Surface (APSS) [66] (an IMLS-type method) on the same input, which is a 1000-point sampling of the Stanford Bunny with normals computed by our method. Each of these methods has a parameter that trades off smoothness with closeness of approximation (fitting weight α in Screened Poisson, larger for a closer fit, and filter scale h in APSS, smaller for a closer fit). Observe that closer approximation using these methods leads to surface artifacts (dimples in (d) and tearing in (e)). On the other hand, Duchon’s interpolant results in exact and smooth interpolation (b).

We next compare with a few methods that do not require oriented points in Figure 4.12. The variational method of [5] (b) has trouble handling sparse samples on thin shapes (e.g., the Bathtub), even after we used a very high data-fitting weight in their formulation. Also, since the variational problem is solved on a tetrahedralization of the domain, their method produces less smooth surfaces than our method (e.g., the Dog). On the other hand, combinatorial methods, such as the ball-pivoting method [19], the tight cocone [49], and the power crust [7], are designed for dense samples and generally unsuited for sparse and non-uniform inputs like these.

4.5.3 Performance

Table 4.1 reports the running time of the four steps of our method (see Section 4.4) on the Max Plank data set in Figure 4.5. Observe that these statistics agree with the complexity analysis in Section 4.4.2. The running time is dominated by the computation of H (inverting A) and the initialization of optimization (computing eigenvectors of H), both exhibiting cubic growth with the input size.

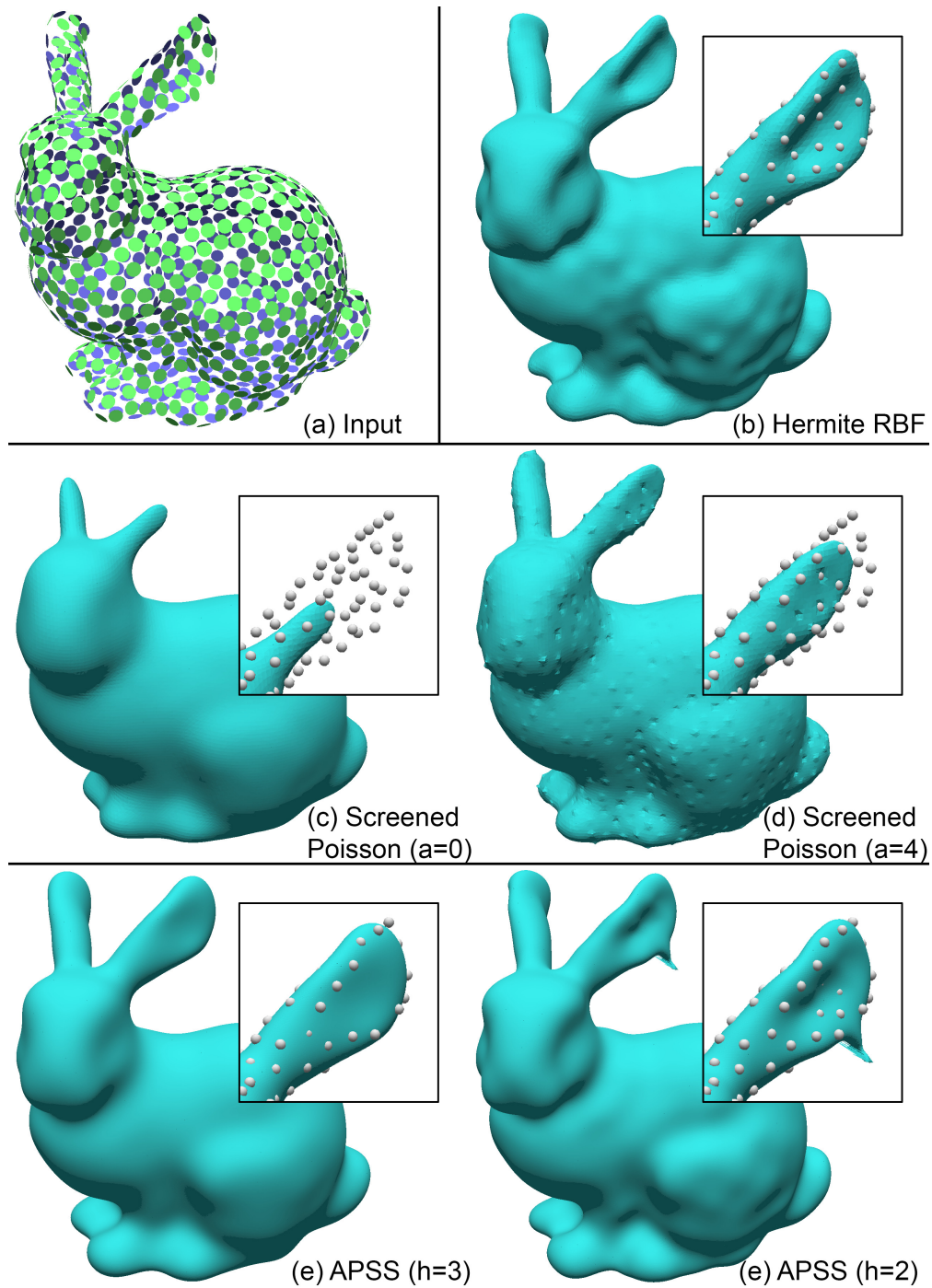


Figure 4.11: Comparing Hermite RBF (Duchon's interpolant) (b) with Screened Poisson [86] (c,d) and APSS [66] (e,f) at different parameters on the same oriented input with 1000 points (a). Orientations in (a) are computed by our method with $\lambda = 0$.

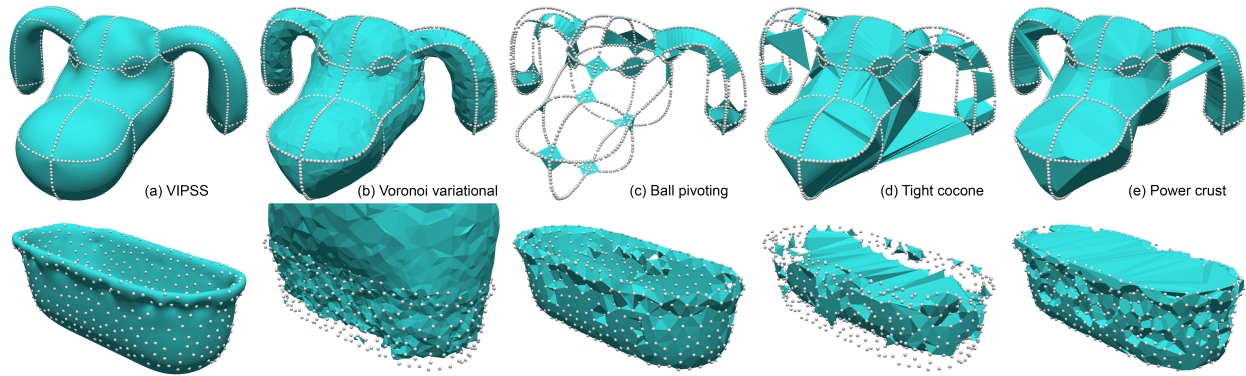


Figure 4.12: Comparing VIPSS ($\lambda = 0$) with methods that do not require oriented inputs: the Voronoi-based variational method of [5], the ball-pivoting method [19], the tight cocone [49], and the power crust [7].

# Points	Compute H	Optimize \mathbf{g}	Build $f_{\mathbf{s},\mathbf{g}}$	Polygonize
500	0.3	2.5 + 0.2	0.0	6.6
1000	1.8	20.0 + 1.1	0.1	13.0
2000	14.6	165.6 + 10.8	0.3	26.0
4000	98.5	1428.2 + 45.4	0.7	51.7

Table 4.1: Running time (in seconds) of each step for Figure 4.5 recorded on a MacBook Pro with 2.5GHz Intel Core i7 CPU and 16 GB memory (implementation done in C++). Timing for Optimization step is written as initialization time (using the offset method of Section 4.4.1) + NLOPT time. Surfacing uses a 100^3 grid.

4.5.4 Application: sketch surfacing

AR/VR sketching tools, such as Google’s Tilt Brush, allow a user to create 3D curvilinear designs in a fully immersive manner. While these tools just display the sketches, the ability to create a surface representation directly from the sketches would further enable interactive 3D modeling. However, freehand 3D sketches are highly unstructured (e.g., with disconnections, over-sketching, missing junctions, etc.), whereas existing methods for surfacing 3D wireframes generally require a clean, connected graph [22, 114, 139].

Due to its resilience to sparse and non-uniform sampling, our method is well-suited to perform this challenging surfacing task. Since we were not able to find a public data set of curves produced by these sketching systems, we simulate the sketches by freehand curves drawn on top of existing surfaces. An example is shown in Figure 4.13 on a sketch of a hand (we use 1000 sample points). The interpolating VIPSS ($\lambda = 0$) is able to resolve the small gap between two nearby fingers (see inserts) at this low sampling rate. A second example is in Figure 4.1, where we further introduced random perturbation to each curve segment (by maximally 5% of the longest dimension of the input) to simulate inaccuracies in free-hand 3D sketches. Our method is able to create a smooth approximation of the Walrus at $\lambda = 0.003$.

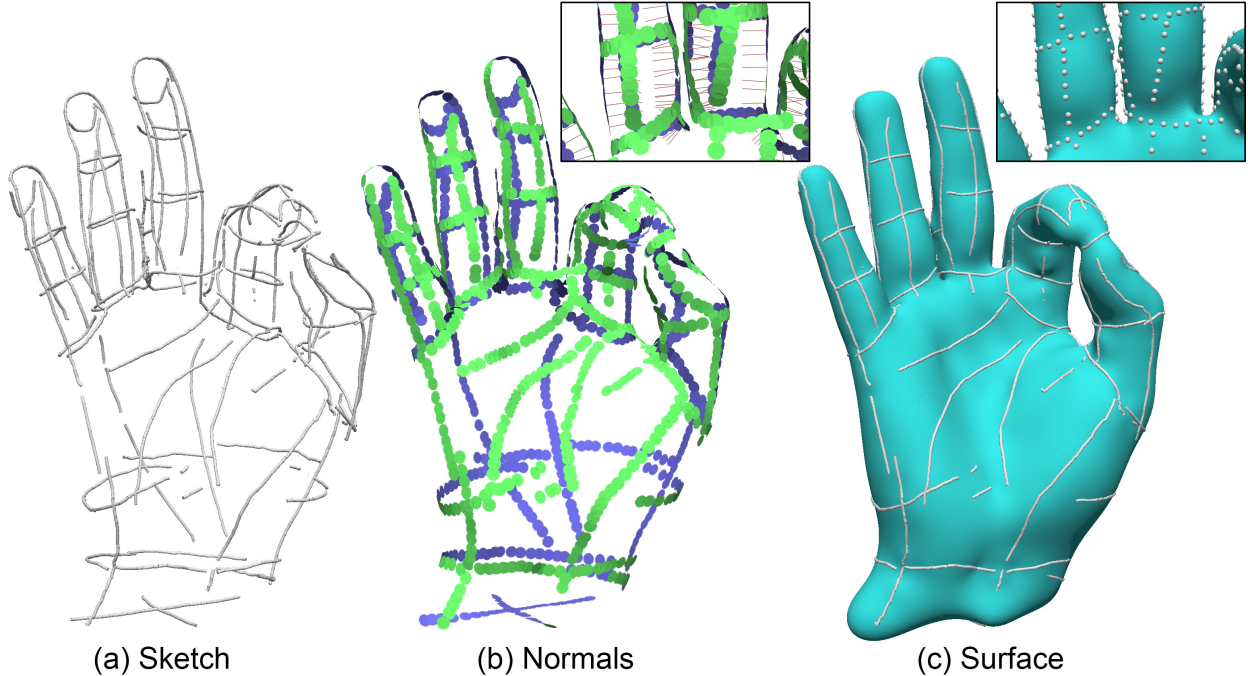


Figure 4.13: VIPSS ($\lambda = 0$) vectors (b) and surface (c) for samples from an unstructured sketch (a). The inserts take a closer look between the index and ring fingers (the line segments in the insert of (b) indicate $-\mathbf{g}$).

4.6 Conclusion and limitations

We describe a novel implicit surface definition (VIPSS) from unoriented point sets that involves a single parameter (zero for exact interpolation), applies to any dimensions, and does not require discretization. Reconstruction using the definition can be easily implemented using standard linear algebra and optimization packages, and the results appear more robust under sparse and non-uniform sampling than existing methods.

Limitations The main limitation of our method is its computational complexity (cubic on the number of points). There are several promising directions that wish to pursue for improving its scalability. Since the matrix M is the Gram matrix of inner products in a semi-Hilbert space, low-rank approximation methods for such matrices [138, 55] can potentially

reduce the complexity of various matrix operations in this work. For large and dense point sets, we may consider an incremental approach akin to [42] that starts with fitting a small subset of points and then incrementally adds more points where the approximation errors are large. Finally, we would like to find more efficient strategies to initialize the optimization than the current offset-based method (Section 4.4.1), which requires multiple eigenvector computations.

The robustness of VIPSS to sparse and non-uniform sampling raises the theoretical question of what is the sampling condition under which VIPSS has guaranteed reconstruction quality (geometrically and topologically). We would also like to explore how the current formulation can be extended to reconstruct shapes with sharp (C^0) features, such as man-made shapes.

Chapter 5

Conclusion and future work

In this dissertation, we aim at surface reconstruction from spatial curves, a fundamental problem in computer graphics. We present novel solutions for pushing the limit of algorithms for this task, tackling challenges resulted from the inherent ambiguity and noise of the curve representation. While various methods have been developed, we extend the existing method or contribute new algorithms allowing them to take the more general form of inputs, such as multi-labeled input, or noisy curve.

We first extend Zou’s work, by introducing the first algorithm for reconstructing multi-labeled material interfaces from cross-sectional curves while providing topological control of individual labels. We define the *interface set* as a tool for systematically exploring material interfaces of a variety of topologies. Beyond the specified application, this work introduces the concept of topologically-controlled reconstruction into the multi-labeled domain. It also have potential to work in problems of different scenarios, such as reconstruction from point cloud data and fixing topological errors on existing material interfaces. In both cases, one can convert the

input in a vector function and potentially apply our method to explore local topological variations in regions surrounding topological ambiguities.

The next two works aim at handling noisy inputs. The second work addresses the critical gap between existing surfacing algorithms which require consistent cross-sectional curves and practical inputs, by developing an algorithm for restoring consistency on non-parallel cross-sections. We formulate the problem into a MIP on implicit functions and provide an efficient optimization strategy. This work builds a bridge between existing algorithms that reconstructs surfaces from cross-sections and the large amount of existing inconsistent data, as well as other wide-spread tools used in medical communities.

The third work directly generates surfaces from noisy wire-frames. We propose the Variational Implicit Point Set Surface that appear more robust under sparse and non-uniform point cloud inputs, and it has exhibited remarkable performance on noisy hand-sketch spatial wire-frames. We believe this work is the first successful attempt in surfacing wire-frames of various kind of unexpected irregularities.

The first two works pave the path of reconstruction algorithms of cross-sectional curves for handling inconsistent inputs and satisfying topological constraint simultaneously. However, we should note that the planar structure and the information of inside/outside, which come with the cross-sections, simplify the problem. The more challenging problem of how to impose topological constraints on the reconstruction of noisy wire-frames, where there is no such additional data, is still under examination. We believe this dissertation contributes a step toward the final goal of controllable and robust algorithms on surface reconstruction of spatial curves.

5.1 Future work

While good performance of the proposed methods has been exhibited in our experiments, interesting venues for future research have also opened up. We briefly discuss several possible directions here.

5.1.1 Analytical formulation of critical offsets

In the first work, we propose the interface set in multi-labeled domain as the counter part of the well-known level set in 2-labeled domain. While thorough theoretical studies have been made on the topological evolution of the level set as the level changes, the theoretical analysis of topological variation of the interface set in the continuous setting, or equivalently, the analytical formulation of critical offsets, demands further investigation. We are already making progress in this direction, and our initial observation is that the criticality of interface sets, like the Jacobi set [57], is linked to certain geometric degeneracy of gradients of the scalar functions f_i . We expect such observation to lead to practical and robust algorithms for analyzing topological events in a piecewise linear interpolation.

5.1.2 Finer level topological control

Another direction for future extension is to offer topological controls at a finer level, such as over the *adjacency* among labels (i.e., whether and how two labels touch). Note that two material interfaces may share the same per-label topology (i.e., components and genus) but differ in their adjacency. Take the 5-labeled input of Figure 5.1 (a) for example, the two reconstructions in (c,d) both have genus-0 for each label but differ in how these labels touch each other. In particular, while the interface between the blue and green labels in (d) forms a

continuous stripe, this interface is broken into several disconnected patches in (c) due to the touching of the other two labels (purple and yellow). Controlling adjacency can be important for applications (e.g., mesh simplification) that are sensitive to the non-manifold structure of the material interface, in addition to the topology of individual labels.

As a simple example for controlling adjacency, we can modify our algorithm to minimize the number of non-manifold *junction points*, where four or more labels meet (red balls in Figure 5.1 (c,d)), in addition to meeting the user-specified per-label topology constraints. This is done by expanding our criticality criteria of active offsets to also check for changes in the number of junction points (Section 2.3.2) and including the total number of junction points on an interface set as part of its score (Section 2.4.1). The modification creates the result in Figure 5.1 (d). We will continue to explore how our algorithm can be extended to offer more extensive and precise controls over label adjacency.

5.1.3 Incremental framework for speeding up VIPSS

The computational complexity (cubic on the number of points) of VIPSS prohibits its application on large point sets. A possible solution for improving the scalability is taking an incremental approach akin to Carr et al. [42]. Given a large/dense point set, we start with sampling a small subset of points uniformly or based on geometry of the point cloud, which we call the chosen set. While a naive way is to directly build up the VIPSS only from the chosen set, we can further consider the residual energy of remaining points as the norm of interpolation values at those points simultaneously. Similar to how we deduce the energy matrix in Eq. 4.9, using Eq.4.5, we can again represent the residual energy into a quadratic term with \mathbf{g} as variables, and then obtain the total energy by combining the residual energy with Duchon’s energy of the chosen set. After getting the estimated normals on the chosen set, we can add more points where the approximation errors are large. For updating the

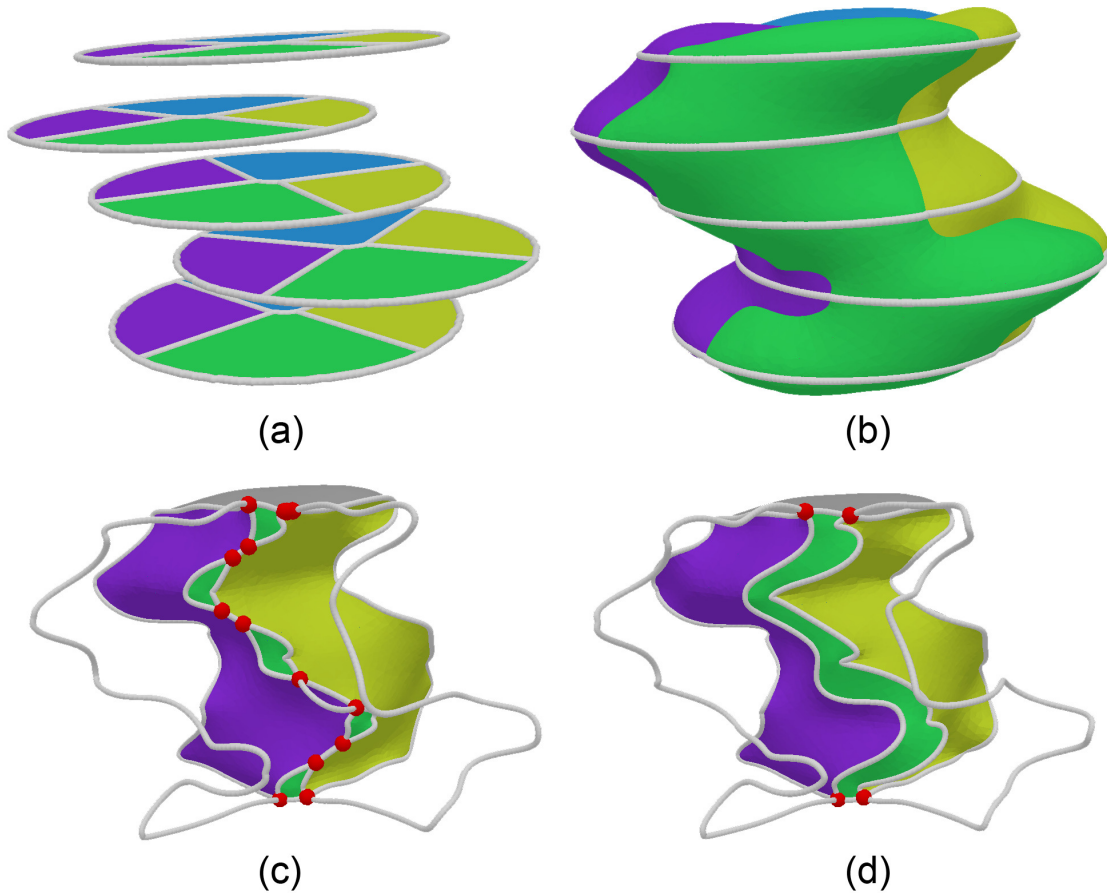


Figure 5.1: Given a stack of 5-labeled slices (a) (only blue and green labels touch on each slice), reconstruction with genus-0 constraint on each label produces multiple patches of interface between the blue and green labels (c), whereas a further modification of the algorithm results in a contiguous interface (d) satisfying the same topology constraints. In (c,d) we only show the surfaces of the blue label colored by its adjacent labels, and junction curves and points are shown as grey wires and red balls. The exterior surface of the reconstruction in (d) is shown in (b).

matrix with the enlarged set of points, we can use block matrix inversion to quickly build up the new energy matrix to avoid the expensive inverting operation, as well as utilize the computed normals on previous chosen set for initialization to avoid the eigen decomposition. We believe this framework can lead to a efficient computation of VIPSS on large point sets.

5.1.4 VIPSS for deforming points

The variational formulation of VIPSS in Duchon’s energy can be informally viewed as finding the best set of unit normals upon the given point sets which leads to the lowest energy among all possible normals. An interesting question is how the best set of normals changes as the position of the point set changes, and can we predict the deviation of normals in an efficient way. The related solution could be potentially applied to modeling deformation, especially continuous deformation over time.

5.1.5 Topologically-controlled VIPSS

An algorithm that offers topological control over the reconstructed surface is always desirable. While VIPSS generates high quality surfaces given only an un-oriented point set, the surface with the lowest energy might not be the desired one that satisfies the prescribed topology. Recently, Poulenard et al. [120] propose an approach for optimizing real-valued functions based on a wide range of topological criteria. They found that the persistence diagrams can be differentiated with respect to the changes in function values, and thus opens up the possibility of topological optimization using continuous optimization techniques. This method could be potentially applied or adjusted to VIPSS, since VIPSS generate a implicit real-value function in the entire domain. It will be interesting to see how the change of the normals and the parameters would change the topology of the underlying surface, and how to use Poulenard’s method to extend VIPSS for taking topological constraints into account.

5.1.6 Learning-based method for surfacing

Deep learning has been proven to be a powerful tool for solving various vision and graphics problem. While it is natural to formulate the task we are solving in this thesis as a building a learning system, who takes a set of curves as input and reconstructs a surface that is best described by these curves as output, there are several challenges whose solution might benefit the deep learning community. For example, taking spatial curves as input or output of a deep neural network has not been thoroughly studied and understood so far. Unlike other types of 3D geometry (e.g., surfaces [105], volumes [156], multi-view projections [141], and point cloud [122]) that have been used in deep neural networks, curves are particularly challenging as they represent a highly non-uniform sampling of the shape: the sampling is dense along the 1-dimensional curves, but non-existing away from these curves. It is not clear how well existing network architectures can be adapted to curves, or whether new representations of curves need to be developed for more effective learning. We are also interesting in directly exploring the possibility of using implicit function as a representation of surfaces, where we can predict the function value given any position in a continuous sense. Another interesting direction is to study the utilization of the energy used in VIPSS as a regularization term or even as main loss function, given the remarkable performance of using the energy in the third work.

References

- [1] Fatemeh Abbasinejad, Pushkar Joshi, and Nina Amenta. Surface patches from unorganized space curves. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, SoCG '12, pages 417–418, New York, NY, USA, 2012. ACM.
- [2] Fatemeh Abbasinejad, Pushkar Joshi, Cindy Grimm, Nina Amenta, and Lance Simons. Surface patches for 3d sketching. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '13, pages 53–60, New York, NY, USA, 2013. ACM.
- [3] Hyung Taek Ahn and Mikhail Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *J. Comput. Phys.*, 226(2):2096–2132, October 2007.
- [4] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003.
- [5] 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*, SGP '07, pages 39–48, 2007.

- [6] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.
- [7] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM, 2001.
- [8] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Transactions on Graphics (TOG)*, 23(3):264–270, 2004.
- [9] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy. Smooth, volume-accurate material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):802–814, 2010.
- [10] John C. Anderson, Christoph Garth, Mark A. Duchaineau, and Ken Joy. Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum (Proc. of Eurographics/IEEE-VGTC Symposium on Visualization 2008)*, 27(3), 2008.
- [11] Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon mesh repairing: An application perspective. *ACM Comput. Surv.*, 45(2):15, 2013.
- [12] Gill Barequet, Michael T. Goodrich, Aya Levi-Steiner, and Dvir Steiner. Straight-skeleton based contour interpolation. *Graph. Models*, 65:323–350, 2004.
- [13] Gill Barequet and Micha Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63:251–272, 1996.
- [14] Gill Barequet and Amir Vaxman. Nonlinear interpolation between slices. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 97–107, 2007.

- [15] Gill Barequet and Amir Vaxman. Reconstruction of multi-label domains from partial planar cross-sections. *Comput. Graph. Forum*, 28(5):1327–1337, 2009.
- [16] Pierre-Louis Bazin and Dzung L. Pham. Topology-preserving tissue classification of magnetic resonance brain images. *IEEE Trans. Med. Imaging*, 26(4):487–496, 2007.
- [17] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1):301–329, 2017.
- [18] Amit Bermano, Amir Vaxman, and Craig Gotsman. Online reconstruction of 3d objects from arbitrary cross-sections. *ACM Trans. Graph.*, 30(5):113:1–113:11, October 2011.
- [19] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.
- [20] Martin Bertram, Gerd Reis, Rolf H. van Lengen, Sascha Köhn, and Hans Hagen. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS’05, pages 199–206, 2005.
- [21] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. Design-driven quadrangulation of closed 3d curves. *ACM Trans. Graph.*, 31(6):178:1–178:11, November 2012.
- [22] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. Design-driven quadrangulation of closed 3d curves. *ACM Transactions on Graphics (TOG)*, 31(6):178, 2012.

- [23] Jules Bloomenthal. Graphics gems iv. chapter An Implicit Surface Polygonizer, pages 324–349. 1994.
- [24] Jules Bloomenthal and Chandrajit Bajaj. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.
- [25] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *ACM SIGGRAPH Computer Graphics*, 24(2):109–116, 1990.
- [26] Jean-Daniel Boissonnat. Shape reconstruction from planar cross sections. *Comput. Vision Graph. Image Process.*, 44(1):1–29, 1988.
- [27] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Computational Geometry*, 22(1-3):185–203, 2002.
- [28] Jean-Daniel Boissonnat and Pooran Memari. Shape reconstruction from unorganized cross-sections. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 89–98, 2007.
- [29] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graph. Models*, 67(5):405–451, September 2005.
- [30] Dobrina Boltcheva, Mariette Yvinec, and Jean-Daniel Boissonnat. Feature preserving delaunay mesh generation from 3d multi-material images. *Comput. Graph. Forum*, 28(5):1455–1464, 2009.
- [31] Kathleen S. Bonnell, Mark A. Duchaineau, Daniel Schikore, Bernd Hamann, and Kenneth I. Joy. Material interface reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 9(4):500–511, 2003.

- [32] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1):213–230, 2008.
- [33] Alexandre Boulch and Renaud Marlet. Fast and Robust Normal Estimation for Point Clouds with Sharp Features. *Computer Graphics Forum*, 2012.
- [34] Alexandre Boulch and Renaud Marlet. Deep Learning for Robust Normal Estimation in Unstructured Point Clouds. *Computer Graphics Forum*, 2016.
- [35] Kenneth A. Brakke. The surface evolver. *Experiment. Math.*, 1(2):141–165, 1992.
- [36] E Brazil, Ives Macedo, M Costa Sousa, Luiz Henrique de Figueiredo, and Luiz Velho. Sketching variational hermite-rbf implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pages 1–8. Eurographics Association, 2010.
- [37] David E Breen and Ross T Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.
- [38] Jonathan R. Bronson, Joshua A. Levine, and Ross T. Whitaker. Lattice cleaving: A multimaterial tetrahedral meshing algorithm with guarantees. *IEEE Trans. Vis. Comput. Graph.*, 20(2):223–237, 2014.
- [39] Martin D Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.
- [40] Hamish Carr and David J. Duke. Joint contour nets. *IEEE Trans. Vis. Comput. Graph.*, 20(8):1100–1113, 2014.
- [41] Hamish Carr, Zhao Geng, Julien Tierny, Amit Chattopadhyay, and Aaron Knoll. Fiber surfaces: Generalizing isosurfaces to bivariate data. *Comput. Graph. Forum*, 34(3):241–250, 2015.

- [42] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.
- [43] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03*, pages 177–187, 2003.
- [44] Daniel Cohen-Or, Amira Solomovic, and David Levin. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics (TOG)*, 17(2):116–141, 1998.
- [45] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):914–924, 2012.
- [46] Fang Da, Christopher Batty, and Eitan Grinspun. Multimaterial mesh-based surface tracking. *ACM Trans. Graph.*, 33(4):112:1–112:11, July 2014.
- [47] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 287–290. ACM, 1995.
- [48] Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2006.
- [49] Tamal K Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. *Journal of Computing and Information Science in Engineering*, 3(4):302–307, 2003.

- [50] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 330–339, 2004.
- [51] Tamal K. Dey, Firdaus Janoos, and Joshua A. Levine. Meshing interfaces of multi-label data with delaunay refinement. *Eng. Comput. (Lond.)*, 28(1):71–82, 2012.
- [52] Tamal K Dey and Jian Sun. An adaptive mls surface for reconstruction with guarantees. In *Symposium on Geometry processing*, pages 43–52, 2005.
- [53] Scott Dillard, Dan Thoma, Bernd Hamann, and John Bingert. Construction of simplified boundary surfaces from serial-sectioned metal micrographs. *IEEE Transactions on Visualization & Computer Graphics*, 13(undefiend):1528–1535, 2007.
- [54] Huong Quynh Dinh, Greg Turk, and Greg Slabaugh. Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE transactions on pattern analysis and machine intelligence*, 24(10):1358–1371, 2002.
- [55] Petros Drineas and Michael W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, December 2005.
- [56] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.
- [57] H. Edelsbrunner and J. Harer. Jacobi sets of multiple morse functions. In *Foundations in Computational Mathematics*, pages 37–57. Cambridge University Press, 2002.
- [58] Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009.

- [59] Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur. Multi-material adaptive volume remesher. *Comput. Graph.*, 58(C):150–160, August 2016.
- [60] Powei Feng, Tao Ju, and Joe D. Warren. Piecewise tri-linear contouring for multi-material volumes. In *Advances in Geometric Modeling and Processing, 6th International Conference, GMP 2010, Castro Urdiales, Spain, June 16-18, 2010. Proceedings*, pages 43–56, 2010.
- [61] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)*, 24(3):544–552, 2005.
- [62] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20(10):693–702, 1977.
- [63] Yotam I. Gingold and Denis Zorin. Controlled-topology filtering. In *Proceedings of the Tenth ACM Symposium on Solid and Physical Modeling 2006, Cardiff University, Wales, UK, June 6-8, 2006*, pages 53–61, 2006.
- [64] Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. Noise-adaptive shape reconstruction from raw point sets. In *Proceedings of the Eleventh Eurographics/ACMSIG-GRAPH Symposium on Geometry Processing*, pages 229–238. Eurographics Association, 2013.
- [65] Cindy Grimm and Pushkar Joshi. Just drawit: A 3d sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, SBIM '12*, pages 121–130, 2012.
- [66] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Transactions on Graphics (TOG)*, 26(3):23, 2007.

- [67] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018.
- [68] David Günther, Alec Jacobson, Jan Reininghaus, Hans-Peter Seidel, Olga Sorkine-Hornung, and Tino Weinkauff. Fast and memory-efficiently topological denoising of 2d and 3d scalar fields. *IEEE Trans. Vis. Comput. Graph.*, 20(12):2585–2594, 2014.
- [69] M. Haitham Shammaa, Yutaka Ohtake, and Hiromasa Suzuki. Segmentation of multi-material ct data of mechanical parts for extracting boundary surfaces. *Comput. Aided Des.*, 42(2):118–128, February 2010.
- [70] Frank Heckel, Olaf Konrad, Horst Karl Hahn, and Heinz-Otto Peitgen. Interactive 3d medical image segmentation with energy-minimizing implicit functions. *Computers & Graphics*, 35(2):275–287, 2011.
- [71] Michelle Holloway, Cindy Grimm, and Tao Ju. Template-based surface reconstruction from cross-sections. *Computers & Graphics*, 58:84–91, 2016.
- [72] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.
- [73] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing*, pages 41–50. Citeseer, 2006.
- [74] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.*, 28(5):176:1–176:7, December 2009.

- [75] Z. Y. Huang, M. Holloway, N. Carr, and T. Ju. Repairing inconsistent curve networks on non-parallel cross-sections. *Computer Graphics Forum*, 37(2):25–35, 2018.
- [76] Zhiyang Huang and Tao Ju. Extrinsically smooth direction fields. *Comput. Graph.*, 58(C):109–117, August 2016.
- [77] Zhiyang Huang, Ming Zou, Nathan Carr, and Tao Ju. Topology-controlled reconstruction of multi-labelled domains from cross-sections. *ACM Transactions on Graphics (TOG)*, 36(4):76, 2017.
- [78] John F Hughes. Scheduled fourier volume morphing. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 43–46. ACM, 1992.
- [79] Takashi Ijiri, Shin Yoshizawa, Yu Sato, Masaaki Ito, and Hideo Yokota. Bilateral Hermite Radial Basis Functions for Contour-based Volume Segmentation. *Computer Graphics Forum*, 32(2):123–132, 2013. Proc. of EUROGRAPHICS’13.
- [80] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189:1–189:15, October 2015.
- [81] Tao Ju, Frank Losasso, Scott Schaefer, and Joe D. Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, 2002.
- [82] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 561–566. ACM, 2005.
- [83] Tao Ju, Qian-Yi Zhou, and Shi-Min Hu. Editing the topology of 3d models by sketching. *ACM Trans. Graph.*, 26(3), July 2007.

- [84] Olga Karpenko, John F Hughes, and Ramesh Raskar. Free-form sketching with variational implicit surfaces. In *Computer Graphics Forum*, volume 21, pages 585–594. Wiley Online Library, 2002.
- [85] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, 2006.
- [86] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013.
- [87] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1):2–11, 1975.
- [88] Byungmoon Kim. Multi-phase fluid simulations using regional level sets. *ACM Trans. Graph.*, 29(6):175:1–175:8, December 2010.
- [89] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Globally optimal direction fields. *ACM Trans. Graph.*, 32(4):59:1–59:10, July 2013.
- [90] Ravikrishna Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms (TALG)*, 4(2):18, 2008.
- [91] Sören König and Stefan Gumhold. Consistent propagation of normal orientations in point clouds. In *VMV*, 2009.
- [92] David Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 37–49. Springer, 2004.
- [93] Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Jin Shiyao. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34(2):94–106, 2010.

- [94] Lu Liu, C. Bajaj, Joseph Deasy, Daniel A. Low, and Tao Ju. Surface reconstruction from non-parallel curve networks. *Comput. Graph. Forum*, 27(2):155–163, 2008.
- [95] Shengjun Liu, Charlie C.L. Wang, Guido Brunnett, and Jun Wang. A closed-form formulation of hrbf-based surface reconstruction by approximate solution. *Comput. Aided Des.*, 78(C):147–157, September 2016.
- [96] Xiuping Liu, Jie Zhang, Junjie Cao, Bo Li, and Ligang Liu. Quality point cloud normal estimation by guided least squares representation. *Comput. Graph.*, 51(C):106–116, October 2015.
- [97] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. Graph.*, 25(3):812–819, July 2006.
- [98] Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer, 1988.
- [99] Josiah Manson, Guergana Petrova, and Scott Schaefer. Streaming surface reconstruction using wavelets. *Computer Graphics Forum*, 27(5):1411–1420, 2008.
- [100] Quentin Merigot, Maks Ovsjanikov, and Leonidas J. Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, June 2011.
- [101] Miriah D. Meyer, Ross T. Whitaker, Robert M. Kirby, Christian Ledergerber, and Hanspeter Pfister. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1539–1546, 2008.
- [102] J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

- [103] Nathan Mitchell, Mridul Aanjaneya, Rajsekhar Setaluri, and Eftychios Sifakis. Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing. *ACM Trans. Graph.*, 34(6):247:1–247:9, October 2015.
- [104] N. J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. In *special issue of International Journal of Computational Geometry and Applications*, volume 14, pages 261–276, 2004.
- [105] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016.
- [106] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings International Conference on Shape Modeling and Applications*, pages 89–98, 2001.
- [107] Patrick Mullen, Fernando De Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum*, 29(5):1733–1741, 2010.
- [108] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. Vis. Comput. Graph.*, 9(2):191–205, 2003.
- [109] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 463–470. ACM, 2003.
- [110] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *Shape Modeling International, 2003*, pages 153–161. IEEE, 2003.

- [111] J-M. Oliva, M. Perrin, and S. Coquillart. 3d reconstruction of complex polyhedral shapes from contours using a simplified generalized voronoi diagram. *Computer Graphics Forum*, 15(3):397–408, 1996.
- [112] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.
- [113] Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. Flow aligned surfacing of curve networks. *ACM Trans. Graph.*, 34(4):127:1–127:10, July 2015.
- [114] Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Changjian Li, and Wenping Wang. Flow aligned surfacing of curve networks. *ACM Trans. Graph. (SIGGRAPH)*, 34(4), 2015.
- [115] Rongjiang Pan and Vaclav Skala. Surface reconstruction with higher-order smoothness. *The Visual Computer*, 28(2):155–162, 2012.
- [116] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, July 2003.
- [117] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *EXPERIMENTAL MATHEMATICS*, 2:15–36, 1993.
- [118] Jean-Philippe Pons, Florent Ségonne, Jean-Daniel Boissonnat, Laurent Rineau, Mariette Yvinec, and Renaud Keriven. High-Quality Consistent Meshing of Multi-Label Datasets. In *International Conference on Information Processing in Medical Imaging 2007*, page 200, Netherlands, 2007.

- [119] Roi Poranne, Craig Gotsman, and Daniel Keren. 3d surface reconstruction using a generalized distance function. In *Computer Graphics Forum*, volume 29, pages 2479–2491. Wiley Online Library, 2010.
- [120] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pages 13–25. Wiley Online Library, 2018.
- [121] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87*, pages 145–152, 1987.
- [122] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.
- [123] Jin Qian and Yongjie Zhang. Dual contouring for domains with topology ambiguity. In *Proceedings of the 20th International Meshing Roundtable, IMR 2011, October 23-26, 2011, Paris, France*, pages 41–60, 2011.
- [124] M. Samozino, M. Alexa, P. Alliez, and M. Yvinec. Reconstruction with voronoi centered radial basis functions. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pages 51–60, 2006.
- [125] R. I. Saye. An algorithm to mesh interconnected surfaces via the voronoi interface. *Eng. with Comput.*, 31(1):123–139, January 2015.
- [126] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *ACM transactions on graphics (TOG)*, volume 25, pages 533–540. ACM, 2006.

- [127] Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold. Towards globally optimal normal orientations for large point clouds. *Comput. Graph. Forum*, 36(1):197–208, January 2017.
- [128] Ryan Schmidt, Brian Wyvill, and Eric Galin. Interactive implicit modeling with hierarchical spatial caching. In *Shape Modeling and Applications, 2005 International Conference*, pages 104–113. IEEE, 2005.
- [129] Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses*, page 43. ACM, 2007.
- [130] Bernhard Schölkopf, Joachim Giesen, and Simon Spalinger. Kernel methods for implicit surface modeling. In *Proceedings of the 17th International Conference on Neural Information Processing Systems, NIPS’04*, pages 1193–1200, 2004.
- [131] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4):151–160, 1986.
- [132] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.
- [133] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse-to-fine surface reconstruction. In *Eurographics*, pages 389–398, Vienna, september 2006.
- [134] Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. Interactive topology-aware surface reconstruction. *ACM Trans. Graph.*, 26(3), July 2007.

- [135] Chen Shen, James F. O’Brien, and Jonathan R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph.*, 23(3):896–904, August 2004.
- [136] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996.
- [137] Hang Si. TetGen. a quality tetrahedral mesh generator and three-dimensional delaunay triangulator., 2007.
- [138] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML ’00, pages 911–918, 2000.
- [139] Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Saguin-Sprynski. Surfacing curve networks with normal control. *Computers & Graphics*, 60:1–8, 2016.
- [140] Florian Steinke, Bernhard Schölkopf, and Volker Blanz. Support vector machines for 3d shape processing. *Computer Graphics Forum*, 24(3):285–294, 2005.
- [141] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *CoRR*, abs/1505.00880, 2015.
- [142] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM, 1995.

- [143] Gabriel Taubin. Smooth signed distance surface reconstruction and applications. In *Iberoamerican Congress on Pattern Recognition*, pages 38–45. Springer, 2012.
- [144] Julien Tierny and Hamish Carr. Jacobi fiber surfaces for bivariate reeb space computation. *IEEE Trans. Vis. Comput. Graph.*, 23(1):960–969, 2017.
- [145] Richard Tsai, Stanley Osher, et al. Level set methods and their applications in image science. *Communications in Mathematical Sciences*, 1(4):1–20, 2003.
- [146] Greg Turk and James F. O’Brien. Shape transformation using variational implicit functions. In *SIGGRAPH ’99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 1999.
- [147] Greg Turk and James F. O’Brien. Shape transformation using variational implicit functions. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’99, pages 335–342, 1999.
- [148] Greg Turk and James F O’Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)*, 21(4):855–873, 2002.
- [149] Greg Turk and James F. O’Brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, 21(4):855–873, October 2002.
- [150] Greg Turk and James F O’Brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, page 13. ACM, 2005.
- [151] J. Waggoner, Y. Zhou, J. Simmons, M. D. Graef, and S. Wang. Topology-preserving multi-label image segmentation. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 1084–1091, 2015.

- [152] C. Walder, O. Chapelle, and B. Schölkopf. Implicit surface modelling as an eigenvalue problem. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 937–944. ACM, 2005.
- [153] Christian Walder, Olivier Chapelle, and Bernhard Schölkopf. Implicit surfaces with globally regularised and compactly supported basis functions. In *Advances in Neural Information Processing Systems*, pages 273–280, 2007.
- [154] Jun Wang, Zhouwang Yang, and Falai Chen. A variational model for normal computation of point clouds. *The Visual Computer*, 28:163–174, 2011.
- [155] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.
- [156] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.
- [157] Hui Xie, Jianning Wang, Jing Hua, Hong Qin, and Arie E. Kaufman. Piecewise C_1 continuous surface reconstruction of noisy point cloud via local implicit quadric regression. In *14th IEEE Visualization 2003 Conference, VIS 2003, Seattle, WA, USA, October 19-24, 2003*, pages 91–98, 2003.
- [158] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. True2form: 3d curve networks from 2d sketches via selective regularization. *ACM Trans. Graph.*, 33(4):131:1–131:13, July 2014.
- [159] Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, Daniel Cohen-Or, and Baoquan Chen. Morfit: Interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph.*, 33(6):202:1–202:12, November 2014.

- [160] Zhan Yuan, Yizhou Yu, and Wenping Wang. Object-space multiphase implicit functions. *ACM Trans. Graph.*, 31(4):114:1–114:10, July 2012.
- [161] Yun Zeng, Dimitris Samaras, Wei Chen, and Qunsheng Peng. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N-D images. *Computer Vision and Image Understanding*, 112(1):81–90, 2008.
- [162] Yongjie Zhang, Thomas J. R. Hughes, and Chandrajit L. Bajaj. Automatic 3d mesh generation for a domain with multiple materials. In *Proceedings of the 16th International Meshing Roundtable, October 14-17, 2007, Seattle, Washington, USA, Proceedings*, pages 367–386, 2007.
- [163] Hong-Kai Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127(1):179 – 195, 1996.
- [164] Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. Simulation of bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '06*, pages 325–333, 2006.
- [165] Yixin Zhuang, Ming Zou, Nathan Carr, and Tao Ju. A general and efficient method for finding cycles in 3d curve networks. *ACM Trans. Graph.*, 32(6):180:1–180:10, November 2013.
- [166] Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph.*, 34(4):128, 2015.

Appendix A

Properties of VIPSS

We show that the VIPSS satisfies a few basic properties that are desirable for surface reconstruction.

A.1 Exact interpolation

It is easy to see that the VIPSS interpolates all the input points when $\lambda = 0$. In this case, $\mathbf{s} = \mathbf{0}$ by (4.11), which implies that the Duchon's interpolant $f_{\mathbf{s},\mathbf{g}}$ is precisely zero at each \mathbf{x}_i . Note that, since the fitting is exact, the optimization problem (4.1) in the general definition of VIPSS reduces to a *parameter-free* form:

$$\text{Minimizes: } E(f)$$

$$\text{Subject to: } \|Df(\mathbf{x}_i)\| = 1, f(\mathbf{x}_i) = 0, \forall i$$

A.2 Linear reproduction

Since Duchon’s energy is 2nd-order, the VIPSS reproduces linear geometry. Specifically, suppose that \mathbf{x}_i span a $(d - 1)$ -dimensional hyperplane in \mathbb{R}^d (e.g., a line in 2D or a plane in 3D). Such hyperplane can be defined as the zero-level set of some linear function f , which has vanishing objective in (4.1). By choosing the f with a unit gradient, we have found a solution to the variational problem of (4.1), whose zero-level set (the VIPSS) is the hyperplane.

A.3 Commutativity with similarity transformations

Ideally, a reconstruction method should be invariant to the change of the coordinate system. In other words, the reconstruction operator should *commute* with similarity transformations (e.g., translation, rotation, and uniform scaling): reconstructing from the transformed points should be equivalent to transforming the reconstruction from the original points.

Commutativity to isometry (translations and rotations) is a direct consequence of the invariance of Duchon’s energy to isometry. Consider a set of transformed points $\mathbf{x}_i = T(\mathbf{x}_i)$ where T is an isometry. For any function f , the objective in (4.1) with respect to the original points \mathbf{x}_i is the same as the objective of the transformed function $\tilde{f}(\mathbf{x}) = f(T^{-1}(\mathbf{x}))$ with respect to the transformed points \mathbf{x}_i . Since gradient magnitudes are preserved under isometry, we conclude that, if f is the solution to (4.1) for \mathbf{x}_i , \tilde{f} must be the solution for \mathbf{x}_i .

Duchon’s energy is not invariant to uniform scaling, but is multiplied by some power of the scale. To ensure that the VIPSS commutes with scaling, the value of λ needs to be properly scaled with the input points. As the next proposition shows, λ should scale *cubically* with the data size.

Proposition 3. *Let f be the solution to (4.1) using Duchon's energy for a given point set \mathbf{x}_i and λ , and $w > 0$. Then $\tilde{f}(\mathbf{x}) = wf(\mathbf{x}/w)$ is the solution for points $\mathbf{x}_i = w\mathbf{x}_i$ ($i = 1, \dots, n$) and $\tilde{\lambda} = w^3\lambda$.*

Proof. In the following, we use symbol \sim for quantities involving the transformed points \mathbf{x}_i . We first note that

$$\begin{aligned}\phi(w\mathbf{x}, w\mathbf{y}) &= w^3\phi(\mathbf{x}, \mathbf{y}) \\ D^{0,1}\phi(w\mathbf{x}, w\mathbf{y}) &= w^2D^{0,1}\phi(\mathbf{x}, \mathbf{y}) \\ D^{1,1}\phi(w\mathbf{x}, w\mathbf{y}) &= wD^{1,1}\phi(\mathbf{x}, \mathbf{y})\end{aligned}$$

Hence matrix \tilde{A} in (4.5) for \mathbf{x}_i is related to A for \mathbf{x}_i by

$$\tilde{A} = w^{-3} W A W$$

where W is a diagonal matrix whose diagonal consists of n of w^3 , dn of w^2 , d of w , and a single 1, in order. The inverse W^{-1} is also a diagonal matrix, whose diagonal consists of n of w^{-3} , dn of w^{-2} , d of w^{-1} , and a single 1, in order. Thus we have the following relation between the inverses, \tilde{A}^{-1} and A^{-1} ,

$$\tilde{A}^{-1} = w^3 W^{-1} A^{-1} W^{-1}$$

and the relations between the sub-matrices,

$$\tilde{J}_{00} = w^{-3} J_{00}, \quad \tilde{J}_{01} = w^{-2} J_{01}, \quad \tilde{J}_{11} = w^{-1} J_{11}$$

Substituting the above into (4.10) and noting that $\tilde{\lambda} = w^3\lambda$ yields

$$\tilde{H} = w^{-1}H$$

Therefore, if \mathbf{g} minimizes $\mathbf{g}^T H \mathbf{g}$, then it also minimizes $\mathbf{g}^T \tilde{H} \mathbf{g}$. For notational consistency, we denote $\mathbf{g} = \mathbf{g}$. By (4.11), we have $\mathbf{s} = w\mathbf{s}$. As a result,

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{e} \\ \tilde{d} \end{pmatrix} = \tilde{A}^{-1} \begin{pmatrix} \mathbf{s} \\ \mathbf{g} \\ \mathbf{0} \\ 0 \end{pmatrix} = w^{-3} W^{-1} A^{-1} W^{-1} \begin{pmatrix} w\mathbf{s} \\ \mathbf{g} \\ \mathbf{0} \\ 0 \end{pmatrix} = \begin{pmatrix} w^{-2}\mathbf{a} \\ w^{-1}\mathbf{b} \\ \mathbf{c} \\ wd \end{pmatrix}$$

Substituting the above into the definition of Duchon's interpolant (4.4) yields $f_{\mathbf{s},\mathbf{g}}(\mathbf{x}) = wf_{\mathbf{s},\mathbf{g}}(\mathbf{x}/w)$. This proves the proposition, because $f_{\mathbf{s},\mathbf{g}}$ and $f_{\mathbf{s},\mathbf{g}}$ are the solutions to (4.1) for inputs $\{\mathbf{x}, \lambda\}$ and $\{\mathbf{x}, \tilde{\lambda}\}$, respectively, due to Proposition 1. \square

The above proposition implies that the VIPSS of the scaled points (zero-level set of \tilde{f}) is the VIPSS of the original points (zero-level set of f) scaled by the same factor w . In summary, the VIPSS undergoes the same similarity transformation with the input data, as long as parameter λ is multiplied by the cubic power of the scaling factor whenever uniform scaling is involved.

Vita

Zhiyang Huang

Degrees

B.S. Electronic Engineering and Information Science, University of Science and Technology of China, June 2014
Ph.D. Computer Science, Washington University in St. Louis, May 2019

Publications

Zhiyang Huang, Nathan Carr, Tao Ju. **Variational Implicit Point Set Surfaces**. *Accepted to ACM SIGGRAPH 2019*.

Zhiyang Huang, Michelle Holloway, Nathan Carr, Tao Ju. **Repairing Inconsistent Curve Networks on Non-parallel Cross-sections**. *Computer Graphics Forum (Proc. Eurographics 2018)*.

Roe Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, Yaron Lipman. **Robust Optimization for Topological Surface Reconstruction**. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2018)*.

Zhiyang Huang, Ming Zou, Nathan Carr, Tao Ju. **Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections**. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2017)*.

Zhiyang Huang, Tao Ju. **Extrinsically smooth direction field**. *Computers & Graphics (58) (Shape Modeling International 2016)*.

May 2019