Variational geometry processing

Wang, Yu

2010

Wang, Y. (2010). Variational geometry processing. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/42057>

<https://doi.org/10.32657/10356/42057>

VARIATIONAL GEOMETRY PROCESSING

WANG YU

School of Physical and Mathematical Sciences

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctoral of Philosophy

2010

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Desheng Wang, for giving me this wonderful opportunity to study and engage in research at NTU and for supporting me over the years. I would also like to thank the Prof. San Ling and Prof. Bernhard Schmidt for giving me the chance to be a part of this wonderful program.

Prof. Alfred M. Bruckstein has long been an inspiration to me. The fruitful discussions with him initiate our project. Prof. Xuecheng Tai's classes and weekly meetings have proved to be one of my best learning experiences at NTU. This thesis has also benefited from various creative ideas due to Prof. Lili Ju. I would also like to thank Prof. Lilian Wang and Dr. Zhijun Tan for their patient helps.

Many thanks due to my friends and colleagues at NTU. It has been a great pleasure getting to work with Songhe Song, Liyong Zhu and Min Wan. Part of results in this thesis comes from the joint works with them. I would like to thank my friends Jie Chen, Jing Zhang, Yuping Duan, Ying Gu, Juan Shi, Qun Wang, Xiaodan Zhao, Michael Daniel Villena Samson, Jiang Liu, Yan Jia and all the members of our groups for making my stay at NTU fun and memorable.

I am very grateful for the help from our former and current graduate program administrator Guat Tin, Cassandra, Wei Har and Denise. Many thanks to them for taking care of all the administrative work for academics and research. I am also very grateful to Melvin, Joe and Peter for establishing fantastic computing facilities and their helpful IT support.

Finally, I would like to thank my family. All this would not have been possible but for the sacrifices, encouragement from them.

Contents

ACKNOWLEDGEMENTS	2
SUMMARY	6
1 INTRODUCTION	7
2 Level-Set Modelling	18
2.1 Introduction	18
2.2 Variational curve smoothing	21
2.3 Level set method on triangular meshes	22
2.4 Adaptive curve smoothing	26
2.4.1 Mesh size modification	26
2.4.2 Mesh Refinement through edge splitting/contraction	27
2.5 Numerical experiments	28
2.6 Concluding Remarks	43
3 Phase-Field Modelling	44
3.1 Introduction	44
3.2 A Phase Field Model for Curve Smoothing	46
3.2.1 The phase field model with varying weight	46
3.2.2 Finite element approximation of the phase field model	49
3.3 Determination of the Weight $\lambda(x)$	50
3.4 Numerical Experiments	54
3.4.1 Two basic examples	54

3.4.2	More complicated examples	58
3.5	Concluding Remarks	61
4	Graph-cuts on Uniform Grids	63
4.1	Introduction	63
4.2	The Curve Smoothing and Reconstruction Functionals	66
4.3	Graph-cuts for Global Minimization	70
4.3.1	Graph Construction of Energy functional	71
4.3.2	Min-cut algorithm	74
4.3.3	Implementation details	75
4.3.4	Corner preserving strategy	76
4.4	Application to Curve Reconstruction from Scattered Points	80
4.5	Numerical Tests	82
4.6	Concluding Remarks	96
5	CVT-based Modelling	105
5.1	Introduction	105
5.2	A discrete model for geometry processing	107
5.3	Generalized edge-weighted centroidal Voronoi tessellations	111
5.3.1	Generalized edge-weighted Voronoi tessellation and its construction	111
5.3.2	Generalized edge-weighted centroidal Voronoi tessellation and its construction	114
5.4	Applications to geometry processing	115
5.4.1	Data representation	116
5.4.2	Determination of λ_{data}	118
5.5	Examples	121
5.5.1	Smoothing Examples	122
5.5.2	Reconstruction Examples	124
5.6	Concluding Remarks	131

6	Graph-cuts on Tetrahedron Meshes	134
6.1	Introduction	134
6.2	Weighted Minimal Surface Model and Level Set Approach	137
6.3	Reconstruction Based on Delaunay Triangulation and Graph Cut . . .	139
6.3.1	The Variational model	139
6.3.2	Mesh and Discretization	141
6.3.3	Graph Construction	144
6.3.4	Algorithm and Discussions	146
6.4	Multiphase Surface Reconstruction	149
6.4.1	Multiphase Segmentation Problems	151
6.4.2	The multiphase Method	153
6.5	Examples	157
6.6	Concluding Remarks	169
7	Conclusion	173
8	Future Works	186
	BIBLIOGRAPHY	200

Summary

In this thesis, two basic topics in geometry processing—curve/surface smoothing and reconstruction are discussed. The variational approaches are used to address these two closely related problems. The methods are based on defining suitable cost functionals to be minimized, and the cost is the combination of a fidelity term and a smoothness term.

By utilizing different representations of interfaces, the energy functional assumes different forms and is minimized by disparate methodologies. In this thesis 3 different representations and 4 different minimization methods are discussed. First two chapters use Partial-Differential-Equation-based (PDE-based) methods to solve curve smoothing problems, in which level-set modelling and phase-field modelling are used respectively. Piece-wise constant functions are used to represent interfaces by discontinues of the functions in Chapters 4, 5, and 6, and combinatorial optimization techniques are applied for the minimization problems. In particular, Chapter 4 discusses a general energy functional framework suitable for geometry processing applications and the corresponding graph-cuts minimization. Under the same variational framework, Chapter 5 solves the minimization problem via centroidal-Voronoi-tessellation-based (CVT-based) methods and discusses multi-phase the problems in terms of clustering language. Chapter 6 dedicates to the surface reconstruction problem, in which the functional is minimized by multi-way graph-cuts on a Delaunay-based tetrahedral mesh so that the advantages of explicit and implicit methods for surface reconstruction are well integrated. Numerous examples substantiate the effectiveness, efficiency and robustness of the proposed methods. In Chapter 7, a systematic comparison is conducted through various examples.

Chapter 1

INTRODUCTION

Smoothing and Reconstruction are two basic topics in geometry processing. For instance, following an image segmentation process, the boundary of the identified object usually has many zigzags and spurious components; due to sampling errors of physical equipments or some other problems, curves or surfaces extracted from images produced by volumetric Magnetic resonance imaging (MRI) or 3D laser scanners inevitably suffer from noises as well [30, 92, 99]. In order to do further analysis, the resulting geometry models have to be smoothed. As the first step of creating computer graphics, curve/surface reconstruction problems from un-organized points received even more attentions from the computer graphics and Computer-aided design (CAD) communities due to the importance of this problem in an era when 3D scanners are commonly used by various industries.

Among the various curve/surface smoothing and reconstruction methods [99, 30, 24, 59, 46, 5, 6, 7, 2, 11, 31, 22, 115, 10, 83, 27, 106, 29, 4, 82, 120], the variational approach is gaining popularity due to its robustness and effectiveness, in which the goal is attained by minimizing well-designed energy functionals via various optimization techniques. In this thesis, these two closely related problems will be treated as different instances of the same variational formalism. One of important variational models for signal and image denoising is the Rudin-Osher-Fatemi (ROF) approach [91]. This denoising framework can be readily adapted to different situations. A particularly interesting scenario is curve, as opposed to function, smoothing: the noise

is assumed to be a perturbation of some smooth underlying curve and the user might be interested in smoothing out the rough curve. Another interesting problem is curve reconstruction which addresses the issue of reconstructing a curve from a set of unordered sample points without any prior knowledge except the fact that the points are noisy samples of a curve.

The standard ROF approach to denoising is: given a noisy function f we seek a smooth version u by minimizing the cost functional

$$E_{ROF}(u, \lambda) = \int_{R^2} |\nabla u| \, dxdy + \lambda \int_{R^2} (u - f)^2 \, dxdy,$$

where $\lambda > 0$ is a relative weight parameter to be selected. In this functional, the first term measures the smoothness of the solution, while the second term represents the similarity between given data and the solution. Following this idea, all curve denoising functionals will comprise the same basic components. And the solution curve is defined as the minimizer of a certain functional, which can be given in an abstract manner as follows,

$$C = \operatorname{argmin}(similarity(C, C_0) + regularity(C)), \quad (1.1)$$

where C_0 is the noisy curve and C is the smooth curve. Thus, to address curve smoothing and curve reconstruction problems in a variational framework one needs first define “distance” functionals that measure the similarity of curves from raw data of curves or sets of points. Several options are available the most natural one, in our opinion, being based on the definition of a “distance field” based on the data and integrating the field intensity along the curve

$$E[C] = \oint_C \varphi_{data}(C(s)) ds, \quad (1.2)$$

where $\varphi_{data}(x, y)$ is a “distance field”, induced by the original noisy curves or points set and C is the curve for which we are looking. This functional is related to the geodesic active contour models in image segmentation [21, 25, 26, 65, 79, 116], or weighted minimal surface models in surface reconstruction [121]. This functional not

only measures the distance between the solution curves and raw data but also controls the smoothness of the solution. However, this functional alone is not suitable for some global optimization techniques such as graph-cuts-based methods discussed in Chapters 4 and 6, since its minimizers are isolated points trivial in most applications, and the meaningful solutions correspond to local minimizers.

Since in the functional (1.2) topology information is not reflected, it is hard to capture global behaviours of the solution. For instance, a simple planar closed curve separates plane into two disconnected regions— interior and exterior. By only investigating the “distance field” induced by the curve, it is impossible to differentiate between interior points and exterior points. Therefore, a similarity measure contains geometric information as well as topological information is needed in our applications. For this purpose, the representation of the curve/surface is crucial. For example, if signed distance functions are used to represent curves/surfaces, the distance between two curves can be measured via the integral of the difference of the signed distance functions :

$$\int_{\Omega} |\phi - \varphi|, \quad (1.3)$$

where ϕ and φ are the signed distance functions of the two curves, and Ω is the domain of interest. Using signed distance to implicitly represent curves/surfaces has been widely accepted since [86] and has the advantage of being continuous around curves/surfaces and topology-friendly as shown in Chapter 2. As a matter of fact, in the above formulation signed distance functions can be replaced by any implicit functions, for instance as used in Chapter 3, the phase functions, which are widely used in the chemical and biological membranes modelling [38, 40]. Different implicit functions may lead to different regularization terms, but the similarity measure can take almost the same form as (1.3) for all implicit functions. Both signed distance function modelling and phase function modelling characterizing underlying curves/surfaces by zero level set, whereas work well in the continuous setting where the implicit function takes real value at each point, they may be lack of efficiency in the discrete

setting where each point only values finite integers. A more natural way in the discrete setting is to characterize the curves/surface by merely labelling functions, or piece-wise constant level set functions [97]. More precisely, the curve/surface is represented by the discontinuities of the labelling function instead of zero level-set. This representation, as used in the Chapters 4, 5 and 6 suits combinatorial optimization techniques perfectly. Moreover, it simplifies the modelling of multiphase problems. Substituting the labelling functions for the signed distance functions in (1.3), the same integral form, representing the symmetric-area-difference in the case of simple closed curves/surfaces concerned, can be used to measure the distance between two curves/surfaces as well.

Besides the basic “similarity” term, the desired functional must have a smoothness or regularity term, and a natural choice for the regularity term is the arc-length of the curve or surface area,

$$\oint_C ds. \quad (1.4)$$

The integral is written in arc-length parameterization, but by applying implicit representations above term can be made independent of parametrization. However, as mentioned before, the integral will assume different forms with different representations. Provided that the signed distance function is utilized, (1.4) will have form [84]

$$\int_{\Omega} |\nabla H(\phi)| d\Omega, \quad (1.5)$$

where ϕ is the signed distance function and $H(x)$ is the Heaviside function defined as

$$H(s) = \begin{cases} 1 & s \geq 0, \\ 0 & s < 0. \end{cases}$$

This approach is conventionally called level set modelling. When the phase function is used to represent curves/surfaces, the regularity term will have a quite different form, and traditionally this is named as phase field modelling. Take two-phase problem as

an example, the phase function ϕ has two phases across the domain Ω — it basically takes positive value inside C and negative value outside. Between these two phases there is a transition layer. Thus, the phase function can be naturally employed to label the inside or outside of the interface C . The level set $\{\vec{x} \mid \phi(\vec{x}) = 0\}$ gives the interface, while $\{\vec{x} \mid \phi(\vec{x}) > 0\}$ represents the inside of the interface C and $\{\vec{x} \mid \phi(\vec{x}) < 0\}$ the outside. With such a phase function ϕ , the regularity term can be defined by the following formulation

$$\frac{\epsilon}{2} \int_{\Omega} |\nabla \phi(\vec{x})|^2 d\Omega. \quad (1.6)$$

It is shown in [112] that the formulation (1.6) is approximately proportional to the length of the curve, $|C|$, when the phase function $\phi(\vec{x})$ has the form

$$\phi(\vec{x}) = \tanh(d(\vec{x}, C)/\sqrt{2}\epsilon),$$

where $d(\vec{x}, C)$ is the signed distance of x to the curve C and ϵ the width of transition layer. The difference between the level set modelling and phase field modelling is that the former measures the derivative of the implicit function in the L_1 norm and the latter in the L_2 sense, which will lead to different types of partial differential equations. It is not hard to find that as ϵ goes to zero the phase function converges to the sign function,

$$\lim_{\epsilon \rightarrow 0} \phi(\vec{x}) = 1 - 2H(d(\vec{x}, C)).$$

In other words, phase function becomes a discrete function as the limit case. This also suggests various approximates in the discrete setting.

Considering the indicator function

$$\mathbf{1}_C(\vec{x}) = \begin{cases} 1 & \vec{x} \in C^{int} \\ 0 & \vec{x} \in C^{ext} \end{cases},$$

where C^{int} is the interior region enclosed by the interface C and $C^{ext} = \Omega \setminus C^{int}$.

Then (1.5) becomes

$$\int_{\Omega} |\nabla \mathbf{1}_C| \, d\Omega, \quad (1.7)$$

where $|\nabla \mathbf{1}_C| = \sqrt{\left(\frac{\partial \mathbf{1}_C}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{1}_C}{\partial y}\right)^2}$. Note that $|\frac{\partial \mathbf{1}_C}{\partial x}|^2 h = |\frac{\partial \mathbf{1}_C}{\partial x}|$ and $|\frac{\partial \mathbf{1}_C}{\partial y}|^2 h = |\frac{\partial \mathbf{1}_C}{\partial y}|$, where h is the mesh size, and here standard finite difference approximation is considered. Thus,

$$h \int_{\Omega} |\nabla \mathbf{1}_C|^2 \, d\Omega \approx \int_{\Omega} \left(\left| \frac{\partial \mathbf{1}_C}{\partial x} \right| + \left| \frac{\partial \mathbf{1}_C}{\partial y} \right| \right) \, d\Omega. \quad (1.8)$$

Define

$$|\nabla \mathbf{1}_C|_{ani} := \left(\left| \frac{\partial \mathbf{1}_C}{\partial x} \right| + \left| \frac{\partial \mathbf{1}_C}{\partial y} \right| \right).$$

The following regularization is extensively used in the graph-cuts context [72, 17]

$$\int_{\Omega} |\nabla \mathbf{1}_C|_{ani} \, d\Omega. \quad (1.9)$$

Therefore, to some extent, this anisotropic approximation is just phase-field-like approximation of the (1.7) if uniform grids are used. To discretize the integral $\int_{\Omega} |\nabla \mathbf{1}|_{ani} \, d\Omega$ by a graph the choice of the neighbourhood system plays an important role, since it determines the connectivity of the nodes as well as the weights of the edges, and hence affects the accuracy of approximation. Neighbourhood system aside, in the graph-based discretization different meshes also lead to different discretization. For example, on the uniform mesh one can use Cauchy-Crofton formula [17] to obtain the corresponding neighbourhood system, and on the triangular or tetrahedral a more straightforward approximation of (1.4) can be used. To summarize, a dual graph is built on a primal mesh, in which each element K_i in the primal mesh is associated with a node x_i in the dual graph and an edge (x_i, x_j) is added to the dual graph if K_i and K_j are neighbors in the primal mesh. Two examples of planar primal meshes and dual graphs are shown in Figure 1.1, where (a) gives the primal-dual relationship for a regular grid mesh and (b) for an unstructured triangular mesh.

Figure 1.1: ‘—’ is the primal mesh. ‘- -’ is the dual graph. (a) shows the primal structured grid mesh and the dual graph. (b) shows the primal unstructured triangular mesh and the dual graph

Similarly, for a 3D regular grid a cube in the primal mesh corresponds to a node in the dual graph while a rectangle face corresponds to an edge. For a tetrahedral mesh a tetrahedron in the primal mesh corresponds to a node in the dual graph while a triangle face corresponds to an edge. The edge weight will be the length of the common edge between two elements in 2D and be the area of the common face between two elements. Hence, if the interface is composed of these common faces, its length/area can be easily computed by adding the corresponding edge weights. This of course requires the mesh has certain property to make sure that the solution is embedded in it, which is the main topic of the Chapter 6.

Another way to approximate the line/surface integral (1.4) is the CVT-based approach, called EWCVT in [110]. For simplicity the idea will be explained with 2D examples. Assume that a clustering is given by an indicator function $\mathbf{1}_C$. The idea is to introduce a predefined neighbourhood $\mathcal{N}_\omega$ and edge length is computed by counting the number of the edge points within $\mathcal{N}_\omega$. A characteristic function is defined as

$$\chi_{(x,y)}((x',y')) = \begin{cases} 1, & \text{if } \mathbf{1}_C(x',y') \neq \mathbf{1}_C(x,y) \\ 0, & \text{otherwise} \end{cases}, \quad (1.10)$$

Figure 1.2: Shadow area is the set of local edge points associated with (x, y) .

where pair (x, y) means a point in 2D space, (x', y') a point in $\mathcal{N}_\omega(x, y)$ the neighbourhood of (x, y) . $\mathcal{N}_\omega(x, y)$ is chosen as a disk centred at (x, y) with radius ω . Thus, the infinitesimal arc-length element is given by

$$\delta l = \int_{\mathcal{N}_\omega} \chi_{(x,y)}(x', y') dx' dy'. \quad (1.11)$$

Then $\frac{3}{4\omega^3} \int_\Omega \delta l dx dy$ is proportional to $\oint_\Gamma ds$, as shown in the following. Without loss of generality, assume $\mathbf{1}_\Gamma(x, y) = 0$. Assume that ω is small enough so that the intersection of C and $\mathcal{N}_\omega(x, y)$ can be viewed as a straight line segment (Figure 1.2). Then the δl is equal to a portion of the weighted area of $\mathcal{N}_\omega(x, y)$ as the shadow area shown in Figure 1.2. Thus, for the points in a band with width 2ω along the Γ the following identity holds

$$\delta l = (\omega^2 \theta - \omega^2 \sin \theta \cos \theta), \quad (1.12)$$

and for other points $\delta l = 0$. Hence, instead of computing the integral on the whole domain, the computation is only carried out in the region near Γ . By some computation,

it can be shown that $\frac{3}{4\omega^3} \int_{\Omega} \delta l dx dy = \oint_{\Gamma} ds$.

Based on the above-mentioned methods, the term $\oint_C \varphi_{data}(C(s)) ds$, also involving surface integral, can be discretized in the corresponding context as well. For instance, by the level-set terminology $\oint_C \varphi_{data}(C(s)) ds = \int_{\Omega} \varphi_{data} |\nabla H(\phi)| d\Omega$, and using clustering language $\oint_C \varphi_{data}(C(s)) ds \approx \int_{N_{\omega}} \varphi_{data} \times \chi_{(x,y)}(x', y') dx' dy'$. In a word, the arc-length/surface-area weighted by the function φ_{data} is computable.

Collecting (1.2), (1.3) and (1.4) together, the following general energy functional will be studied in this thesis

$$E_{general}(C) = \oint_C (\alpha_{data} + \beta_{data} \varphi^0(C)) ds + \int_{\Omega} \lambda_{data} |\phi - \varphi| d\Omega, \quad (1.13)$$

where α_{data} , β_{data} and λ_{data} are some parameter functions need to be selected, whose concrete forms depend on the applications. Due to the choices of the implicit representation, the functional may appear in the different forms in each chapter. In Chapter 2, the curve smoothing problem is concerned whilst the parameter functions are chosen as $\beta_{data} = 0$, $\alpha_{data} = 1$ and λ positive constant. Therefore by means of level-set formulation the functional goes as

$$E_1(\phi, \lambda) = \int_{\Omega} |\nabla H(\phi)| d\Omega + \lambda \int_{\Omega} |\phi - \varphi| d\Omega. \quad (1.14)$$

The curve smoothing problem is considered in Chapter 3 as well. Similarly, $\beta_{data} = 0$. but the other parameter functions are different $\alpha_{data} = \epsilon/2$ (ϵ width of the transition layer) due to the phase-field modelling is used and $\lambda_{data} = \lambda(x)/2$ which is a data-dependent function devised to preserve corners. Thus the functional has form

$$E_2(\phi) = \frac{\epsilon}{2} \int_{\Omega} |\nabla \phi(\vec{x})|^2 d\Omega + \frac{1}{2} \int_{\Omega} \lambda(\vec{x}) (\phi(\vec{x}) - \varphi(\vec{x}))^2 d\Omega. \quad (1.15)$$

Chapter 4 uses indicator function to represent the curve and discusses curve smoothing and reconstruction under the same framework. The term β_{data} can not be zero any more, since it is a necessary ingredient in the reconstruction task. Thus in Chapter 4 the following functional will be studied

$$E_3(C) = \int_{\Omega} (\alpha_{data}(\vec{x}) + \beta\varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda_{data}(\vec{x}) |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega.$$

Similarly, the functional will be used in Chapters 5 and 6 in the discrete setting, but interpreted with graph and clustering language respectively.

A lot of mathematical tools have been used in this thesis, however because of the limitation of space the background knowledge of them is omitted. Some useful references are given for interested reader. The basic knowledge about Partial Differential Equations and Calculus of Variation can be found in [48]. [74] presents a beautiful introduction for Elementary Differential Geometry. The mathematical theory of Finite Element Methods is presented in a quite readable way by [20]. [77] is probably the most complete book on Finite Volume Method. An introduction about mesh generation and delaunay triangulation can be found in [51, 55]. A classic book about graph theory and network flows is [50].

The remainder of this thesis consists of five chapters, each of which is made as self-contained as possible. In order to save notations, the same notation in each chapter may have different meaning and explicit descriptions will be given. Chapter 2 focuses on the curve smoothing problem, in which an adaptive method based on level set implementation is presented. The cost functional is minimized via solving the derived Euler-Lagrangian equation, of which the discretization is conducted on unstructured triangular meshes by employing a simple and effective finite volume scheme. Chapter 3 presents the phase field model for curve smoothing, in which a weight function is associated with the similarity measure term in the model so that important geometric features could be well preserved. Finite element approximation of the proposed model is given for its numerical implementation. In Chapter 4 variational methods for curve smoothing and curve reconstruction problems are discussed in a unified framework. The functional is minimized via the graph-cuts technique. In addition, the comparison between level-set implementation and graph-cut implementation is given. Chapter 5 further interprets curve smoothing and reconstruction

problems in a clustering language and the CVT-based algorithms are utilized to minimize the proposed functional. Moreover, the methods are generalized to surface problems and multiphase cases are considered as well. A novel method for surface reconstruction is presented in Chapter 6, in which the functional is efficiently minimized by graph cut methods on the graph dual to a Delaunay-based tetrahedral mesh so that the advantages of explicit and implicit methods for surface reconstruction are well integrated. Furthermore, multi-phase surface reconstruction is realized based on multi-way graph-cuts methods. In Chapter 7, a systematic comparison is conducted through various examples. Finally, the comments on future works are given in Chapter 8.

Chapter 2

Level-Set Modelling

2.1 Introduction

Curve/Surface smoothing is a necessary pre-processing or post-processing step in various applications. For instance, following an image segmentation process, the boundary of the identified object usually has many zigzags and spurious components as shown in Figure 2.1; due to sampling errors of physical equipments or some other problems, curves or surfaces extracted from images produced by volumetric MRI or 3D laser scanners inevitably suffer from noises as well [30, 92, 99]. In order to do further analysis, we often have to smooth out the resulting geometry models.

Figure 2.1: Segmentation of a squirrel image. (a) The original image; (b) the segmented image; (c) the noisy object boundary.

Among the various existing curve and surface smoothing methods [99, 30, 24], the

variational approach is gaining popularity due to its robustness and effectiveness in handling noises. Variational curve smoothing is based on minimizing certain functionals, from which the Euler-Lagrange partial differential equations are derived via calculus of variation and solved by gradient descent flow methods. The functional (1.1) is employed here, which is the combination of a fidelity term measuring the “distance” of a curve from the data and a smoothness term measuring the curve’s L_1 -norm or length. To measure the distance between the curve and the given data, one option is

$$E[C] = \int_C \phi_{data}(C(s)) ds,$$

where $\phi_{data}(x, y)$ is a “distance field”, induced by the original noisy curve and C is the curve for which we are looking. This functional is related to the geodesic active contour models in image segmentation [21]. An alternative approach is to measure the distance between curves by the area of “symmetric differences” based on indicator functions, which is also suitable for discrete optimization methods such as graph cuts method used in Chapter 4. However, a very fine mesh is required to represent a curve with adequate accuracy, which renders a huge computational burden for PDE-based methods. Instead, in this chapter an implicit level set function is used to represent a curve, which has been widely accepted since [86] and has the advantage of being continuous around the curve and topology-friendly. Thus, the distance between two curves can be measured via the integral of the difference of the signed distance functions :

$$\int_{\Omega} |\phi - \varphi|,$$

where ϕ and φ are the signed distance functions of the two curves. For the smoothness term, the arc-length is a natural choice, which is the total variation of the Heaviside function in level set terminology. The complete form of the energy functional will be given in Section 2.

To minimize the proposed functional, one choice is to apply the graph-cuts based method, where normally uniform grids are used to construct the graph. Another option is to solve the corresponding Euler-Lagrange PDE via time-marching methods, mainly on uniform grids. In both cases, the use of uniform grids causes an unaffordable

computational cost, in order to capture the geometry irregularity of both the given noisy curve and the final solution curve. Thus applying adaptive triangular meshing is a natural method and is of much necessity. Besides, since multi-scale property is inherent in curve smoothing, using adaptive triangular meshes is also expected to improve the solution. For these purposes, the development of an adaptive level set method motivates this work.

Although efforts have been put to solve the level set equation on quadtree (octree in 3D) meshes [81] and also on triangular meshes in [9], there is no well-developed method on non-uniform triangular meshes for geometric smoothing. On the other hand, finite volume methods (FVM's) [62, 93, 101] have been widely used in computational science and engineering due to the nice properties in observing the conservation laws and working well for unstructured meshes. In this work, an easy-to-implement, yet very effective finite volume scheme is developed for the underlying time dependent PDE. The development of the scheme is based on the assumption that the solution is linear on each triangle so that the gradient is piecewise constant in each triangle. Subsequently, the average nonlinear diffusion term is evaluated via summation of the line integrals around the faces of the control volume. The details of the scheme will be presented in Section 3.

Based on the developed FV level set method, an adaptive procedure for curve smoothing is proposed. The mesh around the newly constructed curve is adaptively refined for iterative solutions of the Euler-Lagrange PDE in FVM until convergence. The mesh refinement bases on sizing modification, normalized edge length computation, local mesh edge splitting and contraction. Also, edge swapping and smoothing are performed for the optimization of the mesh, which enhances the robustness of the FVM. Various numerical examples demonstrate the effectiveness of the proposed adaptive procedure very well.

The remaining part of the chapter is organized as follows. Section 2 introduces the functional for curve smoothing problems and briefly reviews the level set implementation on uniform grids. Section 3 discusses the finite volume scheme on unstructured triangular meshes, based on which in Section 4, the adaptive mesh refinement strategy is presented. In Section 5, numerical experiments are presented. Finally, concluding

remarks are given in Section 6.

2.2 Variational curve smoothing

A curve smoothing model usually consists of two basic components, one of which is the similarity measure, and the other is the regularization or smoothness term. And the solution curve is defined as the minimizer of a certain functional, which can be given in an abstract manner as follows,

$$C = \operatorname{argmin}(similarity(C, C_0) + regularity(C)),$$

where C_0 is the noisy curve and C is the smooth curve.

If a curve is represented by its signed distance function, one choice for the similarity term can be the integral of the difference of the two distance functions. And a natural choice for the regularity term is the arc-length of the curve. Therefore, the curve smoothing model can be formulated as

$$E_1(\phi, \lambda) = \int |\nabla H(\phi)| + \lambda \int |\phi - \varphi|, \quad (2.1)$$

where ϕ, φ is the signed distance function of the smooth curve and noisy curve respectively, λ is a given constant parameter and $H(z)$ is the Heaviside function given by

$$H(z) = \begin{cases} 1 & z \geq 0, \\ 0 & z < 0. \end{cases}$$

Minimizing functional E_1 , the corresponding Euler-Lagrangian equation is obtained as

$$-\delta(\phi) \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) + \lambda \frac{\phi - \varphi}{|\phi - \varphi|} = 0.$$

By applying the gradient descent flow method, the time-marching PDE goes as follows :

$$\frac{\partial \phi}{\partial t} = \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda \frac{\phi - \varphi}{|\phi - \varphi|}. \quad (2.2)$$

Note that in many implementations the $\delta(\phi)$ is replaced by the corresponding smooth version delta function [102]. It was also found that replacing $\delta(\phi)$ by $|\nabla\phi|$ or simply 1 could accelerate the convergence in some applications [85, 67]. In our application, replacing $\delta(\phi)$ by 1 works well in most experiments, and thus above equation is applied. For this issue a thorough discussion can be found in [96].

On uniform quad meshes the evolution equation (2) can be discretized in the following finite difference scheme:

$$\frac{\phi^{n+1} - \phi^n}{dt} = D_x^- F_x^{+,n} + D_y^- F_y^{+,n} - \lambda \frac{\phi^n - \varphi^n}{\sqrt{(\phi^n - \varphi^n)^2 + \varepsilon}}, \quad (2.3)$$

where

$$D_x^- = \frac{\phi_{i,j} - \phi_{i,j-1}}{dx}, D_x^+ = \frac{\phi_{i,j+1} - \phi_{i,j}}{dx}, D_y^- = \frac{\phi_{i,j} - \phi_{i-1,j}}{dy}, D_y^+ = \frac{\phi_{i+1,j} - \phi_{i,j}}{dy},$$

and

$$F_x^{\pm,n} = \frac{D_x^{\pm} \phi^n}{\sqrt{(D_x^{\pm} \phi^n)^2 + (D_y^{\pm} \phi^n)^2 + \varepsilon}}, F_y^{\pm,n} = \frac{D_y^{\pm} \phi^n}{\sqrt{(D_x^{\pm} \phi^n)^2 + (D_y^{\pm} \phi^n)^2 + \varepsilon}}.$$

Here ε is a small positive number, which is taken as 10^{-6} in this work.

In order to solve the complicated geometry structures of the given curve, a very fine uniform mesh is required and therefore it causes very high computational costs. Then, the use of unstructured non-uniform triangular meshes is a natural choice. This results in the necessity to develop the level set method on triangular meshes, which will be presented next.

2.3 Level set method on triangular meshes

In [81] adaptive level set methods were developed on quadtree grids and in [9] Barth and Sethian developed level set methods on triangular meshes, and monotone finite element type numerical schemes were proposed for general Hamilton-Jacobi equations.

In addition, the Petrov-Galerkin approximation was utilized to achieve higher order accuracy. In [62] Hu and Shu designed a higher order WENO finite volume schemes for 2-D conservation laws on triangular meshes, which can be applied to level set equations. As all these schemes stem from conservation laws, the numerical fluxes used in these schemes may not work for the nonlinear diffusion problem in equation (2) in a cost-effective manner. In curve smoothing, high order accuracy may not result in a significant difference in the final solution, while in computational fluid dynamics the order of accuracy matters a lot [62]. Therefore, a simpler numerical scheme is expected to work for our curve smoothing problem. For the purpose, a linear finite volume scheme is developed, which is easy to implement and yet very effective.

By Gauss theorem, equation (2.2) can be rewritten as

$$\frac{\partial \bar{\phi}}{\partial t} = \frac{1}{|T_i|} \sum_{j \in N(i)} \int_{e_{ij}} \frac{\nabla \phi}{|\nabla \phi|} \cdot \vec{n} ds - \frac{\lambda}{|T_i|} \int_{T_i} \frac{\phi - \varphi}{|\phi - \varphi|} d\Omega, \quad (2.4)$$

where T_i is the control volume which is the polygon formed by connecting the emanating triangle centroids of vertex i shown in Figure 2.2, and $\bar{\phi}$ is the average of ϕ on control volume T_i whose volume is represented by $|T_i|$. $N(i)$ is the index set of the adjacent vertices and e_{ij} denotes the edge connecting vertices i and j . $\vec{n}$ denotes the unit outward normal vector.

To compute $\nabla \phi$, linear reconstruction via least-square methods is used, which is based on the assumption that $\nabla \phi$ is piecewise constant in each triangle. The reconstruction is of low order and works effectively, although higher order schemes such as WENO [62] could give better numerical accuracy. The latter requires significantly higher computational cost, but hardly makes observable difference on resulting curves. Furthermore, the numerical flux is a subtle issue for higher order reconstruction methods. Thus, for the sake of efficiency, linear reconstruction is preferred here. Without loss of generality, Δ_{iAB} is taken as an example for an illustration of the scheme and

$\partial\Delta_{iAB}$ is its boundary. As $\nabla\phi$ is constant in Δ_{iAB} , by Gauss theorem it follows that

$$\begin{aligned} (\nabla\phi)_{\Delta_{iAB}} &\approx \frac{1}{|\Delta_{iAB}|} \int_{\Delta_{iAB}} (\nabla\phi) d\Omega = \frac{1}{|\Delta_{iAB}|} \oint_{\partial\Delta_{iAB}} \phi \cdot \vec{n} ds \\ &= \frac{1}{|\Delta_{iAB}|} \left(\int_{iA} \phi \cdot \vec{n}_{iA} ds + \int_{AB} \phi \cdot \vec{n}_{AB} ds + \int_{Bi} \phi \cdot \vec{n}_{Bi} ds \right) \\ &= \frac{1}{2|\Delta_{iAB}|} [|iA|(\phi_i + \phi_A)\vec{n}_{iA} + |AB|(\phi_A + \phi_B)\vec{n}_{AB} + |Bi|(\phi_B + \phi_i)\vec{n}_{Bi}]. \end{aligned}$$

Here $\vec{n}_{\{\cdot\}}$ are unit normal vectors and the subscript denotes the corresponding edge in the control volume. Thus, in each triangle a constant $\nabla\phi$ is reconstructed by using nodal values. And the first term of the right hand in (2.4) can be approximated as follows:

$$\frac{1}{|T_i|} \sum_{j \in N(i)} \int_{e_{ij}} \frac{\nabla\phi}{|\nabla\phi|} \cdot \vec{n} ds \approx \frac{1}{|T_i|} \sum_{j \in N(i)} \left[|e_{ij1}| \left(\frac{\nabla\phi}{|\nabla\phi|} \right)_{ij1} + |e_{ij2}| \left(\frac{\nabla\phi}{|\nabla\phi|} \right)_{ij2} \right] \cdot \vec{n}_{ij},$$

where $\left(\frac{\nabla\phi}{|\nabla\phi|} \right)_{ij1}$ and $\left(\frac{\nabla\phi}{|\nabla\phi|} \right)_{ij2}$ are evaluated at the two vertices of the j -th face by the formula of

$$\left(\frac{\nabla\phi}{\sqrt{\phi_x^2 + \phi_y^2 + \epsilon}} \right),$$

and e_{ij1} and e_{ij2} are two segments of the j -th face, which are split by a triangle edge. Although there might be jumps at the intersection points (e.g. the intersection of the segments iA and 17 in Figure 2), the numerical stability is not degraded in our experiments. ϵ is set to be 10^{-6} , which is the same as that on a uniform quad grid. And our numerical experiment results show that the choice is appropriate in terms of numerical stability and solution accuracy. By Taylor expansion, $\phi(x, y) \approx \bar{\phi}_i + \phi_x(x - x_c) + \phi_y(y - y_c)$, which leads to

$$\frac{1}{|T_i|} \int_{T_i} \frac{\phi - \varphi}{|\phi - \varphi|} d\Omega \approx \frac{\phi_i - \varphi_i}{|\phi_i - \varphi_i| + \epsilon}.$$

For the sake of simplicity, the time discretization is conducted in an explicit Euler

Figure 2.2: A example of control volume

finite difference scheme

$$\bar{\phi}^{n+1} = \bar{\phi}^n + \left[\frac{1}{|T_i|} \sum_{j \in N(i)} \int_{e_{ij}} \frac{\nabla \phi^n}{|\nabla \phi^n|} \cdot \vec{n} ds - \frac{\lambda}{|T_i|} \int_{T_i} \frac{\phi^n - \varphi}{|\phi^n - \varphi|} d\Omega \right] \Delta t,$$

and Neumann boundary conditions are enforced by adding ghost cells. The initial condition ϕ^0 is taken to be the distance function generated by the given noisy curve φ .

One important aspect of the above numerical approach is that no numerical flux is defined across control volumes, due to the fact that the gradient is continuous between any two neighboring control volumes. This is because that the common face of two neighboring control volumes lies in the same triangle, whose nodal values are used for the reconstruction of gradients. Therefore, the proposed method is quite different from cell-average based finite volume counterpart, which requires special numerical flux to control oscillations. This merit improves the stability of our method significantly, which is proved by various numerical examples. Also the method is easy to implement and readily to be extended for surface smoothing. And applying the above FV scheme to adaptive curve smoothing is quite straightforward if proper adaptive refining mechanisms can be developed, which will be presented in the following section.

2.4 Adaptive curve smoothing

An adaptive curve smoothing procedure on unstructured triangular meshes consists of loops of the form **SOLVE**→ **SIZING MODIFY**→ **REFINE**. The procedure **SOLVE** solves the time-dependent PDE (2) for the finite volume solution on the current mesh. With the computed level set function values on vertices, the smoothed curve is obtained as the zero level set. Then the procedure **SIZING MODIFY** modifies the sizing of the vertices falling within a narrow band enclosing the zero level set, i.e. the new curve. Finally, the procedure **REFINE** modifies the current triangular mesh by inserting points or contracting edges iteratively so that the refined mesh is consistent with the new mesh sizing distribution. The procedure **SOLVE** was discussed in Section 3, and mesh size modification and refinement will be discussed next, which will be followed by a flow chart of the adaptive curve smoothing algorithm.

2.4.1 Mesh size modification

Basically, the mesh size of each vertex in the current triangular mesh is to be modified based on the computed solution, i.e. the obtained new curve represented by the zero level set. Suppose that h_0 denotes the sizing function of the current mesh, and the signed distance function ϕ is also computed for each vertex, then the new mesh size is modified to $h_0/2$ for the vertices with $|\phi| < ch_0$, which means that the mesh sizing will be halved within a ch_0 -width narrow band around the smoothed curve. And c is a constant parameter which may be tuned in each case. To ensure that the following new solution curve will lie within the refined area, the width of the band shall not be too narrow and c is set to be bigger than 2 in our numerical examples. On the other hand, a too large c will result in unnecessary refinement. A suggested choice is $2 \leq c < 4$. An optimal choice of c will be a future working direction and posteriori error estimate techniques may be pursued.

The mesh sizing modification will be followed by sizing gradation by applying the H-correction procedure proposed in [14], which will reduce the magnitude of ratio of neighboring edge lengths and smooth the sizing distribution as a result.

2.4.2 Mesh Refinement through edge splitting/contraction

The refinement of a given triangular mesh is composed of several local mesh operations which include edge splitting/contraction, edge swapping and point smoothing [52, 53, 107, 108]. Edge splitting and contraction are based on the computation of normalized edge lengths [107], which is utilized as a criterion for refining or coarsening a given edge. Usually, the value $C_s = \sqrt{2}$ is assigned to be the splitting parameter, and the value $C_c = 1/\sqrt{2}$ is set to be the coarsening parameter. For any edge whose normalized length is larger than C_s , a new point is introduced on the midpoint of the edge which is then halved and the two triangles adjacent to the edge is replaced by four new triangles. For any edge whose normalized length of an edge is less than C_c , the edge will be contracted in the way that either two end points are merged into their midpoint or one of them is removed while keeping the other. In edge splitting and contraction, local operations involving edge swapping and point smoothing are also performed to improve the mesh quality. The readers are referred to [54, 100, 13, 68, 109, 105, 42] for the details of the four elementary mesh modification operators.

The refinement procedure modifies, iteratively, an existing triangulation through edge splitting or contraction. The objective is to ensure that mesh elements are in a better conformity with the size specification. Edge lengths computed with respect to vertex sizing function are compared with the given splitting parameter C_s and contraction parameter C_c . After the edge splitting and contraction, the triangulation is finally optimized via a combination of edge swapping and point smoothing. The readers are referred to [107, 108] for the details of a complete procedure of the mesh refinement and coarsening.

To summarize, the following is the flow chart of the algorithm for adaptive curve smoothing.

Algorithm 2.4.1. *Adaptive Curve Smoothing*

Step 1. Define a rectangular region Ω which shall cover the given noisy curve

Γ_0 . Generate an initial coarse triangular mesh M_0 of Ω and derive the mesh sizing function h_0 for the vertices, which can be evaluated as the average length of the edges connecting the vertices. Compute the signed distance function ϕ_0 of Γ_0 by fast sweeping[90]. Give an initial ϕ^0 and compute the initial energy E^0 . Set the current triangular mesh $M = M_0$.

Step 2. Solve equation (2.2) on M by the finite volume scheme and get the solution ϕ^n . Compute the energy E^n . If $\frac{|E^n - E^{n-1}|}{E^{n-1}}$ is less than a given tolerance (which is set to be 10^{-4} for the examples in this chapter), stop; otherwise goto Step 3.

Step 3. Modify the mesh size distribution h_n based on ϕ^n . Refine M based on the new mesh size function h_n and interpolate ϕ_n on the modified mesh M . Let $\phi_0 = \phi^n$, update h_n and Goto Step 2.

2.5 Numerical experiments

In this section, using the above proposed adaptive curve smoothing method several experiments are conducted. In each example, the curve is initially represented by its signed distance function, which is computed via the fast sweeping method introduced in [90]. After several time steps the ϕ in Eq. (2.2) may not be signed distance function strictly due to numerical round-off errors, which is demonstrated in Figure 2.3. But this will not affect the quality of the resulting curve since actually all the level set contours are regularized in modified equation (2.2). In the following examples re-initialization is performed every 1000 iterations so as to correct energy computation. And our numerical results show that it is unnecessary to do more frequent re-initializations. The data statistics of each example involve the comparison of the numbers of mesh nodes, CPU time and energy on uniform and adaptive meshes. CPU times are only counted for FV iterations on each mesh and the overhead of the mesh refinement is excluded since it is negligible comparing to FV iterations, usually only taking 2% – 4%. In this context, by uniform mesh we mean that the mesh size associated to each vertex is almost constant. As for time steps, since the explicit time

Figure 2.3: An example of the evolution of the distance function

	Node	CPU Time	Eng.
Adap. mesh1	503	2.38s	4.267
Adap. mesh2	1335	4.70s	4.201
Adap. mesh3	3026	7.97s	4.155
Adap. mesh4	5293	28.45s	4.142

Table 2.1: Energy and CPU time on adaptive meshes

discretization is utilized, the CFL condition must be satisfied which means $dt \leq h_{min}^2$ where h_{min} is the smallest mesh size. In the following experiments, $dt = 1e - 6$ is small enough even for adaptive meshes.

Experiment 1: A Noisy Circle

The input curve in the first experiment is a noisy circle shown in Figure 2.4(a), which is generated by radially perturbing the points on a circle. The similarity parameter λ is set to be 0.1. The curves shown in Figure 2.4(b), Figure 2.4(c) and Figure 2.4(d) are the solutions obtained on the adaptive meshes shown in Figure 2.4(e), Figure 2.4(f) and Figure 2.4(g), while the energy data are contained in Table 2.1, where ‘Adap.’ denotes ‘adaptive’.

Both the solution curves and the data contained in Table 2.1 clearly show that the adaptation converges as the mesh is refined around the solution curve. The energy is decreasing from 4.267 to 4.155 and 4.142 which demonstrates the convergence. The fourth adaptive mesh *mesh4* is omitted here as the solutions on *mesh3* and *mesh4*

(a) Original noisy circle

(b) Smoothed curve on mesh1

(c) Smoothed curve on mesh2

(d) Smoothed curve on mesh3

(e) mesh1

(f) mesh2

(g) mesh3

Figure 2.4: Noisy circle smoothed on adaptive meshes

Figure 2.5: Smoothed curves on uniform and adaptive mesh

Mesh size	Node	CPU Time	Eng.
0.05	1919	2.33s	4.267
0.025	7462	7.64s	4.17
0.0125	29566	76.45s	4.15

Table 2.2: Energy and CPU time on uniform meshes

have minute difference, which is clearly illustrated by the energy comparison in Table 2.1. The comparison between the solution curve on a uniform mesh with sizing $h = 0.0125$ and that on an adaptive mesh with the minimal mesh sizing $h_{min} = 0.0085$ is conducted and the two curves are shown in Figure 2.5(a) and Figure 2.5(b), respectively. The data statistics for the computations on three uniform meshes with sizing $h = 0.05, 0.025, 0.0125$ are contained in Table 2, which compares with the data in Table 1. It takes 76.45 seconds to obtain a solution with its energy being 4.15 on a uniform mesh, while it only needs around 15 seconds on an adaptive mesh for a solution with the energy being 4.155. This shows that the proposed adaptation procedure is very effective for curve smoothing.

Experiment 2: A Noisy Rectangle

The second experiment is about a noisy rectangle. The computational results on uniform meshes are given in Figure 2.6. It is obvious that the curve becomes smoother

Figure 2.6: Noisy rectangle smoothed on uniform meshes

as the mesh goes finer. Figure 2.7 shows the adaptive meshes and the corresponding solution curves. Here the similarity parameter λ is set to be 3.0. With far less mesh nodes, the adaptive meshes can give results which are as accurate as those on uniform meshes. In Figure 2.8, the curve smoothed on a uniform mesh with sizing $h = 0.0125$ and that on an adaptive mesh (*mesh3* in Figure 2.7) are shown together for comparison.

Table 2.3 contains the CPU time and energy data for the results obtained on uniform meshes with sizing $h = 0.05, 0.025, 0.0125$, respectively. It is clear that the energy is decreasing as long as mesh size becomes small, and also the CPU time becomes longer. Table 2.4 includes the CPU time and energy data for smoothed curves obtained on adaptive meshes shown in Figure 2.7. From Table 2.4, it can be seen that in order to reach the same energy, uniform meshes need four times mesh nodes more than adaptive meshes do, which demonstrates the speed-up of the adaptation procedure significantly.

From the experiment it can also be observed that although smooth curves can be efficiently obtained by adaptive mesh refinement around the whole curve, the corners are smoothed out during the process. The phenomenon will be further investigated later.

Experiment 3: A Noisy Star-Shaped Curve

Figure 2.7: Noisy rectangle smoothed on adaptive meshes

Mesh size	Node	CPU Time	Eng.
0.05	1919	3.00s	5.923
0.025	7462	6.12s	5.834
0.0125	29566	123.55s	5.796

Table 2.3: Energy and CPU time on uniform meshes

Figure 2.8: Comparison of the solution curves on uniform mesh and adaptive mesh

	Node	CPU Time	Eng.
Adap. mesh1	1351	3.72s	5.78
Adap. mesh2	2296	5.96s	5.76
Adap. mesh3	4554	20.72s	5.74

Table 2.4: Energy and CPU time on adaptive meshes

To further show the effectiveness of the proposed adaptive procedure, a complicated star-shaped noisy curve is smoothed in this experiment with $\lambda = 30$. The smoothed curves are shown in Figure 2.9, and the energy and CPU time data are presented in Table 2.5.

The solution obtained on a uniform mesh with 46271 nodes, is given in Figure 2.10(b). The smoothed curve on an adaptive mesh included in Figure 2.9(i) is shown in Figure 2.10(a). The adaptive mesh has only 8271 mesh nodes, which shows the effectiveness of the adaptation procedure very well.

Experiment 4: Refinement for Corner Preserving

	Node	CPU Time	Eng.
Adap. mesh1	503	0.985s	7.026
Adap. mesh2	1236	2.03s	6.917
Adap. mesh3	3386	4.16s	6.897
Adap. mesh4	8271	14.16s	6.599

Table 2.5: Energy and CPU time on adaptive meshes

Figure 2.9: Noisy star-shape curve smoothed on adaptive meshes

Figure 2.10: Smoothed curves on uniform mesh and adaptive mesh

In this experiment, a noisy triangular curve is taken as an example and variant refinement strategies will be investigated. Appropriate choices of similarity parameters and adaptive mesh refinement help to keep corners.

First, we consider the cases when λ is small, which are shown in Figure 2.11. It can be seen that the noisy triangle shown in Figure 2.11(a) is smoothed on a very coarse mesh contained in Figure 2.11(c). The corresponding smooth curve is shown in Figure 2.11(b), where $\lambda = 3$. Next, the mesh is refined around the smoothed curve as in Figure 2.11(e), and smoothing is conducted. It can be clearly observed that the finer mesh improves the smoothness of the curve. However, the three corners are smoothed out.

In order to keep the corners, the similarity parameter λ is increased to 50 and the smoothing is conducted on four different meshes. The first mesh is of uniform size and shown in Figure 2.12(c), on which a smooth curve is obtained as shown in Figure 2.12(b). The three corners are all cut off. Then, a finer uniform mesh is used and the obtained curve preserves the corners much better, but with less smoothness, which is illustrated in Figure 2.12(d) and Figure 2.12(e). Based on the observation, curvature-based adaptive meshing could be of the aid to keep the corners, in the sense that finer meshes can be used in the region where the curvature is relatively larger and coarser mesh can be used in the region where the curvature is smaller. The first curvature-based adaptive mesh is shown in Figure 2.13(c), where the triangulation is refined locally in the region around the three corners where the curvature is relatively larger. The curve obtained on the mesh has a fake corner. However, after

Figure 2.11: Noisy triangle smoothed on different meshes

further refinement both in the corner area and the region around the fake corner, the smoothed curve resolves the fake corner and preserves the three corners very well, which is shown in Figure 2.13(e). The nice property of keeping the corners through curvature-based mesh refinement demonstrates that adaptive meshing not only saves the computational cost, but also assists in corner preserving, which will be further demonstrated in the following examples.

Experiment 5: Another Example for Corner Keeping

The strategy developed in the previous experiment is used here to handle the noisy rectangle introduced in Experiment 3. First, the noisy curve is smoothed with $\lambda = 3$ on a uniform mesh shown in Figure 2.14(e). Then, the mesh sizes of the nodes lying in the circles around the four corners are halved. And the similarity parameter λ is modified to be 30 around the corners and $\lambda = 2$ elsewhere. Figure 2.14(f) shows the mesh after the first refinement and Figure 2.14(c) includes the corresponding smoothed curve. Finally, the triangular mesh is further refined as shown in Figure 2.14(g). The final curve is presented in Figure 2.14(d). Comparing with the result in Figure 2.7(g), the solution curve is smoother and more regular in the sense that the four corners are well kept and the other parts of the curve is much more straight, which is clearly demonstrated in Figure 2.15.

Experiment 6: Two More Complicated Curves

The sixth experiment considers two noisy curves with more complicated geometries. Figure 2.16 contains the noisy and smoothed curves. The solutions show that the proposed adaptive curve smoothing method can handle complicated curves very well. The CPU times on uniform and adaptive meshes are given in Table 2.6 and the data clearly illustrates the effectiveness of the method. It should be pointed out that as solution curves on adaptive and uniform meshes have minute difference, only the solution curves on adaptive meshes are presented.

Experiment 7: Curves from Image Segmentation

The last experiment handles a noisy curve obtained from image segmentation.

Figure 2.12: Noisy triangle smoothed on uniform meshes with $\lambda = 50$

	Robot		Rabbit	
	Uniform mesh	Adaptive mesh	Uniform mesh	Adaptive mesh
Node	46271	13095	46271	8913
CPU time	128.22s	15.36s	108.53s	13.47s
Energy	6.52	6.49	4.84	4.89

Table 2.6: CPU time Comparison

(a) Noisy triangle

(b) Smoothed curve on mesh1 with $\lambda = 50$

(c) mesh1

(d) Smoothed curve on mesh2 with $\lambda = 50$

(e) mesh2

Figure 2.13: Noisy triangle smoothed on curvature-based adaptive meshes

Figure 2.14: Noisy rectangle smoothed on the adaptive meshes further refined around the corners

(a) Noisy rectangle smoothed on the adaptive mesh refined around the curve
(b) Noisy rectangle smoothed on the adaptive mesh with further refinement around the corners

Figure 2.15: Noisy rectangles smoothed on different type of adaptive meshes

(a) Noisy robot

(b) Smoothed robot

(c) Noisy rabbit

(d) Smoothed rabbit

Figure 2.16: Examples with complicated geometry

Figure 2.17: Smooth contours from image segmentation

Figure 2.17(a) contains the contours after segmentation and the smoothed curve is shown in Figure 2.17(b). For simplicity, the data statistics including the CPU time, energy and mesh node numbers are omitted here. All the computational results show that the proposed approach is as effective as in the previous examples. Especially, the capability of the method to deal with noisy input and to handle topological changes is clearly demonstrated.

2.6 Concluding Remarks

In this chapter, a level-set-based method is devised specifically for curve smooth application. In particular, the FVM is utilized to discretize the underlying PDE on unstructured triangular meshes. Based on the proposed FVM a natural adaptive strategy is developed. Comparing with the traditional level-set implementation, the proposed adaptive level-set implementation has two favourable features: (1) the adaptive implementation is able to achieve higher accuracy result with less mesh points, and in turn save computational cost considerably; (2) by means of data dependent meshes, the method's ability of capturing geometric features is enhanced. Through extensive numerical examples, the method is proven to be effective, efficient, and easy to implement.

Chapter 3

Phase-Field Modelling

3.1 Introduction

As discussed in the previous chapter, the problem of *geometry smoothing* or *fairing* such as curve/surface smoothing is of importance in a wide range of fields of science and engineering.

Following the last chapter, this chapter also addresses curve smoothing via Partial differential equation (PDE) based methods. PDE-based methods have gained great attention and success in geometry smoothing. Existing approaches fall into two categories: one is evolution and the other is optimization. The main idea of evolution-based approaches is borrowed from linear heat equations. This technique was originally transplanted into image denoising, called the “P&M diffusion” [89], and in turn extended to geometry smoothing. In [99] Taubin discussed a discretized version of the Laplacian operator on surface meshes. Desbrun et al. [30] used an implicit scheme to obtain a stable diffusion algorithm. Clarenz et al. [24] introduced a process called “anisotropic geometry diffusion” to enhance geometric features of the object while smoothing out noises. Some other evolution-based techniques are discussed in [3, 49]. All these methods are carried out on a discretized manifold. In optimization-based approaches, one first constructs an optimization problem that minimizes certain energy functional [69], and then the goal of smoothing is attained

by solving linear or nonlinear PDEs derived from the corresponding variational formulation as in Chapter 2. The optimization-based approaches usually have well-founded mathematical foundation which may alleviate the need for developing heuristics, and more important, can be easily incorporated into other re-construction tasks.

Among various optimization-based techniques, the level set method is widely employed to define the energy functional by expressing some geometry information such as normal, tangent and curvature in term of the distant function [84]. A major advantage of the level-set formulation is its topology-free property, which makes it suitable for processing very complex shapes with topology change. Motivated by the level set approach, several phase field models [37, 38, 39, 40, 45, 112] were recently developed based on a general energetic variation framework. Similar to the level set formulation, representing curves implicitly enables the phase field method to easily handle changes of topologies. When the transition width between two phases approaches zero, the phase field model with diffuse-interface gradually becomes identical to a sharp-interface level-set formulation and then it can be reduced properly to a classical sharp-interface model. Since all computations can be performed on a fixed regular mesh, the difficult implementation issues such as re-meshing needed by front tracking type methods can be avoided.

In this chapter, a novel phase field model for curve smoothing is proposed, whose energy functional consists of a regularity measure term and a similarity measure term; in particular, the approximated symmetric area difference is used in the similarity measure together with a data-dependent weight function in order to preserve important geometric features such as corners of the curve during the smoothing process. This idea differs from some previous works [24, 98] in the sense that instead of decreasing the diffusion effect, controlling the effect of the external force is considered. More remarkably, we developed an effective algorithm for determination of the weight, in which the weight is computed automatically from the given noisy curve without any artificial interference. An advantage of the proposed phase field model over the level set formulation developed in Chapter 2 is that the weak variational form derived from the proposed model is linear, which enables us to employ existing simple and efficient numerical solution techniques with solid mathematical foundations. A finite

element method is used to discretize and numerically solve the phase field model in this chapter. We note that the method developed in the chapter also can be extended to surface/manifold smoothing without much extra effort.

The rest of this chapter is organized as follows. In Section 2, a novel phase field model for curve smoothing is first proposed together with its finite element approximation. Then in Section 3 we develop an effective algorithm for automatic determination of the weight function associated with the similarity measure from the given data so that important geometric features of the original noisy curve can be preserved during the smoothing process. In Section 4 various numerical examples are presented to demonstrate effectiveness and robustness of the proposed method. Finally, concluding remarks are given in Section 5.

3.2 A Phase Field Model for Curve Smoothing

In this section, we first recall the abstract model for curve smoothing, and present a variational phase field model with varying weight across the physical domain. The weight function is associated with the similarity measure in the energy functional and its value at each point is usually determined by using information from neighborhood of the point. A finite element method is then employed to discretize the proposed phase field model.

3.2.1 The phase field model with varying weight

A curve smoothing model is usually formulated as certain functional which can be presented in an abstract manner (1.1):

$$\Gamma = \operatorname{argmin}(\operatorname{regularity}(\Gamma) + \operatorname{similarity}(\Gamma, \Gamma_0)) \quad (3.1)$$

where Γ_0 and Γ denote the initial noisy curve and the final smoothed one, respectively. The first term is a regularity measure of Γ while the second one is a similarity measure between Γ and Γ_0 . The similarity term plays a role as an external force controlling the *distance* or *difference* between the the noisy and smoothed curves. The solution

(smoothed) curve Γ must be the minimizer of the above functional (3.1).

As in [112], we use a phase function ϕ defined on the physical domain Ω containing the curve Γ to implicitly represent the curve Γ . The phase function ϕ has various phases across the domain Ω — it basically takes positive value inside Γ and negative value outside. Between these two phases there is a transition layer. Thus, the phase function can be naturally employed to label the inside or outside of the curve Γ . The level set $\{x \mid \phi(x) = 0\}$ gives the curve, while $\{x \mid \phi(x) > 0\}$ represents the inside of the curve Γ and $\{x \mid \phi(x) < 0\}$ the outside.

With such a phase function ϕ , we may define the regularity term as in Introduction by the following formulation

$$\text{regularity}(\Gamma) = \frac{\epsilon}{2} \int_{\Omega} |\nabla \phi(x)|^2 dx. \quad (3.2)$$

It is shown in [112] that the formulation (3.2) is approximately proportional to the *length of the curve*, $|\Gamma|$, when the phase function $\phi(x)$ has the form $\phi(x) = \tanh(d(x, \Gamma)/\sqrt{2}\epsilon)$. Here $d(x, \Gamma)$ is the signed distance of x to the curve Γ and ϵ the width of transition layer.

For the similarity term, the *symmetric area difference* of regions surrounded by the initial curve Γ_0 and the smooth one Γ respectively is a natural choice as suggested in [113]. Assume that ϕ_0 is a phase function representing the initial noisy curve. Then the following formulation may be employed as the similarity term

$$\text{similarity}(\Gamma, \Gamma_0) = \frac{1}{2} \int_{\Omega} (\phi(x) - \phi_0(x))^2 dx, \quad (3.3)$$

which is a good approximation to the symmetric difference of areas surrounded by Γ and Γ_0 , $\frac{1}{2} \int_{\Omega} |\phi(x) - \phi_0(x)| dx$.

Using (3.2) and (3.3), we then can define a *basic phase field model* for curve smoothing as follows:

$$\min_{\phi} \mathcal{E}_0(\phi) = \frac{\epsilon}{2} \int_{\Omega} |\nabla \phi(x)|^2 dx + \frac{\lambda}{2} \int_{\Omega} (\phi(x) - \phi_0(x))^2 dx \quad (3.4)$$

where $\lambda > 0$ is a constant weighting parameter and added to control the closeness of

the resulting curve to the noisy curve.

In many applications, feature preservation when fairing curves or surfaces is often very important. As shown by many examples in Chapter 2, a major disadvantage of the above basic curve smoothing model (3.4) is that important geometric features such as corners of the curve are often eliminated along with noises during the smoothing process when a constant weighting parameter λ over the whole physical domain is used. Several anisotropic diffusion models were recently proposed for avoiding this problem. Basically all existing anisotropic diffusion approaches are based on the simple principle that the diffusion effect should be small in the feature regions. In [24, 98] the authors tried to achieve this goal by a properly designed diffusion function which changes the diffusion coefficient across different regions. On the other hand, it is worth noting [113] that changing the diffusion effect is almost equivalent to changing the the weighting parameter λ associated with the similarity measure in (3.4).

Inspired by the above work, we propose the following modified model of (3.4), called a *phase field model with varying weight* as follows:

$$\min_{\phi} \mathcal{E}(\phi) = \frac{\epsilon}{2} \int_{\Omega} |\nabla \phi(x)|^2 dx + \frac{1}{2} \int_{\Omega} \lambda(x) (\phi(x) - \phi_0(x))^2 dx \quad (3.5)$$

where a weight function $\lambda(x) > 0$ is used instead of a constant parameter. The question about how to appropriately determine $\lambda(x)$ for a given noisy curve so that corner information could be well preserved will be addressed in the next section. Thus, the original problem of curve smoothing can be formulated as finding the function $\phi = \phi(x)$ defined on the domain Ω that minimizes the functional $\mathcal{E}(\phi)$. It is easy to see that the functional $\mathcal{E}(\phi)$ is strictly convex, and we have the following theorem on existence of the minimizer of $\mathcal{E}(\phi)$.

Theorem 3.1. *The minimization problem defined by (3.5) has a unique minimizer.*

3.2.2 Finite element approximation of the phase field model

Assume that the minimization problem (3.5) has a minimizer $\phi \in H_0^1(\Omega)$, then ϕ can be characterized by the following weak variational formulation:

$$a(\phi, \psi) = \int_{\Omega} \lambda(x) \phi_0(x) \psi(x) \, dx, \quad \forall \psi \in H_0^1(\Omega), \quad (3.6)$$

where

$$a(\phi, \psi) = \epsilon \int_{\Omega} \nabla \phi(x) \cdot \nabla \psi(x) \, dx + \int_{\Omega} \lambda(x) \phi(x) \psi(x) \, dx.$$

According to the standard Lax-Milgram theory [20], we have the following theorem on well-posedness of the above weak variational formulation:

Theorem 3.2. *If $\phi_0 \in L^2(\Omega)$, then the weak variational formulation defined by (3.6) has a unique solution ϕ in $H_0^1(\Omega)$.*

To construct a finite element approximation of the weak variational formulation (3.6), we take a discrete function space

$$V_h = \{v_h \in C^0(\Omega) \cap H_0^1(\Omega) \mid v_{h|_K} \in P_k(K), \forall K \in J_h\}, \quad (3.7)$$

where J_h is a uniformly regular triangulations of $\bar{\Omega}$ consisting of triangles K whose diameters are bounded above by the mesh size parameter h , and P_k denotes the space of all polynomials of degree not larger than k . Then the finite element approximation of (3.6) is to find $\phi_h \in V_h$ such that

$$\epsilon \int_{\Omega} \nabla \phi_h(x) \cdot \nabla \psi_h(x) \, dx + \int_{\Omega} \lambda(x) \phi_h(x) \psi_h(x) \, dx = \int_{\Omega} \lambda \phi_0(x) \psi_h(x) \, dx, \quad (3.8)$$

for any $\psi_h \in V_h$. The resulting linear system is symmetric positive definite and can be efficiently solved by the preconditioned conjugate gradient or algebraic multi-grid method.

For easy reference, we name the above algorithm the *phase field method for curve smoothing* (PFMCS). According to the finite element theory, convergence of the solution of (3.8) can be proved by standard arguments (see [20]):

Theorem 3.3. *The finite element approximation (3.8) of the weak variational formulation (3.6) has a unique solution $\phi_h \in V_h$ and ϕ_h satisfies the following error estimates:*

$$\|\phi - \phi_h\|_{s,\Omega} \leq Ch^{k+1-s}\|\phi\|_{k+1,\Omega}, \quad 0 \leq s \leq k \quad (3.9)$$

where $\|\cdot\|_{s,\Omega}$ denotes the standard Sobolev space norm.

We also note that the strong form of the Euler-Lagrange equation for the minimization problem (3.5) is given by

$$-\epsilon\Delta\phi(x) + \lambda(x)(\phi(x) - \phi_0(x)) = 0 \quad (3.10)$$

with certain Dirichlet boundary conditions. Thus, the system (3.6) may also be viewed as a weak variational formulation of (3.10).

3.3 Determination of the Weight $\lambda(x)$

The remaining important question is how to appropriately choose the weight function $\lambda(x)$ for a given data (a noisy curve) so that geometric features of the curve such as corners can be well preserved during the smoothing process. Provided that a proper corner detector is available, this can be achieved by specifying a non-homogeneous “attraction potential” which makes corner more attractive — setting the weight $\lambda(x)$ large in regions near corners and small in flat regions. Discrete curvature could be one of good corner detectors in many applications, but since the raw datum are usually eroded by noises in geometry smoothing cases, point-wise evaluation or approximation of the curvature is not reliable at all. Inspired by the neighborhood idea proposed in [110], a similar *averaging technique* is utilized in this chapter to overcome this hurdle in designing the corner detector.

We first describe the main idea behind our approach. Let $x \in \Omega$ be a concerning point. A reference circle $S(x)$ centred at the point x is introduced for the “*averaging*” purpose — for removing effects caused by noises and detecting corners. If the boundary (smooth) curve Γ dose not pass through $S(x)$, then clearly x is not close to the boundary Γ and therefore $\lambda(x)$ should be small. Now, let us concentrate on

the situation of the reference circle $S(x)$ being divided into two parts A and B by Γ . First we suppose that the part of Γ intersecting $S(x)$ is quite straight (i.e., curvature is small); There are totally three cases, see Figure 3.1 for an illustration. In these cases $\lambda(x)$ should be small without a doubt. If a corner of Γ is contained inside of $S(x)$, then we must be much more careful; see Figure 3.2 where four such cases are presented. We think $\lambda(x)$ should be small for the second and fourth cases since x is not very close to the corner, but $\lambda(x)$ should be large for the first and third cases. The above observations also basically hold for the case of a noisy curve if the radius of the circle is large enough to cover the amplitude of noises.

Figure 3.1: A quite straight part of Γ passes through $S(x)$ (i.e., no corner is present). Left: $\text{Area}(A) - \text{Area}(B) \approx 0$; middle: $\text{Area}(B) - \text{Area}(A) > 0$ and $\phi(x) = \phi(B)$; right: $\text{Area}(A) - \text{Area}(B) > 0$ and $\phi(x) = \phi(A)$.

In the following, we present an algorithm for determination of the weight $\lambda(x)$ using local information around x . Denote J_h as a triangulation of the computational domain Ω with nodes $\{v_i\}$.

Algorithm 1 (Determination of $\lambda(x)$):

1. For every node $v_i \in J_h$ in the computational domain Ω :
 - (a) choose a reference circle $S(v_i, r)$ centered at v_i with a radius r ;
 - (b) count the number of nodes, $N = N^+ + N^-$, contained inside the reference circle $S(v_i, r)$ where N^+ denotes the number of nodes with a positive phase value ($\phi_0 > 0$) in $S(v_i, r)$ and N^- that with a negative phase value ($\phi_0 < 0$). Set $N_{min} = \min\{N^+, N^-\}$;
 - (c) if $N_{min} = N^+$ and $\phi_0(v_i) > 0$ or $N_{min} = N^-$ and $\phi_0(v_i) < 0$, then set $s(v_i) = N_{min}/N$, otherwise, $s(v_i) = 1/2$;

Figure 3.2: A highly curved part of Γ passes through $S(x)$ (i.e., a corner is present). Top-left: $|Area(A) - Area(B)| \approx (1 - \alpha/\pi)Area(S(x))$ where α denotes the angle of the corner; top-right: $Area(B) - Area(A) > 0$ and $\phi(x) = \phi(B)$; bottom-left: $Area(A) - Area(B) \approx 0$; bottom-right: $Area(A) - Area(B) > 0$ and $\phi(x) = \phi(A)$.

(d) set

$$\lambda(v_i) = f(s(v_i)) \quad (3.11)$$

where $f(s)$ is a continuous positive function defined on $[0, 1]$.

2. Interpolate $\lambda(v_i)$ based on the triangulation J_h to get a piecewise linear function $\lambda(x)$ which will be used in the finite element approximation (3.8).

In practical implement of Algorithm 1, two things need to be carefully considered. The first one is the choice of the function $f(s)$. It is expected that $f(s(x))$ is small when the curve is relatively straight which means that $f(s(x))$ should attain its minimum at $s = 1/2$. Meanwhile, it is also expected that $f(s(x))$ changes dramatically with respect to $s(x)$ so as to produce big difference between corner points and the other points. In our experiments, $f(s)$ is chosen to be

$$f(s) = \lambda_0 e^{w(1-2s)^2}, \quad (3.12)$$

where λ_0 and w are some given positive constants. The function (4.5) gives $f(s) = \lambda_0$ when $s = 1/2$, and changes rapidly when s goes away from $1/2$. Another possible

candidate is

$$f(s) = \lambda_0(1 + w(1 - 2s)). \quad (3.13)$$

The other issue is the choice of the radius r of the reference circle $S(v_i, r)$. To characterize corners of the curve, the reference circle should be large enough so as to suppress the influence of noises. On the other hand, too large r may lead to a lot of unnecessary computation cost and give inaccurate results especially when $S(v_i, r)$ contains more than one corner. Usually, one may start from a small circle, and gradually increases the radius until a satisfactory result is achieved. In practice, when x is not close enough to a corner of the boundary curve Γ , $s(x)$ calculated by the proposed algorithm is often quite small. In order to make the sharp corners more distinguishable, a threshold can be set such that $s(x)$ is assigned to be zero when $s(x)$ falls below that threshold.

The proposed Algorithm 1 works well except for outliers which might be wrongly recognized as corners by the algorithm. To eliminate such an undesirable effect, the following extra process for detecting outliers can be added into Algorithm 1:

Algorithm 2 (Detection of outliers):

For every node $v_i \in J_h$ in the computational domain:

- (a) choose the reference circle $S(v_i, r_1)$ and count the number of nodes, $N_1 = N_1^+ + N_1^-$, contained inside $S_1(v_i, r_1)$;
- (b) increase the radius of the circle to r_2 , and count the number of nodes, $N_2 = N_2^+ + N_2^-$, contained inside $S(v_i, r_2)$;
- (c) if $N_1^+ \approx N_2^+$ or $N_1^- \approx N_2^-$, then v_i is regarded as a outlier and set $s(v_i) = 1/2$; otherwise, $s(v_i)$ is remained unchanged (evaluated by Algorithm 1).

We set $r_2 = 2r_1$ in our numerical experiments. For curves with outliers, Algorithm 1 is applied and followed by Algorithm 2 so that the weight $\lambda(x)$ of normal points and outliers can be correctly computed. In the following section, various examples will be applied to test and evaluate our PFMCS method.

3.4 Numerical Experiments

The proposed phase field method for curve smoothing with the linear finite element approximation is implemented using C Language. We note that the resulting linear system is solved by a Algebraic Multigrid (AMG) solver from AFEPack (<http://www.acm.caltech.edu/~rli/AFEPack/>). The computational domain is $\Omega = [0, 1] \times [0, 1]$. We set the transition parameter $\epsilon = 0.01$ and use a uniform structured triangular mesh of Ω with 400×400 nodes so that the mesh is fine enough to resolve the transition layers very well. High-order Gaussian quadrature points are used in our experiments to evaluate numerical integrations of the phase function and basis functions.

In the following, the PFMCS method is investigated by experiments in which corners of the given curves are desired to be preserved during the smoothing process; in particular, the PFMCS method with varying weight will be compared with that with constant weight to demonstrate its big advantage over the latter one. We will always take the form of (4.5) for $f(s)$ in Algorithm 1 for computing the weight function $\lambda(x)$ when needed.

3.4.1 Two basic examples

The curve shown in Figure 3.3-(b) is a noisy H-shape curve. It is obtained by randomly perturbing a clear H-shape curve (see Figure 3.3-(a)). We note there is no noise around the twelve corners of the curve, which make it convenient to test whether the PFMCS method with varying weight can preserve these corners. The weight function $\lambda(x)$ is determined by Algorithm 1 with $\lambda_0 = 12$, $r = 0.075$ and $w = 25$; see Figure 3.3-(c)&(d) for visualization of $\lambda(x)$. It is clear that $\lambda(x)$ has larger values around the corners. The smoothed H-shape curve obtained by the PFMCS method with such weight is given in Figure 3.4-(a) and it shows that the noises are removed effectively as well as all its twelve corners being well preserved. It is easy to see that the smoothed curve is an excellent approximation to the original clear one. In addition, the smoothed curve by the PFMCS method with constant weight $\lambda(x) \equiv 75$ is also presented in Figure 3.4-(b), which shows that corners of the rectangle are smeared

out. For the purpose of comparison, the above two smoothed curves are put together in Figure 3.4-(c), which demonstrates that the PFMCS method with varying weight easily outperforms that with constant weight. It is variation of the weight function $\lambda(x)$ that makes such a big difference in smoothing results.

Figure 3.3: A noisy H-shape curve and the corresponding weight $\lambda(x)$ calculated using Algorithm 1. (a) The original clear curve; (b) the initial noisy curve; (c) contour of the corresponding weight; (d) mixture of the noisy curve and the weight contour.

In Figures 3.5 and 3.6, a noisy curve (denoted by “Circ-Rect”) formed by gluing parts of a circle and a rectangle is smoothed by the PFMCS method. The noisy curve presented in Figure 3.5-(b) is again produced by randomly perturbing a clear version of the corresponding “Circ-Rect” curve (see Figure 3.5-(a)), and we specially note that the noise is present on the whole curve in this example. The PFMCS method with varying weight is first tested. We set $\lambda_0 = 40$, $r = 0.0375$ and $w = 40$ in the function (4.5) and the weight function $\lambda(x)$ obviously has larger values in regions close to the corners as seen in Figure 3.5-(c)&(d). From the resulting smoothed curve presented in Figure 3.6-(a), we easily observe that the proposed method preserves all

Figure 3.4: Smoothing of a noisy H-shape curve by the PFMCS method. (a) The smoothed curve using varying weight; (b) the smoothed curve using constant weight; (c) comparison of the two smoothed curves (the curve using varying weight is colored with blue).

corners effectively during the smoothing process and produces a good approximation to the original clear curve at the same time. Compared with the resulting curve by the PFMCS method with constant weight $\lambda(x) \equiv 150$ (see Figure 3.6-(b)&(c)), the former approach clearly does better in preserving the sharp corners.

Figure 3.5: A noisy “Circ-Rect” curve and the corresponding weight $\lambda(x)$ calculated using Algorithm 1. (a) The original clear curve; (b) the initial noisy curve; (c) contour of the corresponding weight; (d) mixture of the noisy curve and the weight contour.

In the following, we will quantitatively justify our observations in the above two

Figure 3.6: Smoothing of a noisy “Circ-Rect” curve by the PFMCS method. (a) The smoothed curve using varying weight; (b) the smoothed curve using constant weight; (c) comparison of the two smoothed curves (the curve using varying weight is colored with blue) .

examples. Denote by C_{clear} the original clear curve (without any noise) and by C_{smooth} the smoothed version of the initial noisy curve by using the PFMCS method. A distance between C_{clear} and C_{smooth} can be defined quantitatively by

$$Dist(C_{clear}, C_{smooth}) = \frac{A_{diff}}{A_{clear}},$$

where A_{diff} is the symmetric area difference of the regions enclosed by the curves C_{clear} and C_{smooth} , and A_{clear} is the area of the region surrounded by C_{clear} . The quantity $Dist(C_{clear}, C_{smooth})$ is in fact the normalized symmetric area difference. Denote by N_{diff} the number of mesh points on which the phase functions representing the curves C_{clear} and C_{smooth} respectively have different signs, and by N_{clear} the number of mesh points on which the phase function representing C_{clear} has a positive sign. When the (uniform) mesh is fine enough, we have

$$Dist(C_{clear}, C_{smooth}) \approx \frac{N_{diff}}{N_{clear}}. \quad (3.14)$$

It is clear that the smaller $Dist(C_{clear}, C_{smooth})$ is, the closer two curves are (i.e., the magnitude of $Dist$ shows how good a denoising result is).

Table 3.1 contains the statistics of the normalized symmetric area difference $Dist(C_{clear}, C_{smooth})$ computed for the H-shape and “Circ-Rect” curve examples. For

H-shape Curve					
Constant weight $Dist(C_{clear}, C_{smooth})$	$\lambda = 25$ 0.0230	$\lambda = 50$ 0.0173	$\lambda = 75$ 0.0165	$\lambda = 150$ 0.0168	$\lambda = 200$ 0.0176
Varying weight $Dist(C_{clear}, C_{smooth})$	$\lambda(x) = 12e^{25(1-2s(x))^2}$ 0.0059				
“Circ-Rect” Curve					
Constant weight $Dist(C_{clear}, C_{smooth})$	$\lambda = 50$ 0.0172	$\lambda = 100$ 0.0130	$\lambda = 150$ 0.0121	$\lambda = 200$ 0.0122	$\lambda = 250$ 0.0125
Varying weight $Dist(C_{clear}, C_{smooth})$	$\lambda(x) = 40e^{40(1-2s(x))^2}$ 0.0080				

Table 3.1: The normalized symmetric area difference $Dist(C_{clear}, C_{smooth})$.

the the H-shape curve example, we listed results by the PFMCS method with constant weights $\lambda(x) \equiv 25, 50, 75, 150, 200$ respectively and the varying weight used above, $\lambda(x) = 12e^{25(1-2s(x))^2}$; for the “Circ-Rect” curve example, we reports results by the PFMCS method with constant weights $\lambda(x) \equiv 50, 100, 150, 200, 250$ respectively and the varying weight $\lambda(x) = 40e^{40(1-2s(x))^2}$. It is clear in both examples that the smoothed curve obtained by the PMFCS method with varying weight gives the minimum $Dist(C_{clear}, C_{smooth})$.

3.4.2 More complicated examples

In the following we will investigate performance of the PFMCS method for more complicated examples. Figure 3.7-(a) shows a noisy polygonal curve and Figure 3.7-(b)&(c) presents the corresponding weight $\lambda(x)$ with $\lambda_0 = 40$, $r = 0.1$, and $w = 20$. It can be observed that due to the influence of noises the contour of $\lambda(x)$ has very large effecting area around some sharp corners, and for some natty corners it even shrinks to a point. This is partly because the noises around the sharp corners have almost the same amplitude as the corners so that the proposed algorithm hardly differentiates corners from noises. The smoothed curve by the PFMCS method with varying weight is given in Figure 3.8-(a). We also compare it with the result by the PFMCS method with constant weight $\lambda(x) \equiv 40$ in Figure 3.8-(b)&(c). The resulting curve does preserve all corners well, but some noises are also kept after smoothing in

regions close to the corners, especially the sharp corners. A obvious compromise is to decrease the parameters λ_0 in (4.5) in order to get smoother results.

Figure 3.7: A noisy polygonal curve and the corresponding weight $\lambda(x)$ calculated using Algorithm 1. (a) The initial noisy curve; (b) contour of the corresponding weight; (c) mixture of the noisy curve and the weight contour.

Figure 3.8: Smoothing of a noisy polygonal curve by the PFMCS method. (a) The smoothed curve using varying weight; (b) the smoothed curve using constant weight; (c) comparison of the two smoothed curves.

The next example is a noisy eight-sided star, as shown in Figure 3.9-(a), with many outliers. If only Algorithm 1 is applied, the outliers will be wrongly recognized as corners. For instance, in Figure 3.9-(b) the outliers are kept as well as corners. In this case, Algorithm 2 has to be applied to further detect outliers and modify the corresponding weights. Figure 3.9-(c) shows the result computed with the weights corrected by Algorithm 2, and it is clear that all the outliers are eliminated. A clearer illustration is given in Figure 4.11, in which contours of the weight functions used in Figure 3.9-(b) and Figure 3.9-(c) are presented respectively.

Figure 3.9: Smoothing of a noisy eight-sided star curve by the PFMCS with varying weight. (a) The noisy eight-sided star curve with outliers; (b) the smoothed curve with the weight function determined by Algorithm 1 only ; (c) the smoothed curve with the weight function determined by Algorithm 1 and Algorithm 2 together.

Figure 3.10: Comparison of the weight functions. (a) Contour of the weight used in Figure 3.9-(b); (b) contour of the weight used in Figure 3.9-(c).

The last example we investigate is from a realistic image. A painted “NTU” logo was photographed (see Figure 4.12-(a)) and then segmented by a simple threshold algorithm. The resulting image and the corresponding boundary curves are shown in Figure 4.12-(b)&(c). Figure 4.12-(d) reports the smoothed curve by the PFMCS method with constant weight and Figure 4.12-(e) shows that by the PFMCS method with varying weight. In Figure 4.12-(d), left parts of the letter “N” and right parts of “U” are disconnected since they are elongated and noisy in the segmented image, and using constant weight parameter is difficult to identify and keep them during the smooth process. However, by using the PMFCS method with varying weight, the very thin parts of “N” and “U” are well identified as important features and therefore

preserved as shown in Figure 4.12-(e).

Figure 3.11: Segmentation result of an image containing words “NTU” and its smoothed boundary curves by the PFMCS method. (a) The original picture; (b) the segmentation result; (c) boundaries of the object; (d) the smooth curve with constant weight; (e) the smoothed curve with varying weight.

3.5 Concluding Remarks

In this chapter a novel phase field model and its finite element approximation are proposed for curve smoothing; Specifically, the weighting parameter associated with the similarity measure in the energy functional may vary across the whole physical domain such that important geometric features such as corners may be well preserved during the smoothing process. We also develop an algorithm for determination of the weight from the given noisy curve without any artificial interference. Various numerical examples are presented to demonstrate effectiveness and robustness of the proposed method.

By means of phase-field modelling and reasonable simplifications, the proposed phase-field model leads to a linear weak variational form, which is efficiently solved by the AMG solver. This makes the proposed phase field approach much more efficient than the level-set formulation developed in Chapter 2. On the other hand, instead of using constant similarity parameters, the weighting parameter enables the phase-field model to preserve important geometric features, especially in some difficult cases which the level-set formulation in Chapter 2 is unable to handle .

The disadvantage of the phase-field modelling comes from the introduce of transition layers. The existence of the transition layer improves the regularity of the phase function and reduces the nonlinearity of the formulation, but also introduces model errors. For example, in our application, a wide transition layer may yield an over-smoothed curve. Thus, phase-field methods normally need the width of transition layers to be small enough and therefore require very fine meshes.

Chapter 4

Graph-cuts on Uniform Grids

4.1 Introduction

The first two chapters have shown that curve smoothing is an interesting problem from both practical and theoretical point of views. Another important and related problem is curve reconstruction from an unorganized set of data points. Figure 4.1 is an illustration of these two problems. This chapter will address these two problems in the same variational formalism. Unlike the previous chapters, the energy functional will be minimized in the discrete context via graph-cuts technique instead of PDE-based methods.

Surface reconstruction problems from 3D points or noisy 3D triangulated surfaces

Figure 4.1: Curve smoothing and reconstruction.

received a lot of attentions from computer graphics and CAD communities due to the importance of this problem in an era when 3D scanners are commonly used by various industries. Therefore many of the issues discussed in this chapter have been the subject of intense investigation in the context of surface fitting, reconstruction and smoothing. For example, Hornung and Kobbelt [61] proposed to perform surface reconstruction based on a variational functional that measures the surface fit to the data by integrating a distance function to the given 3D point set. This is, of course, a Geodesic Active Surface functional [26, 121, 25]. By minimizing a functional with TV regularization and L^1 data fidelity, Zach et. al. [117] presented an approach to reconstruct 3D models from range images. Lempitsky and Boykov in [75] proposed a so-called “touch-expand” graph-cuts algorithm to surface fitting. Combining flux with a regularization term, an idea developed in [70] enabled their method to reconstruct fine details by graph-cuts. The above-mentioned, and many more papers that deal with surface smoothing and reconstruction proposed a wealth of ideas aimed to solve difficult topological and geometric issues that arise in 3D. These works might be regarded as 3D generalizations of the issues we deal with in this chapter and hence one might argue that the simpler 2D problems have already been dealt with, as particular cases of 3D problems. While this might be true in some instances, the 3D studies necessarily address a wealth of issues that cannot arise in 2D, and hence we should be able to do more in the planar case by carefully considering the simpler geometry and fully exploiting it. Furthermore we can readily test new ideas in the 2D case and sometimes export them to the higher dimensional cases. In our view, this chapter servers a dual purpose: it is a survey of ideas on variational methods in the context of curve smoothing and reconstruction and a vehicle for testing some new ideas that involve combination of functionals and data adaptive selection of weighting parameters in a context where the results can be readily obtained and assessed.

To address curve smoothing and curve reconstruction problems in a variational framework we define “distance” functionals that measure the dissimilarity of curves from raw data of curves or sets of points. Several options are available, the most natural one, in our opinion, being based on the definition of a “distance field” based on the data and integrating the field intensity along the curve. Other options could

involve the Hausdorff distance between the geometric objects of interest or Frechet distances, but these are much harder to deal with. The “distance field” approach is also better suited at incorporating various types of weighting functions with adaptive distances that may take into consideration knowledge of spatially varying and data dependent noise (see e.g. heteroscedastic cases [66]).

Using distance functions to measure similarity to data is not a new idea. The idea appears in various papers that deal with image segmentation of edge integration [21, 25, 26, 65, 79, 116] or in using shape priors for object detection and tracking [28, 87]. The distance functions defined in image analysis measured distances to edge regions on high gradient points and were often used to define metrics implicitly based on the 2D image analysed. We shall here also consider an alternative approach which represents closed curves via inside-outside indicator functions and measures distance between curves by integrating functionals measuring the area of “symmetric differences” based on such indicator functions, see e.g. [23]. These different similarity measures yield different functionals whose details will be discussed in Section 2. As an addition to the basic “similarity to data” term we must add to our functional a smoothing term, which, following [91], will be the total variation of the curve. Geometrically, this quantity turns out to simply be the curve’s length.

We shall analyze methods for planar curve smoothing and reconstruction, both in order to assess the properties of such variational methods in the basic and relatively simple 2D case, which has many practical applications (post-processing segmentation results, and various geometric problems like determining the minimal length curves contained in a band) and test new ideas that might be beneficial for all the other applications in 3D and even video processing. Our work tests two important functionals and the combination of these functionals and thoroughly compares the basic numerical implementations in these fundamental 2D problems — something that to the best of our knowledge was not done before.

To carry out the minimization, level set methods are commonly considered. However, level set methods have some well documented problems, such as converging to local minima and long iteration times. To address such problems one may use narrow-band algorithms and design an adaptive method to save the computational

cost [113]. Another popular choice is the use of “graph-cuts based” methods. This approach, based on min-cut/max-flow algorithms, has the merit that global minima are efficiently attained. In this chapter we apply graph cuts on both symmetric-difference based and on distance-field based measures and compare the results with the level-set implementation. For symmetric-differences-based functionals the graph cuts approach is generally faster than the standard level set implementation. When the graph-cuts method is used in conjunction with the active contour model, narrow band implementations are usually used to effectively eliminate the trivial global minima and reduce computational cost. As a matter of fact, the narrow band implementation is equivalent to solving a modified active contour model, which profitably combines the merits of symmetric- difference-based and distance-function-based measures. Moreover, instead of using uniform relative weight parameters, data dependent parameters are proposed, which enable our functional to preserve corners.

This chapter is organized as follows. In Section 2 we introduce functionals for curve smoothing and reconstruction problems. In Section 3 we briefly review the level set implementation and related numerical schemes. In section 3, we discuss the implementation of graph-cuts method. A description of the graph construction is briefly revised. In Section 4, we discuss the curve reconstruction application and its main challenges. In Section 5 several variations and many examples are presented. Finally, some concluding remarks are given in Section 6.

4.2 The Curve Smoothing and Reconstruction Functionals

The Rudin-Osher-Fatemi approach for signal and image denoising has had great success in recent years [91]. This denoising framework can be readily adapted to different situations. A particularly interesting scenario is curve, as opposed to function, smoothing: the noise is assumed to be a perturbation of some smooth underlying curve and the user might be interested in smoothing out the rough curve. Another

interesting problem is curve reconstruction which addresses the issue of reconstructing a curve from a set of unordered sample points without any prior knowledge except the fact that the points are noisy samples of a curve. These two problems are closely related and in this chapter we treat them as different instances of the same variational formalism.

The standard ROF approach to denoising is: given a noisy function f we seek a smooth version u by minimizing the cost functional

$$E_{ROF}(u, \lambda) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (u - f)^2 dx dy$$

where $\lambda > 0$ is a relative weight parameter to be selected. In this functional, the first term measures the smoothness of the solution, while the second term represents the similarity between given data and the solution. Following this idea, all curve denoising functionals will comprise the same basic components. As mentioned in [23], the area of symmetric difference of the indicator functions defined by the simple closed curves can be used to measure the distance between two curves. This functional is

$$E_1(C, \lambda) = \int_{\Omega} |\nabla \mathbf{1}_C| d\Omega + \lambda \int_{\Omega} |\mathbf{1}_C - \mathbf{1}_{C_0}| d\Omega \quad (4.1)$$

where Ω is the region of interest, $\mathbf{1}_C$ and $\mathbf{1}_{C_0}$ are the indicator functions of two curves, C_0 the given one and C the smoothed curve we seek. In this case, the second term is exactly the difference of the area enclosed “between” two curves while the first term is the perimeter of the smoothed curve we seek. The merit of this functional is that the optimal curves are global minimizers of the E_1 . As mentioned in the Introduction, this functional has been adopted in various applications, such as reconstructing 3D model from range image [117]. In fact, the indicator functions of the solution curve and the given curve in 4.1 can be replaced by signed distance functions as well [113], which is convenient for level-set implementation. However, this functional is not readily adapted to the problem of curve/surface reconstruction from point clouds, in which case indicating all interior and exterior points based on a non-existent curve/surface (an object that is being sought by us) is essentially impossible. Therefore, an ideal

functional that can handle these problems should rely only on the data points given and/or on partial topology information.

Provided that we represent a curve as a level-set of a bivariate function, and in particular via the distance function induced by itself in the plane, the distance between two curves can be written in the form,

$$dist(C^0, C) = \oint \varphi^0(C(\tau))|C_\tau|d\tau + \oint \varphi(C^0(\tau))|C_\tau^0|d\tau$$

where φ^0 is the distance function to a given “data” curve $C^0(\tau)$ and φ is the distance function to the expected smooth “output” curve $C(\tau)$. Considering only the first part of this distance measuring the closeness of $C(\tau)$ to $C^0(\tau)$ via his induced distance field φ^0 , we get the following functional

$$dist_{C^0}(C) = \int_0^1 \varphi^0(C(\tau))|C_\tau|d\tau,$$

Adding to the distance to C^0 a functional measuring the L_1 norm of the curve, i.e. $\int |C_\tau|d\tau$ (which is the curve length) yields

$$E_2(C, \alpha, \beta) = \int_0^1 (\alpha + \beta \varphi^0(C(\tau)))|C_\tau|d\tau \quad (4.2)$$

where α, β are parameters (constants or functions) to be selected. This functional is formally identical to the geodesic active contours model, the main problem of which is that global minimizers are isolated points trivial in most applications, the meaningful solutions corresponding to local minimizers. This is not a big problem for PDE-based methods which always converge to a local minimizer near properly selected initial solution, but would be a disaster for straightforward graph-cuts implementations which always yield the global minimizers.

It is interesting to note that, like in the image segmentation problem, for simply closed curve smoothing we have here two types of methods: the ones using a “region based” functional as given by (4.1); and an “edge based” functional, as exemplified by (4.2). The fact that each of them has advantages and drawback leads to the natural

idea of combining them together so as to combine the advantages of both functionals and jointly avoid their drawbacks. This leads to the consideration of

$$E_3(C, \alpha, \beta, \lambda) = \oint_C \alpha ds + \int_{\Omega} \lambda |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega \\ + \oint_C \beta \varphi^0(C) ds,$$

where $\mathbf{1}_{data}$ is the indicator function of the data set. It is worth pointing out that for curve smoothing problem $\mathbf{1}_{data}$ is just the indicator function of the given curve C_0 , and for the curve reconstruction problem the way of determining it depends on the choice of a narrow band whose detail will be given in the Section 5. Changing the line integral to area integral, we get

$$E_3(C, \alpha, \beta, \lambda) = \int_{\Omega} (\alpha + \beta \varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \\ \int_{\Omega} \lambda |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega. \quad (4.3)$$

If $\lambda = 0$, above functional becomes the one used in [61, 116]. When $\lambda = const$ and $\beta = 0$, it turns out to be the functional used in [117]. If λ has the form

$$\lambda(x) = \begin{cases} 0 & \text{if } \varphi^0(x) \leq c \\ \infty & \text{if } \varphi^0(x) > c \end{cases},$$

where c is a constant, this is the functional involved in the narrow-band implementation of the graph-cuts-based-method [61, 116]. It can be seen that various modifications can be incorporated in the functional (4.3), each of them just corresponding to a certain confidence function $\lambda(x)$, which “encodes” that to what extent the topology information is reliable. On the other hand, the different choices of the $\lambda(x)$ may result in different global minima for the same problem, e.g. the width of the narrow band affecting the final reconstruction result. Moreover, α, β, λ could be functions depending on the data curve. For example, the curves obtained by minimizing functional E_3 may lose meaningful corners. It hence would be natural to use a large similarity parameter in a corner region and small one elsewhere to remedy this problem.

Therefore, a simple choice is to let the similarity parameter or regularity parameter depend on certain cornerness measures of the data curve, e.g. a function of the local averaged curvature. Here, the functional would be

$$E_4(C) = \int_{\Omega} (\alpha_{data}(x) + \beta\varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda_{data}(x) |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega.$$

By proper choice of the weight functions $\alpha_{data}(x)$ and $\lambda_{data}(x)$, it will then be possible to preserve important features of curves. In section 4, some such choices will be discussed and implementation details will be given and the corresponding experiments will be shown in Section 6.

Standard level set methods can be applied to all functionals. In the next section we shall briefly discuss the possibility of various level-set implementations and for simplicity we shall use a constant relative weight parameter.

4.3 Graph-cuts for Global Minimization

In this section, a description of how to construct a graph where each cut represents a state of energy functional will be given.

Suppose $G = (V, E)$ is a directed graph with nonnegative edge weights that has two special vertices (called terminals), namely, a source s and a sink t . An s - t -cut $C = \{S, T\}$ is a partition of the vertices in V into two disjoint sets S and T such that $s \in S$ and $t \in T$. The cost of the cut is the sum of costs of all edges that go from S to T :

$$c(S, T) = \sum_{u \in S, v \in T, (u, v) \in E} c(u, v).$$

The minimum s - t -cut problem is to find a cut C with the smallest cost. Due to the theorem of Ford and Fulkerson this is equivalent to computing the maximum flow from the source to sink. Note that a cut is a binary partition of the graph and may be viewed as a labeling. Actually our problem can be thought of as a binary partition problem as well. This can be easily seen from the functional (4.1) in which the curve

is identified by the inside-outside indicator. Thus, to find a solution is equivalent to seek a partition of the graph nodes which represents the inside-outside indicator of the optimal curve. Using Cauchy-Crofton formula, it is also possible to translate GAC model to a binary problem, as shown in [17]. It is to be noted that this approach necessarily introduces some metrication errors of the magnitude of the order of grid size that will, however, not bother us too much. Therefore our aim is to construct the graph whose minimum cut or partition is the minimizer of the energy functional.

Let us consider a graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_N, s, t\}$. Each cut on G has some cost; therefore, G represents the energy function mapping from all cuts on G to the set of nonnegative real numbers. Any cut can be described by n binary variables $x_1, \dots, x_n$ corresponding to vertices in G (excluding the source and the sink): $x_i = 0$ when $v_i \in S$ and $x_i = 1$ when $v_i \in T$. Therefore, the energy functional which G represents can be viewed as a function of n binary variables: $\mathcal{E}(x_1, \dots, x_n)$ which is equal to the cost of the cut defined by the configuration $x_1, \dots, x_n$ ($x_i \in \{0, 1\}$).

4.3.1 Graph Construction of Energy functional

Normally, there are two types of edges in the graph: N-links and T-links. N-links connect neighboring nodes; T-links connect nodes with terminals. In this subsection we shall state how to use these two kinds of links to represent energy functional. Assume our energy functional has the following form as in [70],

$$\mathcal{E}(x_1, x_2, \dots, x_n) = \sum_i E^i(x_i) + \sum_{i,j} E^{i,j}(x_i, x_j)$$

The term E^i corresponds to similarity term or T-link. For instance, in the functional (4.1) after discretizing similarity integration by summation E^i has the form

$$E^i(x_i(v_i)) = |\mathbf{1}(v_i)_C - \mathbf{1}(v_i)_{C_0}|$$

where v_i is the grid point. Therefore, $E^i(x_i)$ either is 1 or 0.

While, the term $E^{i,j}$ corresponds to N-link and has the flavour of the smoothness term such as $\int_{\Gamma} ds$. There are several ways to view this term. Arc length can be

expressed by total variation of inside-outside indicator function of the curve

$$|\nabla \mathbf{1}| = \sqrt{(\mathbf{1}_x)^2 + (\mathbf{1}_y)^2}$$

since this term is not easy to be discretized by graph, the following anisotropic expression is used to measure the smoothness

$$|\nabla \mathbf{1}|_{ani} := (|\mathbf{1}_x| + |\mathbf{1}_y|) .$$

Note that for $|\nabla \mathbf{1}|_{ani}$ and $|\nabla \mathbf{1}|$ the following inequalities hold

$$\frac{\sqrt{2}}{2} |\nabla \mathbf{1}|_{ani} \leq |\nabla \mathbf{1}| \leq |\nabla \mathbf{1}|_{ani} ,$$

which imply $|\nabla \mathbf{1}|_{ani}$ is a proper smoothness measure. If we use finite difference scheme to discretize $|\nabla \phi|$ on uniform grids we get an isotropic discretization

$$|\nabla \mathbf{1}| \approx \frac{1}{h} \sqrt{(\mathbf{1}_{m,n} - \mathbf{1}_{m,n-1})^2 + (\mathbf{1}_{m,n} - \mathbf{1}_{m-1,n})^2}$$

and the corresponding anisotropic discretization is given by

$$|\nabla \mathbf{1}|_{ani} := \frac{1}{h} (|\mathbf{1}_{m,n} - \mathbf{1}_{m,n-1}| + |\mathbf{1}_{m,n} - \mathbf{1}_{m-1,n}|) .$$

In this case the term $E^{i,j}$ has form $|\mathbf{1}_i - \mathbf{1}_j|$ where i and j are indices of the nodes and it can be represented by a bi-directed edge between the node i and j . Above observation is only valid for 4-neighbourhood. Of course, we can use higher order finite difference scheme or unstructured grids to approximate $|\nabla \mathbf{1}|$ and get corresponding anisotropic smoothness term $|\nabla \mathbf{1}|_{ani}$, but this will involve three or more terms. Actually, by means of the Cauchy-Crofton formula discretization can be also carried out by dividing angles. The reader is referred to [17] for more discussions on discretizing the arc-length on other neighbourhood systems. Figure 4.2 is an example of a complete 3 by 4 graph

Figure 4.2: An example of graph construction.

of the functional (4.1). In this graph the given curve has characteristic function

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

As shown in Figure 4.2, the nodes which have function value 1 connect to s with an edge of capacity λ while the nodes whose function value are 0 have an edge with t . The bi-directed edges between grids points having capacity 1 represent the $E^{i,j}$.

Turning to geodesic active contour functional (4.2), it is just the shortest path under the Riemann metric and has the form

$$\psi(C(\tau)) = \oint \sqrt{[x', y'] G [x', y']^t} d\tau$$

where

$$G = \begin{bmatrix} (\lambda\varphi^0(C(\tau)) + 1)^2 & 0 \\ 0 & (\lambda\varphi^0(C(\tau)) + 1)^2 \end{bmatrix}.$$

Thus by Cauchy-Crofton formula [17] the weight of N-links in GAC functionals is just multiplied by $(\lambda\varphi^0(C(\tau)) + 1)$. Therefore, for functionals (4.2) and (4.3) the N-link

edges control both similarity and smoothness terms, and T-links just play a role like area constraints or boundary condition.

4.3.2 Min-cut algorithm

The efficiency of the min-cut algorithm is crucial for our applications. In combinatorial optimization there are several polynomial algorithms for min-cut/max-flow. Most of the algorithms fall in one of the following groups: algorithms based on Ford-Fulkerson style “augmenting paths” [50] and Goldberg-Tarjan style “push-relabel” methods [57]. Standard augmenting paths based algorithms, such as Dinic algorithm [32], work by pushing flow along non-saturated paths from the source to the sink until the maximum flow in the graph is achieved. An important fact in Dinic’s algorithm is the use of breadth-first search to find the shortest paths from s to t on the residual graph, which significantly improves the theoretical running time. The worst case running time complexity for Dinic’s algorithm is $O(mN^2)$ where N is the number of nodes and m is the number of edges in the graph. By clever implementation, “push-relabel” algorithm can be improved to $O(N^3)$. Although the worst complexity of min-cut algorithm is the same with explicit level-set implementation, this worst case is seldom attained in our application. Actually, in all our examples the min-cut algorithm attains the order $O(N^\alpha)$ where $1 \leq \alpha \leq 2$.

In push-relabel algorithms a pre-flow is maintained during the operation. Besides, the algorithms maintain a labeling of nodes giving a low bound estimate on the distance to the sink along non-saturated edges. The distances labels progressively increase as edges are saturated by push operations. Push-relabel algorithms have the same theoretical complexity with “augmenting paths” style algorithms. Recently, Boykov and Kolmogorov [18] developed a new algorithm based on an augmenting paths technique and a new BFS method, in which the search tree is not constructed from scratch every time. Although their method has a worse theoretical complexity $O(mn^2|C|)$ ($|C|$ being the cost of the minimum cut), it proves to be more efficient in many image processing applications. In this chapter the standard push-relabel algorithm is used.

4.3.3 Implementation details

There are several implementation issues which need to be pointed out. The first one is the way to tag interior and exterior regions. For the functional (4.1), we need characteristic functions to represent curves. This is not an issue if curves come from segmentation results since partition is already there. However, most often curve is defined by an ordered set of points. In this case the curve is linearly approximated by a polygon. Then our problem becomes a classic computational geometry problem: determining whether a point is in a polygon. Two rather different methods for solving this problem are available: counting ray crossings and computing “winding” numbers. Because of its simplicity and efficiency we use the ray crossing method. Basic idea of ray crossings is that if a point p is in the polygon a ray from p will intersect polygon boundary odd times otherwise even times. If data set is unordered points it is hard to locate the boundary of the curve by any means and such ray crossing computation is impossible in this scenario. Therefore, we conclude that the curve based functional (4.1) is not appropriate for curve reconstruction.

The second issue is about computing distance function. Since for the functional (4.2) and (4.3) we need compute the distance function corresponding to the given curve. Again, if data is defined by an ordered point set then the simplest way to do this is for each grid point computing the distance to all segments and setting the smallest as the final distance. This method does not work for unorganized points and is very slow if points set is large. Generally, we can use a sweeping method [118, 90] to compute distance function. This method has complexity $O(N)$ for uniform grids and $O(N \log N)$ for unstructure grids, where N is the grids number.

Since the global minimum for the GAC model (4.2) corresponds to a point, direct application of the graph-cuts method will always yield a point which makes no sense in our application. To eliminate this trivial solution, certain Dirichlet boundary conditions [16] need to be imposed. Since desired curve is always near the given curve or data points, it should fall in a narrow band around the given curve or data points as illustrated in Figure 4.3. The choice of the bandwidth δ depends on the noise level and the resolution. Based on these observations, [61, 116] used a narrow-band implementation: (a) If the point is in the narrow band, the weight of the n-link

Figure 4.3: Dashed is the band boundary; solid line is the given curve. The distance of the point in the shadow region to given curve is less than δ . Candidate curve should be bounded by the shadow region.

is set as usual; set the weight of the n-link to be infinity or a large number in practice. (b) If point is not in the narrow band and it is in the interior of the curve for sure. Connect the point with s and set the T-link to be a large number. On the other hand, connect points which are certainly outside the curve with t and set T-link to be a large number as well.

Actually, seeking minimum s-t-cut on the graph described above means minimizing functional (4.3)

$$E_3(C, \alpha, \beta, \lambda) = \int_{\Omega} (\alpha + \beta \varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega.$$

For curve smoothing problem $\mathbf{1}_{data}$ can be computed by ray crossing method, and relative weight function $\lambda(x)$ can be just a constant. For curve reconstruction problem, $\mathbf{1}_{data}$ and $\lambda(x)$ need special treatments, which will be discussed in Section 5.

4.3.4 Corner preserving strategy

Next, we present our corner-preserving strategy. To preserve corners the first concern will be designing a good corner detector. Estimating the discrete curvature could yield a good corner detector, but note that the raw data is contaminated by noise, hence point-wise evaluation or approximation of the curvature is not reliable. Inspired

by the neighborhood idea proposed in [110], an *averaging technique* is utilized in this chapter to design a robust corner detector.

For an arbitrary polygon $P = \{P_1 P_2 \dots P_n\}$, denote I_P by its indicator function and A_{x_i} by the angle at point P_i . Then the following thing is true,

$$A_{x_i} = \lim_{t \rightarrow 0} \int_{R^2} I_P \bullet G_t(x_i, y) dy$$

where $G_t(x_i, y) = \frac{1}{2t} \exp(-\frac{\pi \|y - x_i\|^2}{2t})$. When t is small enough, it is reasonable to use following approximation

$$A_{x_i} \approx \int_{R^2} I_P \bullet G_t(x_i, y) dy$$

Moreover, for a curve/surface approximated by linear polygon/polyhedra, the following equality holds

$$\sum_i (\pi - A_{x_i}) = 2\pi,$$

therefore, defect angle $\pi - A_{x_i}$ is used to approximate curvature in many cases. notice that

$$\int_{R^2} (1 - I_P) \bullet G_t(x_i, y) dy = 2\pi - A_{x_i},$$

thus

$$\begin{aligned} \int_{R^2} (1 - I_P) \bullet G_t(x_i, y) dy - \int_{R^2} I_P \bullet G_t(x_i, y) dy \\ = 2\pi - A_{x_i} - A_{x_i}. \end{aligned}$$

Therefore the measure of cornerness can be obtained as

$$\tau(x_i) = \frac{1}{2} \int_{R^2} (1 - 2I_P) \bullet G_t(x_i, y) dy.$$

This approximation can overcome the influence of the noises, if variance of the Gaussian is large enough to cover the noise level. In other words, by tuning parameter t , measure $\tau(x_i)$ is able to characterise corners in different scales. For a given t , the larger $|\tau(x_i)|$ means the shaper corner in the scale t . However, this formula is computational expensive since when the variance of Gaussian function is large the

computation has to be carried on whole domain so as to evaluate the integral accurately. Note that the above quantity actually only measures how different a point is from being on a straight line which has an angle π . Therefore the quantity can also be,

$$\hat{\tau}(x_i) = \frac{|\int_{R^2} G_t(x_i, y) dy|}{\min(g_1, g_2)},$$

where

$$g_1 = |\int_{R^2} (1 - I_P) \bullet G_t(x_i, y) dy|,$$

and

$$g_2 = |\int_{R^2} I_P \bullet G_t(x_i, y) dy|.$$

$\hat{\tau}(x_i)$ has value 2 for straight line and becomes larger when the corner becomes shaper. The integration is replaced by summation and R^2 is replaced by the domain of interests Ω ,

$$\hat{\tau}_h(x_i) = \frac{|\sum_{y \in \Omega} G_t(x_i, y) \delta y|}{\min(g_1, g_2)},$$

where

$$g_1 = |\sum_{y \in \Omega} (1 - I_P) \bullet G_t(x_i, y) \delta y|,$$

and

$$g_2 = |\sum_{y \in \Omega} I_P \bullet G_t(x_i, y) \delta y|.$$

Furthermore, in practical experiments $G_t(x)$ can be simply replaced by 1 and the summation can be only computed in a small neighborhood $\mathcal{N}$ which is determined by the variance or the noise level while numeric errors in numerator and denominator are somewhat cancelled. Thus, in practice the formula may be,

$$\hat{\tau}_h(x_i) \approx \frac{|\sum_{y \in \mathcal{N}} I_P(y)|}{\min(|\sum_{y \in \mathcal{N}} (1 - I_P(y))|, |\sum_{y \in \mathcal{N}} (I_P(y))|)}.$$

By this formula $\hat{\tau}_h$ is equal to infinity for the points inside or outside the curve, and therefore to remedy this problem all points not close to the curve take the value 2 the same as the points on straight line. We use the following Algorithm to compute

corner measure $\hat{\tau}_h$.

Algorithm (Determination of $\hat{\tau}_h(x)$): For every node x_i in the computational domain Ω :

1. choose a reference circle $\mathcal{N}(x_i, r)$ centered at x_i with a radius r ;
2. count the number of nodes, $N = N^+ + N^-$, contained inside the reference circle $\mathcal{N}(x_i, r)$ where N^+ denotes the number of nodes with indicator value 1 in $\mathcal{N}(x_i, r)$ and N^- that with indicator value 0. Set $N_{min} = \min\{N^+, N^-\}$;
3. if $N_{min} = N^+$ and $I(x_i) = 1$ or $N_{min} = N^-$ and $I(x_i) = 0$, then set $\hat{\tau}_h(x_i) = N/N_{min}$, otherwise, $\hat{\tau}_h(x_i) = 2$;

Once the “corneriness” measure is available the λ in the the functional

$$E_4(C) = \int_{\Omega} (\alpha + \beta \varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda_{data}(x) |\mathbf{1}_C - \mathbf{1}_{data}|^2 d\Omega,$$

can be set to be large in regions near corners and small in flat regions. Thus, we specify a non-homogeneous “attraction potential” making corners more attractive. In our study, $\lambda(x)$ was taken to be,

$$\lambda(x_i) = \begin{cases} \lambda_1 & \text{if } f(\hat{\tau}_h) \leq 0.4 \max(f(\hat{\tau}_h)) \\ \lambda_2 & \text{if } f(\hat{\tau}_h) > 0.4 \max(f(\hat{\tau}_h)) \end{cases} \quad (4.4)$$

$$f(s) = e^{w(2-s)^2}, \quad (4.5)$$

where λ_1 , λ_2 , and w are some parameters. The reason that $f(s)$ is selected as above is because it changes rapidly in corner regions and is able to make the corner more distinguishable. Once these parameters are selected, the $\lambda(x)$ is determined.

However, the above-mentioned computation relies on the indicator function which is not available for unordered points. In the curve reconstruction application the corner detector should only be based on geometric information residing in the data points. Thus, the approach described below outlines an alternative corner detector.

Let $P = \{P_1, P_2, \dots, P_n\}$ denote the set of unordered points, where order does not imply connectivity. Provided the point set is sampled from a closed curve and dense enough, for each point P_i the neighbourhood $\mathcal{N}(P_i, r)$ should contain some points P_j different from P_i , where $\mathcal{N}(P_i, r)$ is the circle centred at P_i with radius r . Define the average direction vector as,

$$\vec{v}(P_i) = \frac{1}{N} \sum_{P_j \in B(P_i, r) \text{ and } P_j \neq P_i} \frac{P_j - P_i}{|P_j - P_i|}, \quad (4.6)$$

where N is the number of points in the circle $B(P_i, r)$. When P_i 's are on a straight line $\vec{v}(P_i)$ will be close to zero vector, and as P_i is a corner point $\vec{v}(P_i)$ points to the direction of the angle bisector. Therefore, the cornerness measure for P_i can be defined as the length of the average direction vector,

$$\tau_2(P_i) = |\vec{v}(P_i)|. \quad (4.7)$$

Moreover, if r is large enough this measure can suppress the influences of noises as well. Utilizing this cornerness measure, the corner preserving strategy for reconstruction is either to decrease regularity parameter α around detected corners or modify distance field induced by points set. There are several ways to implement these ideas, the precise choice of which is immaterial so long as corners are made more attractive (an illustrative example is shown in Section 6).

4.4 Application to Curve Reconstruction from Scattered Points

As we have seen curve reconstruction from an unordered set of points is an interesting and challenging problem, and there is no well defined, unique solution for this problem. To construct a curve that is a good approximation of the data set, a reconstruction procedure should be able to deal with complicated topologies and geometries as well as noise and nonuniformity of sampling. This problem is closely related to curve smoothing, since both problems need a proper distance measure. The functional

(4.1) measures distance between two curves by area difference. As we pointed out before this is a proper distance measurement for the case of curve smoothing but in the reconstruction scenario an indicator function becomes meaningless since we can not completely mark the interior and exterior regions. In this case, the topology information is only available for the region away from the point sets, and near the point sets only the geometry or distance information is readily usable. Hence the functional (4.2) and (4.3) which only rely on distance information or partial topology information can handle curve reconstruction problem. As discussed in Section 2, the modified functional (4.3) is more stable than (4.2), and can be solved by graph-cuts techniques. To address reconstruction problem by functional (4.3), the first issue is to compute the function $\lambda(x)$. In practice, once a band with width c is chosen then the weight function λ is given by

$$\lambda(x) = \begin{cases} 0 & \text{if } \varphi^0(x) \leq c \\ \infty & \text{if } \varphi^0(x) > c \end{cases}, \quad (4.8)$$

where φ^0 is the unsigned distance function induced by point set. Meanwhile, the band defined by the set $S_{band} = \{x | \varphi^0(x) \leq c\}$ separates the whole domain into two or more disconnected subdomains, in each of which the indicator function can be determined by region growing methods. For the points belonging to S_{band} , the indicator function can be set to an arbitrary value since $\lambda(x)$ is zero for them, and the term $\lambda(\mathbf{1}_C - \mathbf{1}_{data})^2$ will add nothing to the energy cost. Essentially, this is the narrow band implementation used in various applications. As described in Section 4.4, based on proposed corner detectors (4.7) it is also possible to develop corner preserving strategy for curve reconstruction. A simple way is to decrease regularity parameter α around the detected corners. Because τ_2 in (4.7) is defined on the data points not on grids points, one needs impose corner information on grid points by locating the nearest grid points x_i . Thus, $\alpha(x)$ may have the form,

$$\alpha(x_i) = \begin{cases} \alpha_1 & \text{if } x_i \text{ is in the corner regions} \\ \alpha_2 & \text{otherwise} \end{cases} \quad (4.9)$$

where α_1 and α_2 are some constants to be selected. Another way is to modify the distance field around detected corners. For example, the distance field can be generated by a function $(\frac{dist}{c})^n$ in a neighbourhood of the detected corners, where c is the constant used to define $\lambda(x)$ in (4.8). In both cases, the solution curve inclines to pass through detected corners in order to decrease energy.

4.5 Numerical Tests

All the experiments reported herein were performed on a PC with Intel Xeon CPU of 3.33GHz and 4GB memory. To do graph cuts we use the push relabel algorithm, see [57].

The first example is a noisy circle as shown in Figure 4.4-(a). The curve is smoothed using the functional (4.1)

$$E_1(\phi, \lambda) = \int_{\Omega} |\nabla \mathbf{1}_C| d\Omega + \lambda \int_{\Omega} |\mathbf{1}_C - \mathbf{1}_{C_0}| d\Omega,$$

functional (4.2)

$$E_2(C, \alpha, \beta) = \int_0^1 (\alpha + \beta \varphi^0(C(\tau))) |C_\tau| d\tau,$$

and functional (4.3)

$$E_3(C, \alpha, \beta, \lambda) = \int_{\Omega} (\alpha + \beta \varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega.$$

First, three functionals are minimizing by the level set method on a 400×400 uniform mesh. Here, forward difference is used for time discretization, time step being 1×10^{-6} and reinitialization is performed every 50 steps. Figure 4.4-(b) shows the smoothed circle obtained by minimizing functional (4.1) and the comparison with the noisy circle. Figure 4.4-(c) shows the results minimizing functional (4.2) and Figure 4.4-(d) presents the results minimizing the functional (4.3). It can be observed that the

	200 Resolution	400 Resolution
Level set method	117.12s	1326.23s
Graph cuts	1.9s	7.9s

Table 4.1: CPU time (in seconds) of the level set method and the graph cut method for the noisy circle example.

result in Figure 4.4-(b) is slightly smoother than the result in Figure 4.4-(c), and the result in Figure 4.4-(d) is more like the result in Figure 4.4-(b) because of the choices of the parameter λ and β . For comparison the results of the graph cut method and the level set method are presented together in Figure 4.5, in which both smoothed curves are obtained by minimizing the functional (4.3) with the same parameters as the one in Figure 4.4-(d) and 32-neighbourhood system is used for the graph cut method. It can be observed that under the same resolution, the level set method presents better result than the graph cuts method because of the metrication errors of graph-discretization. However, the computation time of the level set method is much longer than the graph cut method, as can be clearly seen from the statistics of CPU time listed in Table 4.1.

The following example considers a noisy triangle as shown in Figure 7, where the results obtained by minimizing three different functionals are illustrated. In this example the graph cuts method is applied on 800*800 grids with a 64 neighbor system. Figure 7-(b) presents the result of the functional (4.1) and Figure 7-(c) gives the result of the functional (4.2). It can be observed that two smoothed curves have almost the same quality. As discussed before, to solve the functional (4.2) by graph cut, the narrow band implementation is needed. The width of the band is selected to be 0.2 in this example. Table 4.2 lists the CPU times consumed by minimizing functionals (4.1) and (4.2) with different neighbor systems, from which we can see that to minimize the functional (4.2) is slightly faster than the functional (4.1). Figure 7-(d) shows the smoothed curve produced by the functional (4.3), which quite resembles the result in Figure 7-(c) because here the parameter β is dominated. However, to solve the functional (4.3) there is no need for narrow band implementation and the CPU time is almost as same as the functional (4.1).

In order to quantitatively compare above results, we use the following quantity to

(a) Noisy Circle

(b) The result minimizing functional 1 and the comparison of the noisy curve and smoothed curve

(c) The result minimizing functional 2 and the comparison of the noisy curve and smoothed curve

(d) The result minimizing combined functional 3 and the comparison of the noisy curve and smoothed curve

Figure 4.4: The resulting curves smoothed for the same data and three different functionals. (b) The smoothed curve with parameters $\lambda = 0.05$ and $\alpha = 1$ in functional (4.1). (c) The smoothed curve with parameters $\beta = 5$ and $\alpha = 1$ in the functional (4.2). (d) The smoothed curve with parameters $\lambda = 0.05$, $\beta = 3$ and $\alpha = 1$ in functional (4.3).

(a) Smoothed Circle produced by the graph-cut method

(b) Smoothed Circle produced by the level set method

Figure 4.5: The first column contains the smoothed curve and the second column includes the mixture of the smoothed curves and noisy curves; parameters in functional (4.3) are $\alpha = 1$, $\beta = 3$ and $\lambda = 0.05$ and 32 neighbor system is used.

Neighborhoods	Time	
	Functional 1	Functional 2
4	11.32s	9.96s
16	28.73s	23.11s
64	86.31s	85.16s

Table 4.2: CPU time of graph cuts results of two models with different neighborhoods. Resolution is $800 * 800$ and similarity parameters in two model (1) and (2) are 0.02 and 4 respectively while regularity parameter is $\alpha = 1$ for both models.

(a) Noisy triangle

(b) The smoothed curve produced by functional 4.1

(c) The smoothed curve produced by functional 4.2

(d) The smoothed curve produced by functional 4.3

Figure 4.6: A noisy triangle example. From the second row to the last row, the first column contains the smoothed curves and the second column includes the mixture of the smoothed curves and noisy curves; parameters in the functional (4.1) are $\lambda = 0.03$ and $\alpha = 1$; parameters in the functional (4.2) are $\beta = 4$ and $\alpha = 1$ while narrow band has width 0.2. In model (4.3) $\lambda = 0.02$, $\beta = 4$, and $\alpha = 1$. 64 neighbor system and $800 * 800$ resolution are used in all methods.

Figure 4.7: A noisy annulus example. The first figure presents the noisy curve and the second figure shows the result minimizing model (4.3); 64 neighborhoods are used and similarity parameter $\lambda = 0.06$, $\beta = 4$ and $\alpha = 1$.

measure the difference of the smoothed curve and the original curve:

$$D_{C_0}(C) = \frac{\sum |\mathbf{1}_C - \mathbf{1}_{C_0}|}{\sum \mathbf{1}_{C_0}} \quad (4.10)$$

where C_0 is the original curve and C is a noisy curve or smoothed curve. This normalized area difference measures the relative difference of C and C_0 , since the smaller it is, the closer two curves are.

The next example is a noisy annulus. The result is shown in Figure 4.7. To do this example with the functional (4.2) one can let points between two circles be connected with source and both boundary of the domain and the center of the inner circle be connected with sink. In our implementation the T-links are automatically determined by region growing method.

The next examples consider noisy curves with more complicated geometry. Hereafter all examples will be implemented based on the functional (4.3) and $\alpha = 1$ unless specified otherwise. It can be observed from Figure 4.8 that the smaller the similarity parameter is chosen, the smoother curve is obtained but also the more details are lost. We remark that with different parameters the computation costs of the algorithm are somewhat different. Table 4.3 gives the CPU time consumed during smoothing Figure 4.8 with different parameters and narrow bands. Here we only use 64 neighborhoods within narrow band and 4 neighborhoods outside the narrow band. It can be observed from Table 4.3 that this adaptive implementation considerably

Figure 4.8: An example with complicated geometry; 64 neighborhoods are used and $\beta = 0.0$ in all results but with different parameter λ . (a) the noisy curve; (b) the curve smoothed with $\lambda = 0.2$; (c) the curve smoothed with $\lambda = 0.1$; (d) the curve smoothed with $\lambda = 0.05$.

reduced the computation cost. However, narrow band should be wide enough so that at least solution curve is included.

The example in Figure 10 is formed by a circle intersecting a noisy rectangle. This is a more challenging example. From Figure 10 we can see that circle and rectangle separate when the similarity parameter is decreased and since the intersection regions for the two shapes are very small, it is hard to determine the correct local topology.

The “Squirrel surrounded by grass” example in Figure 4.10 tests the smoothing of curves resulting from image segmentation. The direct image segmentation results have a lot of small features and the boundary of the object often has zig-zags. It can be observed that in the last picture of Figure 4.10, all the insignificant features are removed by our method.

The main disadvantage of the functional (4.1) and (4.2) is that resulting curves may lose meaningful corners. Therefore, the functional (4.4) is used to handle the

λ	Time	
	Bandwidth <1	Bandwidth <0.5
0.2	73.11s	35.45s
0.1	76.12s	41.86s
0.05	111.43s	58.24s

Table 4.3: CPU time with different parameters under $800 * 800$ resolution. 64 neighborhood is chosen within narrow band.

Figure 4.9: An example formed by a noisy circle intersected a noisy rectangle; 64 neighborhoods are used and $\beta = 0.0$ in all results but with different similarity parameters λ . (a) the noisy curve; (b) the result with $\lambda = 0.2$; (c) the result with similarity $\lambda = 0.1$; (d) the result with $\lambda = 0.05$.

Figure 4.10: Image segmentation result. (a) Original image; (b) segmentation result; (c) boundary curve of the segmentation; (d) smooth curve processed by our method.

examples with sharp corners,

$$E_4(C) = \int_{\Omega} (\alpha + \beta \varphi^0(C)) |\nabla \mathbf{1}_C| d\Omega + \int_{\Omega} \lambda_{data}(x) |\mathbf{1}_C - \mathbf{1}_{data}| d\Omega,$$

where $\lambda(x)$ is computed as described in Section 5.1. The example in Figure 4.11 is a noisy eight-sided star, which has obvious sharp corners. Figure 4.11-(c) shows the smoothed curve with constant similarity parameters $\lambda = 0.5$ and $\beta = 5$. Although all corners are preserved, the curve is not smooth enough. By decreasing λ to 0.03 the curve becomes much smoother but also loses all corners as shown in Figure 4.11-(d). However, using $\lambda(x)$ computed by the method described in Section 5.1 the result presented in Figure 4.11-(e) is smooth enough as well as preserves all sharp corners. The contour of the function $\lambda(x)$ is also illustrated in Figure 4.11-(f), which clearly demonstrates that all corners are correctly identified. In order to quantitatively compare above results, we use the following quantity to measure the difference of the smoothed curve and the original curve:

$$D_{C_0}(C) = \frac{\sum |\mathbf{1}_C - \mathbf{1}_{C_0}|}{\sum \mathbf{1}_{C_0}} \quad (4.11)$$

where C_0 is the original curve and C is a noisy curve or smoothed curve. This normalized area difference measures the relative difference of C and C_0 , since the smaller it is, the closer two curves are. Let C_0 denote the original curve shown in Figure 4.11-(a) and C_1, C_2, C_3 , and C_4 denote the curves shown in Figure 4.11-(b)–(e) respectively. The normalized area differences for them are $D_{C_0}(C_1) = 0.0641$, $D_{C_0}(C_2) = 0.0542$, $D_{C_0}(C_3) = 0.1640$, $D_{C_0}(C_4) = 0.0231$ respectively. C_1 is the noisy curve, $D_{C_0}(C_1)$ actually measures the noise level and it is 0.0641 for this example. C_2 is the smoothed curve as shown in Figure 4.11-(c), from which it can be seen that C_2 is a reasonable approximation of the original curve, and this is also justified by the fact $D_{C_0}(C_2) = 0.0542 < 0.0641$. Although C_3 in Figure 4.11-(d) is very smooth, it is not a good approximation since visually it loses some important features of the original curve and quantitatively it is not close to the original curve $D_{C_0}(C_3) = 0.1640$. C_4 is

Curves	C_1	C_2	C_3	C_4
Area Difference	0.0641	0.0542	0.1640	0.0231

Table 4.4: Comparison of the normalized area difference .

an excellent approximation of the original curve, which can be seen from the figure clearly and is proven by the normalized area difference as well, $D_{C_0}(C_4) = 0.0231$, the smallest value among these curves. This example shows that our corner preserving strategy indeed works well and a relatively complex example will be presented next.

The example in Figure 4.12 is a result from image segmentation, a logo of “NTU”. Figure 4.12-(a)&(b) present segmentation result and the corresponding boundary curve. Figure 4.12-(c) shows the smoothed curve with constant similarity parameters $\lambda = 0.03$ and $\beta = 5$. It can be observed that not only the corners are missed but also the elongated parts in letter “N” and “U”. Figure 4.12-(d) presents the result with the varying similarity function, which is smooth and preserves most important features. Again, one can see the correctly identified important features from the contour of the function $\lambda(x)$ shown in Figure 4.11-(e).

The next three examples are the results of curve reconstruction, where bandwidth is chosen as $c = 0.1$ and the similarity parameter $\beta = 20$. Figure 4.13 is the curve recovered from points which have strong noise at certain parts. We can observe that the noisy clouds are gracefully ignored by our method. Figure 4.14 depicts a cloud of points outlining an elephant shape. The point cloud is very noisy but our method recovers a meaningful result.

Figure 4.15 shows a cloud of points sampled from a Chinese character which has obvious corners. In this example, the proposed corner preserving strategy is clearly demonstrated. Figure 4.15-(a) gives the set of noisy points. Figure 4.15-(b) illustrates the reconstructed curve with constant regularity parameter $\alpha = 1$ while Figure 4.15-(c) presents the reconstructed curve with with regularity parameter $\alpha = 0.01$ around corner regions and $\alpha = 1$ otherwise. Figure 4.15-(d) gives the comparison of two reconstructed curves, clearly manifesting that the curve reconstructed by the proposed corner preserving strategy has sharper corners and better visual effects. Figure 4.15-(e) shows the contour of the proposed corner detector (4.7), in which the locations of

all corners are correctly identified.

Furthermore, by choosing different distance fields, the variational model proposed in this paper can actually deal with a large range of applications. The example shown in Figure 4.16 demonstrates that our method can readily produce the shortest path within a bounded region between two curves. In this example, the distance field is generated by formula

$$D = \left(\frac{dist}{\mathbf{B}} \right)^n$$

where $\mathbf{B}$ is the width of the band which is here set to 0.05, $dist$ is the unsigned distance function and n is an integer constant. We see that bigger n yields smoother curves. Figure 4.17 and ?? are two examples which show that our method can determine the shortest polygonal path in a bounded region defined between two nested polygons.

As an interesting generalization, the distance field naturally solves heteroscedastic probabilistic curve reconstruction problems [66], as described below. Consider the samples of a planar curve corrupted by independent additive Gaussian noises whose covariance matrices vary according to the location of the sample points in space as follows:

$$\vec{v} = \vec{v}_0 + n,$$

where $n \sim \mathcal{N}(0, \Sigma_{\vec{v}_0})$ is an independent Gaussian noise vector with zero mean and point-specific covariance ($\vec{v} = (x, y)$, $\vec{v}_0 = (x_0, y_0)$). Therefore, for a given sample point $\vec{v}_0$, the position of point $\vec{v}$ is a random vector with Gaussian distribution $\mathcal{N}(\vec{v}_0, \Sigma_{\vec{v}_0})$. Provided the covariance matrix is given for each point, the question is to find the original point (x_0, y_0) given that (x, y) is observed. Thus, the following conditional density function is of interest,

$$p(\vec{v}_0|\vec{v}) = \frac{p(\vec{v}|\vec{v}_0)p(\vec{v}_0)}{\int_{R^2} p(\vec{v}|\vec{v}_0)p(\vec{v}_0)dx_0dy_0} \quad (4.12)$$

describing the distribution of the curve sample given the observation point $\vec{v}$. Noticed that, for each possible (x_0, y_0) and a fixed sample point (x, y) , the density function

$p(\vec{v}|\vec{v}_0)$ is a function of (x_0, y_0) proportional to,

$$\frac{1}{\sqrt{(2\pi)|\Sigma_{\vec{v}_0}|^{1/2}}} \exp\left(-\frac{1}{2}(\vec{v} - \vec{v}_0)' \Sigma_{\vec{v}_0}^{-1} (\vec{v} - \vec{v}_0)\right).$$

Without prior knowledge about the original curve, it is reasonable to assume that $p(\vec{v}_0)$ in the formula (4.12) is a uniform distribution in a bounded region, which means each point in the region is equally likely to be a point on the curve that is sampled. Under this assumption, the conditional density function generated by an observed point can be computed by formula (4.12), in which $p(\vec{v}|\vec{v}_0)$ is a function of (x_0, y_0) , $p(\vec{v}_0)$ a constant, and $\int_{R^2} p(\vec{v}|\vec{v}_0)p(\vec{v}_0)dx_0dy_0$ a scaling.

After the conditional density function is determined for each sample point, the influence potential of the points $\{\vec{v}_i\}$ can be defined as

$$A(x, y) := \frac{1}{T} \sum_i p((x, y)|\vec{v}_i), \quad (4.13)$$

where $\vec{v}_i$'s are the T sample points. This density function describes how likely points in the plane are the original samples of a curve given points we observe. Therefore, the problem of finding the original curve becomes a process of seeking curves having the largest cumulative probability, which is equivalent to the following optimization problem

$$\max_{C(x,y)} \oint_C A(x, y) ds. \quad (4.14)$$

By adding a constant and changing the sign of the function, the maximization problem (4.14) is changed to the minimization problem (4.15)

$$\min_{C(x,y)} \oint_C \psi ds, \quad (4.15)$$

where

$$\psi = \max(A(x, y)) - A(x, y).$$

Since ψ is non-negative, functional $\oint_C \psi ds$ looks exactly like the GAC functional

discussed before, in spite of the fact that the “distance” field is now defined by the probability density function instead of geometric distance. In contrast to the simple geometric approach, the probability density function is able to incorporate prior knowledge on the noise distribution. For example, assuming that $\vec{v}_0$ uniformly distributes in a band around the sample points, the resulting density function will be consistent with narrowband graph-cuts implementation. Furthermore if the location of the point $\vec{v}_0$ is roughly known, one can assume $\vec{v}_0$ is subject to a Normal distribution with $\vec{v}_0$ as mean. We next present several examples to demonstrate the capability of the above proposed framework to nicely handle heteroscedastic cases.

The examples considered assume that each sample point generates a probability or influence field, and the whole influence field is of form (4.13). In the sequel, a influence function defined in (4.13) will be called “sum probability”. In the examples considered the 2-D Gaussian distributions around each point are given by

$$\frac{1}{\sqrt{2\pi|\Sigma_{\vec{v}_0}|}} \exp\left(-\frac{1}{2}(\vec{v} - \vec{v}_0)^T \Sigma_{\vec{v}_0}^{-1} (\vec{v} - \vec{v}_0)\right)$$

where $\vec{v}$ is the observed point and $\vec{v}_0$ is the grid point. The covariance matrix Σ is given by

$$\Sigma_{\vec{v}_0} = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}.$$

Our first example shown in Figure 4.18 is an L-shape polygon. In this example the following spatially varying covariance is used to generate the noisy sample points $\sigma_x = 0.05 + |\sin(5\pi x)|/30$, $\sigma_y = 0.05 + |\cos(5\pi y)|/30$, $\rho = (x^2 + y^2)/4$. The first row of Figure 4.18 includes the original curve and the noisy samples. It can be seen that the noise varies at different locations. The second row shows the curve obtained by maximizing the sum probability.

The example in Figure 4.19 is of a curve with very noisy samples. The first row of the Figure 4.19 contains the original curve and the noisy samples. The influence field generated can be observed in Figure 4.19-(c). The result shown in 4.19-(d)&(e) show that the noise effect is removed. It is clear that the variance of Gaussian distribution

drastically changes from point to point. This means that we trust more the data-points in certain regions. For example in Figure 4.19-(d) we can observe that at point P the curve is far from the corner because of large variance and at point Q the curve is close to the corner because of small variance. We note that instead of using homogeneous and isotropic distance fields, inhomogeneous and anisotropic distance fields allow us much more flexibility in modelling the effects of noise and in incorporating prior knowledge.

4.6 Concluding Remarks

In this chapter we survey general variational frameworks for curve smoothing and reconstruction problems. By combining the merits of functionals (4.1) and (4.2), the functional (4.3) is able to gracefully handle curve reconstruction and smoothing within the same process, and will be the general framework used in the next two chapter as well. To preserve corners the idea developed in the last chapter that similarity parameters or regularity parameters can be made dependent on “corneriness measures” is generalized. For smoothing application the corneriness measure proposed in the last chapter is reused and a mathematical justification is given. For reconstruction application a novel corner measure is proposed and successfully applied on several examples. By introducing a probability distance field, functional (4.3) is also able to deal with applications involving heteroscedastic noises, which manifests the generality of the proposed functional. Various examples were presented to demonstrate the flexibility, effectiveness and robustness of the proposed joint functional.

Different from previous two chapters, in this chapter curves are represented by discrete implicit functions or piecewise constant level-set functions. By this representation, the energy functionals can be discretized by graphs, on which fast combinatorial optimization techniques such as min-cut/max-flow algorithms can be applied. Due to the fast optimization techniques, the graph-cuts approach is much faster than the standard level-set implementation, which can be seen from the statistics shown in Table 4.1. Since the min-cut/max-flow algorithms always seek global optimizers in the finite space, the graph-cuts methods are usually more robust than the level-set

methods. In fact, on the same uniform mesh the graph-cut methods are also faster than the phase-field methods proposed in Chapter 3, provided lower neighborhood systems (less than 16 nodes) are used. However, due to the artifacts of discrete functions, graph-cuts methods need finer meshes than level-set methods and phase-field methods need. Besides, in complicated cases, graph-cuts methods need higher neighborhood systems to eliminate metric errors of graph discretization. These phenomenon have been reflected by several examples in the Numerical Tests. These add a lot of computational cost to graph-cuts methods because the number of nodes and edges increases considerably. Based on our experience, to achieve the same visual effects, the overall efficiency of the graph-cuts methods is better than the standard level-set implementation but worse than the phase-field method used in Chapter 3, and even worse than the adaptive level-set method used in Chapter 2. In order to improve the efficiency of the graph-cuts methods, especially for the 3D problems, the number of nodes and edges in the graph has to be reduced. A possible solution is to build the graph on adaptive meshes, and this will be the topic discussed in Chapter 6.

Figure 4.11: Noisy eight-sided star example. (a) The original curve; (b) the noisy curve; (c) the smoothed curve produced with parameters $\lambda = 0.5$ and $\beta = 5$; (d) the smoothed curve produced with parameters $\lambda = 0.03$ and $\beta = 5$; (e) the smoothed curve produced with varying $\lambda(x)$ and $\beta = 5$, where $\lambda(x)$ is computed with $\lambda_1 = 0.5$, $\lambda_2 = 0.5$ in equation (5.24), $w = 4$ in equation (4.5) and $r = 0.1$ for reference circle; (f) the contour of the function $\lambda(x)$ in (d).

Figure 4.12: Segmentation result of an image containing words “NTU”. (a) The segmentation result; (b) the noisy boundaries; (c) the smoothed curve produced with parameters $\lambda = 0.03$ and $\beta = 5$; (d) the smoothed curve produced with varying $\lambda(x)$ and $\beta = 5$, where $\lambda(x)$ is computed with $\lambda_1 = 0.03$, $\lambda_2 = 0.5$ in equation (5.24), $w = 4$ in equation (4.5) and $r = 0.1$ for the reference circle; (e) the contour of the function $\lambda(x)$ in (d).

Figure 4.13: Curve reconstruction from noisy points data. The points in the first figure are the noisy data and the curve shown in the second figure is the reconstruction result.

Figure 4.14: Curve reconstruction from noisy points data. The points in the first figure are the noisy data and the curve shown in the second figure is the reconstruction result.

Figure 4.15: Curve reconstruction from points sampled from a Chinese character. (b) The reconstructed curve with constant regularity parameter; (c) the reconstructed curve with varying regularity parameter; (d) mixture of the curves in (c) and (d); (e) the contour of the corner detector function.

Figure 4.16: Shortest path within a bounded region. Distance function is $\left(\frac{dist}{B}\right)^n$. (a) A bounded region generated by a curve with width 0.05; (b) the solution curve with $\lambda = 0, \beta = 20, n = 3$; (c) the solution curve with $\lambda = 0, \beta = 4, n = 3$; (d) the solution curve with $\lambda = 0, \beta = 4, n = 6$;

Figure 4.17: Shortest polygonal path in a bounded region, obtained by "variational smoothing".

Figure 4.18: An example of L-shape polygon. (a) The original curve; (b) the noisy sample; (c) the smoothed curve maximizing sum probability; (d) the comparison of the curve in (c) and the original curve.

Figure 4.19: A polygon example. (a) The original curve; (b) the noisy samples; (c) the influence field generated by the noisy samples; (d) the smoothed curve; (e) the comparison of the smoothed curve and original curve.

Chapter 5

CVT-based Modelling

5.1 Introduction

Smoothing and reconstruction of curves (in $\mathbb{R}^2$) or surfaces (in $\mathbb{R}^3$) are two basic topics in geometry processing. As discussed in the previous chapter, these problems can be addressed in the same variational framework, in which we always need to minimize certain functional such as

$$\mathcal{E}(\Gamma) = \int_{\Omega} \lambda |u_{\Gamma} - u_{data}|^2 d\mathbf{x} + \oint_{\Gamma} (\alpha + \beta \text{dist}_{\Gamma_{data}}) ds \quad (5.1)$$

where Ω is the computing domain (occupied by the data), Γ is the interface (curve or surface) to be smoothed or reconstructed, $\text{dist}_{\Gamma_{data}}$ is the unsigned distance function of points on Γ to the given data (point sets or line segments or surface patches, etc.), u_{Γ} and u_{data} are the labeling (characteristic) functions induced from Γ and the given data respectively. α , β and λ in (5.1) are tuning parameters to be selected (could be constants or functions). The first term in the right hand side measures the similarity between the curve/surface Γ and the input data, and the second term measures the regularity of Γ .

Some popular methods used to solve this problem include PDE-based methods [30, 92, 99, 120] and recently developed graph-based methods [72, 8, 104]. Differences between these two approaches are mainly from two aspects. The first main

difference is that the PDE-based methods using continuous functions to represent interfaces whereas the graph-based methods using discrete functions, and the other one is that the PDE-based methods basically solve the Euler–Lagrangian equations derived from the proposed energy functional, which usually give local minimizers unless the functional is convex, while the graph-based methods always seek global minimizers in certain finite space as long as the functional is graph-representable. Thus, graph-based methods are often more efficient and robust while PDE-based methods are more general and accurate to some extent. Due to more extensive researches conducted on both sides the gap between them is becoming smaller and smaller.

In the fields of statistics and machine learning, the k -means clustering is a classic method of wide use. Interested reader please refer to [63] and the references cited therein. The basic k -means algorithm can be viewed as a special case of the centroidal Voronoi tessellation (CVT) methodology [36]. As a versatile methodology, the model of CVT [34, 35] has been introduced to numerous fields and applications such as image processing, data analysis and quantization, computational geometry, sensor network, numerical partial differential equations [34, 36], and so on. Recently, in [110] the authors first developed an edge-weighted centroidal Voronoi tessellation (EWCVT) model for image segmentation applications (discrete structured points in $\mathbb{R}^2$), in which the image intensity information is appropriately combined together with the length of cluster boundaries. The method is guaranteed to converge to local minimizers, to this aspect which is analogous to PDE-based methods. EWCVTs for continuous domains in $\mathbb{R}^2$ are then discussed in [111]. The corresponding EWCVT-based algorithms are essentially clustering algorithms like k -means [63] so they are computationally much less expensive than the traditional PDE-based algorithms when the number of phases (or clusters or level sets) are large. In contrast to the graph-based methods, the EWCVT-based methods converge to local minima but it often requires much less memory, especially for multi-phase cases.

The objective of this chapter is to develop a generalized EWCVT model and corresponding algorithms, and apply them to geometry processing problems such as curve/surface smoothing and reconstruction. The first term in the functional (5.1) can be naturally approximated by a discrete clustering energy, and the key part is

how to interpret the second term in the CVT context. Following the formation of the EWCVT model proposed in [110], our idea is to introduce a predefined neighborhood and a generalized edge energy (weighted curve length or surface area) that can be computed by counting the number of the edge points and multiplying with some associated weight (constant or nonconstant) within the neighborhood. An generalized edge-weighted distance then can be derived for the discretized energy functional, and consequently, generalized edge-weighted Voronoi regions can be defined. With this new edge-weighted distance, we then develop a generalized EWCVT model and corresponding algorithms for its constructions.

The remainder of this paper is organized as follows. In Section 2 the energy functional for curve/surface smoothing and reconstruction is interpreted and discretized in the CVT context, and then in Section 3, we present the generalized EWCVT model and algorithms. In Section 4, some implementation issues are discussed regarding the proposed algorithms. In Section 5, various numerical experiments are provided to demonstrate effectiveness and efficiency of our method for geometry processing. Finally, concluding remarks are given in Section 6.

5.2 A discrete model for geometry processing

As discussed in [113, 122], the basic variational models for geometry processing problems will often have the following form:

$$\Gamma = \operatorname{argmin}(similarity(\Gamma, \text{data}) + regularity(\Gamma)) \quad (5.2)$$

where “data” denotes the given data which could be curves, surfaces or a points set, and Γ denotes the codimension one object, for the 2D space it is a curve and for the 3D space it is a surface. In this paper, we consider the following general functional for modeling (5.2): find Γ that minimizes the energy

$$\mathcal{F}(\Gamma) = \int_{\Omega} \lambda |u_{\Gamma} - u_{data}|^2 d\mathbf{x} + \oint_{\Gamma} r ds. \quad (5.3)$$

The first term in the right hand side of (5.3) measures the similarity of Γ to the data set as usual, and the second term is a measure of the regularity of Γ with the function r being a chosen non-negative continuous function defined on whole domain Ω related to u_Γ . There are many other options for the similarity measure, for example, integrating the difference of the signed distance functions over the domain of interest [122].

The first term in the right hand side of (5.3) can be easily expressed by the CVT language such as

$$\sum_{\mathbf{x} \in D} \lambda(\mathbf{x}) |u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 \quad (5.4)$$

where D is a discrete representation of Ω . We note that D could be structured or non-structured, uniform or nonuniform meshes. This is essentially a weighted *clustering energy*. The crucial step is to seek a discretization of the second term which is suitable for clustering algorithms. For simplicity, in the following we derive everything in the two dimensional space (i.e., $\mathbf{x} = (x, y)$), but formulations and results for three or higher dimensional spaces can be similarly obtained.

Assume that the domain Ω is divided by Γ into several clusters. For the indicator function u_Γ , the points in the same cluster have the same value. For a point $(x, y) \in D$, its neighborhood $\mathcal{N}_\omega(x, y)$ is chosen as a disk centered at (x, y) with radius ω . A point in D is identified as an edge point if there is point within its neighborhood belonging to a different cluster.

Let us define a local characteristic function as

$$\chi_{(x,y)}(x', y') = \begin{cases} 1 & \text{if } u_\Gamma(x', y') \neq u_\Gamma(x, y) \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

where $(x', y') \in \mathcal{N}_\omega(x, y)$. Assume that ω is small enough so that the intersection of Γ and $\mathcal{N}_\omega(x, y)$ can be viewed as a straight line segment (Figure 5.1). Then

$$l(x, y) = \int_{\mathcal{N}_\omega} \chi_{(x,y)}(x', y') \, dx' dy', \quad (5.6)$$

Figure 5.1: Shadow area is the set of local edge points associated with (x, y) .

is clearly area inside the shadowed region in Figure 5.1. We can show that

$$\lim_{\omega \rightarrow 0} \frac{3}{4\omega^3} \int_{\Omega} r(x, y) l(x, y) \, dxdy = \oint_{\Gamma} r \, ds,$$

with the proof given in the following.

For points in a band with width 2ω along Γ , the following identity holds

$$l(x, y) = \omega^2 \theta - \omega^2 \sin \theta \cos \theta, \quad (5.7)$$

where θ is the angle showed in Figure 5.1. Obviously, for points outside the band, we have $l(x, y) = 0$. Hence, instead of computing the integral on the whole domain, the

computation is only carried out within the band. Let $h = \omega \cos \theta$, then we have

$$\begin{aligned}
 \lim_{\omega \rightarrow 0} \frac{3}{4\omega^3} \int_{\Omega} r l \, dx dy &= \lim_{\omega \rightarrow 0} \frac{3}{2\omega^3} \int_0^{\omega} \oint_{\Gamma} r(s, h) (\omega^2 \theta - \omega^2 \sin \theta \cos \theta) \, ds dh \\
 &= \lim_{\omega \rightarrow 0} \frac{3}{2\omega} \int_0^{\pi/2} \oint_{\Gamma} r(s, \omega \cos \theta) (\theta - \sin \theta \cos \theta) \omega \sin \theta \, ds d\theta \\
 &= \lim_{\omega \rightarrow 0} \frac{3}{2} \int_0^{\pi/2} \oint_{\Gamma} r(s, \omega \cos \theta) (\theta - \sin \theta \cos \theta) \sin \theta \, ds d\theta \\
 &= \frac{3}{2} \int_0^{\pi/2} \oint_{\Gamma} \lim_{\omega \rightarrow 0} r(s, \omega \cos \theta) (\theta - \sin \theta \cos \theta) \sin \theta \, ds d\theta \\
 &= \frac{3}{2} \int_0^{\pi/2} \oint_{\Gamma} r(s, 0) (\theta - \sin \theta \cos \theta) \sin \theta \, ds d\theta \\
 &= \frac{3}{2} \int_0^{\pi/2} (\theta - \sin \theta \cos \theta) \sin \theta \, d\theta \oint_{\Gamma} r(s, 0) \, ds \\
 &= \oint_{\Gamma} r \, ds. \tag{5.8}
 \end{aligned}$$

Here $r(s, h)$ is assumed to be continuous along the normal direction in order to let changing variables of the integration and taking limits make sense, and this assumption is valid for most applications. The above proof is only for the case of using the circle as neighborhood, and for other type of neighborhoods such as rectangle, the proof is similar although the constant will be different instead of $\frac{3}{4\omega^3}$.

Thus, disregarding the scaling effect, we get a discrete version of the energy functional $\mathcal{F}$ as follows in the two dimensional space:

$$\hat{\mathcal{F}}(u) = \sum_{(x,y) \in D} \lambda(x,y) |u(x,y) - u_{data}(x,y)|^2 + \frac{1}{\omega^3} \left[\sum_{(x,y) \in D} r(x,y) l(x,y) \right]. \tag{5.9}$$

The second term in the right hand of (5.9) is called the *generalized edge energy*, thus we name the functional (5.9) an *generalized edge-weighted clustering energy*. We also note that it is not hard to show for a general d dimensional space, one can obtain the following discrete energy:

$$\hat{\mathcal{F}}(u) = \sum_{\mathbf{x} \in D} \lambda(\mathbf{x}) |u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 + \frac{1}{\omega^{d+1}} \left[\sum_{\mathbf{x} \in D} r(\mathbf{x}) l(\mathbf{x}) \right]. \tag{5.10}$$

This is a generalization of the discrete energy functional developed in [110]. Based on this discrete energy functional, the CVT-based minimization algorithm then can be derived as shown in the next section.

5.3 Generalized edge-weighted centroidal Voronoi tessellations

The aim of this section is to build the generalized EWCVT model for the discrete functional (5.10) and develop fast methods for its minimization based on the CVT methodology studied in [34, 110]. By computing variation of the functional, the corresponding Euler-Lagrangian equation of (5.10) can be obtained and then the minimization can be achieved by some classical descent methods. However, these approaches, like PDE-based methods, usually lack efficiency. As a matter of fact, the minimizer of the functional (5.10) is merely a labeling function and can be obtained by classical clustering algorithms which are essentially integer programming problems.

5.3.1 Generalized edge-weighted Voronoi tessellation and its construction

The problem is to determine variation of the generalized edge-weighted clustering energy (5.10) when the labeling function $u(\mathbf{x})$ of a grid point $\mathbf{x}$ is changed from its current label c_{t_1} to another label c_{t_2} . First, the energy (5.10) can be rewritten as

$$\begin{aligned} \hat{\mathcal{F}}(u) = & \left(\sum_{\mathbf{x}' \in D, \mathbf{x}' \neq \mathbf{x}} \lambda(\mathbf{x}') |u(\mathbf{x}') - u_{data}(\mathbf{x}')|^2 \right) + \lambda(\mathbf{x}) |u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 \\ & + \frac{1}{\omega^{d+1}} \left[\left(\sum_{\mathbf{x}' \in D, \mathbf{x}' \neq \mathbf{x}} r(\mathbf{x}') l(\mathbf{x}') \right) + r(\mathbf{x}) l(\mathbf{x}) \right]. \end{aligned} \quad (5.11)$$

The change of the first term is of clearly zero. The change of the second term is given by

$$\lambda(\mathbf{x}) |c_{t_1} - u_{data}(\mathbf{x})|^2 - \lambda(\mathbf{x}) |c_{t_2} - u_{data}(\mathbf{x})|^2 \quad (5.12)$$

The change of edge energy for $\mathbf{x}$ caused by such change of the label is

$$r(\mathbf{x})[n_{t_2}(\mathbf{x}) - n_{t_1}(\mathbf{x})] \quad (5.13)$$

where n_t is the size of the set

$$A_{\mathbf{x}}^t = \{\mathbf{x}' \mid \mathbf{x}' \in \mathcal{N}_{\omega}(\mathbf{x}), \text{ and } u(\mathbf{x}') = c_t, \text{ and } \mathbf{x}' \neq \mathbf{x}\}.$$

The change of edge energy (without the weight $\frac{1}{\omega^{d+1}}$) for $\mathbf{x}'$ is

$$\sum_{\mathbf{x}' \in A_{\mathbf{x}}^{t_2}} r(\mathbf{x}') - \sum_{\mathbf{x}' \in A_{\mathbf{x}}^{t_1}} r(\mathbf{x}'). \quad (5.14)$$

We note that when $r(\mathbf{x})$ is constant this naturally reduces to change of the edge-weighted energy proposed in [110]. Summing (5.12), (5.13), (5.14), the variation of the total energy $\hat{\mathcal{F}}(u)$ due to the transferring of the point $\mathbf{x}$ from the label c_{t_1} to c_{t_2} is given by

$$\begin{aligned} \lambda(\mathbf{x}) & \left(|u_{data}(\mathbf{x}) - c_{t_1}|^2 - |u_{data}(\mathbf{x}) - c_{t_2}|^2 \right) \\ & + \frac{1}{\omega^{d+1}} \left(\sum_{\mathbf{x}' \in A_{\mathbf{x}}^{t_2}} [r(\mathbf{x}') + r(\mathbf{x})] - \sum_{\mathbf{x}' \in A_{\mathbf{x}}^{t_1}} [r(\mathbf{x}') + r(\mathbf{x})] \right). \end{aligned}$$

Thus, the *generalized edge-weighted distance* from a point $(\mathbf{x})$ to a label c_s can be defined as

$$\text{dist}^*(\mathbf{x}, c_s) = \sqrt{\lambda(\mathbf{x})|u_{data}(\mathbf{x}) - c_s|^2 + \frac{1}{\omega^{d+1}} \sum_{\mathbf{x}' \in \mathcal{N}_{\omega}(\mathbf{x}) \setminus A_{\mathbf{x}}^s} [r(\mathbf{x}') + r(\mathbf{x})]}. \quad (5.15)$$

Note that r is non-negative ensures that the above distance function is always meaningful. Given a set of labels $\mathcal{C} = \{c_s\}_{s=1}^L$, corresponding Voronoi regions associated with the generalized edge-weighted distance dist^* defined in (5.15), $\mathcal{D} = \{D_s\}_{s=1}^L$ of D can be defined as

$$D_s = \{\mathbf{x} \in D \mid \text{dist}^*(\mathbf{x}, c_s) < \text{dist}^*(\mathbf{x}, c_t), t = 1, \dots, L, t \neq s\}. \quad (5.16)$$

We call such set of $\{D_s\}_{s=1}^L$ as an *generalized edge-weighted Voronoi tessellation* (GEWVT) of D associated with the generators $C = \{c_s\}_{s=1}^L$ and the distance function $dist^*$. An iterative algorithm for its calculating is given below:

Algorithm 1 (GEWVT). Given a set of generators $\{c_s\}_{s=1}^L$ and an arbitrary partition $\{\tilde{D}\}_{s=1}^L$ of the domain D :

1. For every $(x, y) \in D$,
 - (a) calculate and compare the generalized edge-weighted distance defined in (5.15) from the point (x, y) to all the generators $\{c_s\}_{s=1}^L$;
 - (b) move the point (x, y) to the cluster whose generator has the smallest edge-weighted distance to the (x, y) , and update $\{\tilde{D}\}_{s=1}^L$.
2. If no point is moved, return $\{\tilde{D}\}_{s=1}^L$ and exit; otherwise go to Step 1.

Since Algorithm 1–GEWVT strictly decreases the energy $\hat{\mathcal{F}}(u)$ along the iterations and the (discrete) input data are finite, this Algorithm will terminate in finite steps.

A way to accelerate Algorithm 1–GEWVT is to only consider the points near the edge points and in each iteration the computation will be carried out merely in a narrow-band around the edge points. The overhead is to update the narrow band within each iteration, which is negligible comparing with the number of points in the whole domain. Such a strategy dramatically improves the efficiency of the Algorithm GEWVT, and leads to the following Narrow-Banded GEWVT algorithm.

Algorithm 2 (Narrow-Banded GEWVT). Given a set of generators $\{c_s\}_{s=1}^L$ and an arbitrary partition $\{\tilde{D}_{s=1}^L\}$ of the domain D :

1. For every $(x, y) \in D$, if there exists some $(x', y') \in \mathcal{N}_{\omega_1}(x, y)$ such that $(x, y) \in \tilde{D}_{s_1}$ and $(x', y') \in \tilde{D}_{s_2}$ ($s_1 \neq s_2$) then mark the point (x, y) as **Band**; otherwise, mark (x, y) as **Fixed**. Repeat this process until each point in D is either marked as **Band** or **Fixed**.
2. For every (x, y) marked as **Band**,
 - (a) calculate and compare the generalized edge-weighted distance defined in (5.15) from the point (x, y) to all the generators $\{c_s\}_{s=1}^L$;

- (b) move the point (x, y) to the cluster whose generator has the smallest edge-weighted distance to the (x, y) , and update $\{\tilde{D}\}_{s=1}^L$.
- 3. If no point is moved, return $\{\tilde{D}\}_{s=1}^L$ and exit; otherwise go to Step 1.

Algorithm 2–Narrow-Banded GEWVT is a far more efficient implementation than Algorithm 1–GEWVT.

5.3.2 Generalized edge-weighted centroidal Voronoi tessellation and its construction

Any feasible solution $u(\mathbf{x})$ of the minimization problem of $\hat{\mathcal{F}}(u)$ defined in (5.10) can be uniquely represented by a set of generators $\{c_s\}_{s=1}^L$ and their corresponding GEWVTs $\{D_s\}_{s=1}^L$ such that

$$u(\mathbf{x}) = c_s \quad \text{for some } s \text{ such that } \mathbf{x} \in D_s. \quad (5.17)$$

Then, to seek a minimizer of $\hat{\mathcal{F}}(u)$ is equivalent to find the optimal generators and the associated GEWVT.

When the $\{D_s\}_{s=1}^L$ is fixed, the generalized edge energy at each point is fixed, and therefore, the problem of finding the optimal generators becomes the classical problem of determining the centroids of the Voronoi regions, which can be computed by

$$\tilde{c}_s = \frac{\sum_{\mathbf{x} \in D_s} \lambda(\mathbf{x}) u(\mathbf{x})}{\sum_{\mathbf{x} \in D_s} \lambda(\mathbf{x})}, \quad i = 1, \dots, L. \quad (5.18)$$

Following the CVT methodology [34], the process of seeking minimizers of (5.10) is to find a set of generators $\mathcal{C} = \{c_s\}_{s=1}^L$ and the associated GEWVT $\mathcal{D} = \{D_s\}_{s=1}^L$ such that

$$c_s = \tilde{c}_s, \quad i = 1, \dots, L. \quad (5.19)$$

In this case, we call the $\{\mathcal{C}, \mathcal{D}\}$ a *generalized edge-weighted centroidal Voronoi tessellation* (GEWCVT) of D , similar to the EWCVT developed in [110].

Analogous to classical CVT approach, the minimization is achieved by *Lloyd iteration* [78, 33] – iteratively updating generators $\{c_s\}_{s=1}^L$ and generalized edge-weighted Voronoi tessellations $\{D_s\}_{s=1}^L$ until they converge, which leads to the algorithm below.

Algorithm 3 (GEWCVT). Given an integer L and an arbitrary partition $\{\tilde{D}_s\}_{s=1}^L$ of the domain D :

1. For every $\tilde{D}_s$, $s = 1, \dots, L$, determine its centroid $\tilde{c}_s$ by (5.18).
2. Replace the generators $\{c_s\}_{s=1}^L$ by $\{\tilde{c}_s\}_{s=1}^L$, and construct the associated GEWVT $\{\tilde{D}_s^{new}\}_{s=1}^L$ by Algorithm 2.
3. If the difference between $\{\tilde{D}_s^{new}\}_{s=1}^L$ and $\{\tilde{D}_s\}_{s=1}^L$ is small, return $\{\tilde{D}_s^{new}\}_{s=1}^L$ and exit; otherwise set $\tilde{D}_s = \tilde{D}_s^{new}$ for $s = 1, \dots, L$ and go to Step 1.

To compare the difference between two tessellations, the number of the point $(x, y) \in \Omega$ such that

$$(x, y) \in \tilde{D}_s \quad \text{and} \quad (x, y) \notin \tilde{D}_s^{new}$$

will be counted, which is denoted by N_{change} . Therefore, when the N_{change} is small enough the iterations will terminate. In practice, one also can use energy change ratio as the stopping criteria as discussed in the Section 5.5.

In Algorithm 3–GEWCVT, although the point transferring between clusters occurs immediately after the possible energy reduce is found, the update of the generators is delayed until the start of the next iteration. Actually, the generators can be updated just after point transferring happens and this has been used to improve efficiency of the GEWCVT-based algorithms in many applications [36, 110].

5.4 Applications to geometry processing

In the last two sections we presented a GEWCVT-based approach for discretizing and minimizing the energy functional of type (5.3). We now will apply the above proposed techniques to two specific applications in geometry processing, i.e., *smoothing*

and *reconstruction*, and also discuss some details on implementation and parameter setting.

5.4.1 Data representation

These two problems are closely related and can be addressed in the following discrete frame work:

$$\begin{aligned} \hat{\mathcal{F}}(u) = \sum_{\mathbf{x} \in D} \lambda_{data}(\mathbf{x}) |u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 \\ + \frac{1}{\omega^{d+1}} \left[\sum_{\mathbf{x} \in D} (\alpha(\mathbf{x}) + \beta(\mathbf{x})) \text{dist}_{\Gamma_{data}(\mathbf{x})} l(\mathbf{x}) \right] \end{aligned} \quad (5.20)$$

where Γ is the curve (or surface) we seek and Γ_{data} the given data that can be either a set of line segments (or triangular faces) describing the given curve (surface) or just scattered points whose exact form depends on applications. The unsigned distance function from a point $\mathbf{x}$ to the data set Γ_{data} is defined as usual, i.e.,

$$\text{dist}_{\Gamma_{data}}(\mathbf{x}) = \inf_{\mathbf{y} \in \Gamma_{data}} \text{dist}(\mathbf{x}, \mathbf{y}),$$

u_Γ is the piecewise constant level set function induced by Γ , and u_{data} is the one induced by Γ_{data} . λ_{data} , α_{data} , and β are some weight functions to be selected, especially λ_{data} is *input-data-dependent*. In fact, if we treat u_Γ as a labeling function and let $r(x) = \alpha_{data} + \beta \text{dist}_{\Gamma_{data}}$, then the above functional has exactly the same form as the functional (5.3). Thus, the curve/surface smoothing and reconstruction problems are optimization problems in this sense.

The curve/surface Γ is implicitly represented by u_Γ , i.e., Γ is the zero-level set of u_Γ . In our approach, the discrete domain D will be divided into several clusters by Γ and each cluster share a unique value of u_Γ , which is called “label” of that cluster. Therefore it is also obviously different with the traditional level set method [86] or phase-field method [38, 40, 122]. When the data are given by line segments (or triangular surface mesh), the labeling function u_{data} can be generated by counting ray crossings. The basic idea of ray crossings is that if a point $\mathbf{p}$ is in the polygon

(polyhedron) a ray from $\mathbf{p}$ will intersects polygon (or polyhedron) boundary odd times, otherwise even times. Let us take a simple 2D problem as an example. Suppose the closed curve is discretized as a closed polygon, which separates the domain of interest into two disconnected subdomains, i.e. interior and exterior ones. Interior region composes of the points with odd crossings and exterior region the points with even ray crossings. It is natural to give interior subdomain a label and exterior subdomain another label. When applied to multi-subdomain problems, the principles are essential similar.

When the data are not given in the form of line or surface segments, for example, in most surface reconstruction problems, it is impossible to completely mark the interior and exterior regions. Even for surface smoothing problems, sometimes it is also unnecessary to completely mark the interior and exterior regions. To resolve these issues, the following method is used. For easy presentation our discussions will be based on two-cluster problem. First, assume the unsigned distance function is available (whose computations will be discussed later), then a crust region is defined as

$$Crust = \{\mathbf{x} : dist_{\Gamma_{data}}(\mathbf{x}) \leq d_{th}\} \quad (5.21)$$

where the thickness of the crust d_{th} may depend on the noisy level and data resolution. If the data are dense and within a reasonably noise level, the crust will separate the domain into at least two subdomains. After the crust is established, the whole domain has a partitioning: inside, outside and crust regions. Then we can give labels for each regions, i.e., the value of u_{data} . We also need to determine the topological information of the points in the crust region by ray-crossing counting in curve/surface smoothing.

The second issue is about computation of the unsigned distance function, which is needed in the functional (5.20) and the crust construction. If the data are given by a structured point set then the simplest way to do this is for each grid point to compute the distance to all segments and then take the smallest as the final distance; otherwise, we use fast sweeping method [119] to compute the distance function using a background grid. This method has complexity $O(N)$ for uniform grids, where N is the grids number.

The last issue is to selection of parameter functions λ_{data} , α , and β . Generally, they can be constants for curve/surface smoothing and reconstruction applications. There are also certain differences between these two applications, especially λ_{data} as discussed below.

5.4.2 Determination of λ_{data}

Smoothing process

When the functional is modeling the smoothing process, β can be zero since the u_{data} is determined at each point and therefore the distance between the solution curve (or surface) and the given noisy data are completely controlled by the topological information, i.e. the labeling function. Hence the parameter λ_{data} plays a role controlling the distance between the solution curve (or surface) and the given one, which is usually a constant depends on noise level.

Recently, a better algorithm for determining λ_{data} (for two-phase problems) was proposed in [122] that could help preserve curve/surface features during smoothing process. The main observation is that in order to keep the special features of the given data the smoothing effect should be small in the feature regions or equivalently the closeness parameter $\lambda_{data}(x)$ should be large. Therefore a data-dependent (non-constant) parameter function $\lambda_{data}(x)$ can be used to improve smoothing results. So far as the features such as corners and edges are concerned, a quantity that can indicates singularities of the noisy data is needed. To this purpose, we generalize the idea developed in [122] to more general multi-phase problems. Let $\mathbf{x} \in D$ be a point on the curve (or surface). A reference ball $S(\mathbf{x})$ centered at the point $\mathbf{x}$ with radius r is introduced for the “*averaging*” — for removing effects caused by noises and detecting singularities. If $\mathbf{x}$ is not close to the junctions, the reference circle $S(\mathbf{x})$ is divided into two parts A and B by the boundary Γ . If the portion of the boundary, $\Gamma \cap S(x)$, is smooth (thus locally straight) enough, the volumes of A and B should be roughly the same as the ball is small enough. While if the point $\mathbf{x}$ is a singularity point, i.e. a corner or edge point, the area of two parts will be quite different even when the ball $S(\mathbf{x})$ is small enough. Denote the areas of A , B and $S(\mathbf{x})$ by $|A|$, $|B|$ and $|S(x)|$

respectively. The ratio defined by

$$\tau_r(\mathbf{x}) = \frac{\min(|A|, |B|)}{|S(\mathbf{x})|} \quad (5.22)$$

can be viewed as the singularity indicator. It is obvious that $0 \leq \tau_r(x) \leq \frac{1}{2}$, and $\tau_r(x)$ attains the maximum $\frac{1}{2}$ only when $\Gamma \cap S(x)$ is straight and passes $\mathbf{x}$. The smaller $\tau_r(x)$ is, the more singular $\mathbf{x}$ is. Moreover, the above observations also basically hold for the case of a noisy curve (or surface) if the radius of $S(\mathbf{x})$ is large enough to cover the amplitude of noises. If $\mathbf{x}$ is a junction point of multiple phases (i.e., a singular point), the above observation is still true except that the ball $S(x)$ is divided into k parts $A_1, A_2, \dots, A_k$. The definition of $\tau_r(\mathbf{x})$ becomes

$$\tau_r(\mathbf{x}) = \frac{\min_i(|A_i|)}{|S(\mathbf{x})|}. \quad (5.23)$$

The following algorithm is used in our implementation for determining $\tau_r(\mathbf{x})$.

Algorithm 4 (Determination of $\tau_r(\mathbf{x})$): Suppose the phase number is k . For every node $\mathbf{x}$ in the computational domain D :

1. Choose a reference ball $\mathcal{N}(\mathbf{x}, r)$ centered at $\mathbf{x}$ with a radius r .
2. Count the number of nodes, $N = \sum_i N_i$, contained inside the reference ball $\mathcal{N}(x, r)$ where N_i denotes the number of nodes with label i in $\mathcal{N}(x, r)$. Set $N_{min} = \min\{N_1, N_2, \dots, N_k\}$.
3. If $N_{min} = N_k$ and $u_{data}(\mathbf{x}) = k$ then set $\tau_r(\mathbf{x}) = N_{min}/N$, otherwise, $\tau_r(\mathbf{x}) = 1/2$.

Note that the last step guarantees that only the points near the curve (or surface) are considered to be possible singular points. Once the “singularity” measure τ_r is available λ_{data} in the functional can be set to be large in regions near singular points and small otherwise. In other words, we specify a non-homogeneous “attraction potential” making corners or edges more attractive. For example, we set

$$\lambda_{data}(\mathbf{x}) = \lambda_1 + \lambda_2(1 - 2\tau_r(\mathbf{x})) \quad (5.24)$$

where λ_1 and λ_2 are some positive parameters. Taking this form, $\lambda_{data}(\mathbf{x})$ will be large for corner or edge regions and attain its minimum in flat regions. Both constant and nonconstant λ_{data} will be tested in our numerical experiments.

Reconstruction process

The parameter λ_{data} plays a even more significant role in reconstruction applications. In this case, the topological information is not readily usable, (actually it is the information being reconstructed), and only geometric information is completely applicable. The following weighted minimal surface or geodesic active contour model [21, 120] is commonly used for reconstruction, in which the distance function acts as a potential field attracting the initial surface to the local minimizers of

$$E_{GAC}(\Gamma) = \int_{\Gamma} \text{dist}_{\Gamma_{data}} ds. \quad (5.25)$$

It is well-known that the global minimizers of above functional are isolated points and its meaningful solutions correspond local minimizers. This fact also implies that the final solution is sensitive to initial guesses. Therefore, in order to minimize this functional using global optimization techniques, such as graph-cuts, narrow-banded implementation has to be used [60]. As pointed out in [104], this is equivalent to minimize the functional (5.3) with a non-constant function λ_{data} such as

$$\lambda_{data}(\mathbf{x}) = \begin{cases} C & \mathbf{x} \notin Crust \\ 0 & \mathbf{x} \in Crust \end{cases} \quad (5.26)$$

where $Crust$ is defined in (5.21), and C is a large constant. Here the width of the crust region is the same as the one in computation of the labeling function. Such choice of λ_{data} guarantees that the labeling function of the solution surface in the region away from the data coincides with u_{data} , while in the crust region discontinuity of the labeling function is forced to align with the data set. It has been numerically verified in [104] that by choosing appropriate band width d_{th} , the computed minimizers of the functional (5.20) always give good results.

5.5 Examples

In this section, various examples are presented to illustrate the effectiveness, efficiency and robustness of the proposed GEWCVT-based method. All experiments were conducted on a PC with Intel Pentium 4 CPU of 3.2GHz and 4GB memory. The first subsection presents results of curve/surface smoothing in the discrete setting, and the second subsection gives the examples of curve/surface reconstruction. Since uniform meshes of the domain are used, outputs of our GEWCVT-based algorithm are merely partition of the mesh nodes. We also note that discretization artifacts are inevitable especially for three-dimensional problems, thus surface examples presented here are all post-processed by a Laplacian smoothing filter to effectively eliminate the discretization artifacts. “Marching Cube” [80] is used to extract surfaces from the partitioned voxel data.

The examples are presented in two categories: smoothing and reconstruction. In our experiments, we take $\mathcal{N}_\omega$ to be a circle centered at point (x, y) with radius ω . As to the stop criteria the following two conditions are used: the first one is

$$\frac{P_{tran}}{P_{tot}} \leq c_1 \quad (5.27)$$

where P_{tran} denotes the number of transferred points and P_{tot} denotes the total number of points in the domain of interest; the second one is

$$\frac{|E^{n+1} - E^n|}{E^{n+1}} \leq c_2 \quad (5.28)$$

where E^{n+1} denotes the energy in the current step and E^n denotes the energy in the previous step. In our experiments, we set $c_1 = 1.0\text{e-}4$ and $c_2 = 2.0\text{e-}5$. Once either of the above conditions is satisfied, our GEWCVT-based algorithm will be terminated (In other words, if the energy change is quite small or very few points are transferred, the iteration will end). Particularly, the neighborhood $\mathcal{N}_{\omega_1}(\mathbf{x}, \mathbf{y})$ defined in Algorithm 3 is taken to be a square centered at point (x, y) with side length ω_1 . Based on our experiments, ω_1 can be very small for example just the mesh size. This is partially because in the application of geometry processing, the initial guess can

Figure 5.2: Noisy Air Jordan logo. (a) Noisy curve; (b) smoothed curve with $\beta = 0$, $\alpha_{data} = 0.02$, and 0.93 seconds are used to obtain this result; (c) smoothed curve with $\beta = 0$, $\alpha_{data} = 0.04$, and 1.42 seconds are used to obtain this result.

be easily obtained, for example, in the smoothing process, one may take it to just be the noisy interface. For all numerical examples, we let $h = \max(h_x, h_y)$ for 2D cases where h_x, h_y denotes the mesh size in the x and y directions respectively, and $h = \max(h_x, h_y, h_z)$ for 3D cases.

5.5.1 Smoothing Examples

The first smoothing examples is a two-phase problem, see Figure Figure 5.2 which shows smoothing of a noisy Air Jordan logo. The noisy curves are obtained from segmentation. Due to low resolution, boundaries of the segmentation results have a lot of zig-zags and some outliers, which can be observed from Figure 5.2(a). For this examples, we use a uniform 1000×1000 rectangular mesh and set the neighborhood size $\omega = 4h$ where $h = 1/500$ is the mesh size. We also take a constant $\lambda_{data} = 1.0$ for the example. We would like to show the effect of the parameters α and β through this example. For the Air Jordan logo example, the we fix the parameter $\beta = 0$. The smoothing results with $\alpha = 0.02$ and $\alpha = 0.04$ are given in Figures 5.2(b) and 5.2(c) respectively. It can be easily observed from this example that as α increases, the result becomes smoother. This is because α controls regularization effect and larger α means curves with shorter length are expected.

Figure 5.3 illustrates a multi-phase example. In this example, the target curves separate the plane into 4 disconnected domains, see Figure 5.3(a) which presents the

Figure 5.3: Noisy number eight. (a) Noisy curves; (b) smoothed curve with $\beta_{data} = 0, \alpha_{data} = 0.06$ and $\omega = 4h$, and 1.2 seconds are used; (c) $\beta_{data} = 0, \alpha_{data} = 0.06$ and $\omega = 5h$, and 2.5 seconds are used.

noisy curve of a number eight. A uniform 1000×1000 mesh is again used. In order to show the effect of the neighborhood size ω , we fix $\lambda_{data} = 1.0$, $\alpha = 0.06$ and $\beta = 0$ and vary the value of ω in this example. Figures 5.3(b) and 5.3(c) show the smoothing results with $\omega = 4h$ and $\omega = 5h$, respectively. It is easy to see from the results that with larger neighborhood size the proposed algorithm produces smoother results; on the other hand, using larger neighborhood sizes requires more computational cost. It is worth noting that in smoothing applications, β can take zero since the similarity can be measured by integration of the difference of the labeling function. However, this is not true in the case of reconstruction examples, where the similarity is measured by integration of the distance field induced by the given data. In the next subsection, this will be discussed thoroughly.

Figure 5.4 illustrates another multi-phase example of curves formed by five noisy circles, in which the phase number is 10. Moreover, in this example curves have intersections, which makes smoothing much more difficult. A 1000×1000 grids is used. The initial noisy curves (see Figure 5.4(a)) are first processed by our GEWCVT-based algorithm with $\lambda_{data} = 1.0$, $\alpha = 0.05$, $\beta = 1.0$ and $\omega = 4h$. It takes 1.82 seconds to obtain the result as shown in Figure 5.4(b), from which it can be seen that noises are removed and resulting curve are smooth indeed. However, the intersections of the curves are shifted a lot, and the resulting curves have quite different topology than the original ones. In order to preserve these intersections, we then change to a

Figure 5.4: Five noisy circles. nb= 4(a) noisy curves; (b) is the result with $\beta_{data} = 0$, $\alpha_{data} = 0.05$ and 1.82 seconds is used. (c) total 15.3 seconds, 0.83 seconds. (d) one more iteration $\alpha_{data} = 0.01$.

data-dependent similarity parameter function α . First, Algorithm 4 is used to detect corners. To this purpose, the reference ball in Algorithm 4 is chosen as a ball with radius $r = 8\omega$, and $\lambda_{data}(\mathbf{x})$ is computed by formula (5.24) with $\lambda_1 = 1.0$, $\lambda_2 = 1000.0$. Figure 5.4(c) presents the smoothing result with this similarity parameter function, from which it can be observed that all intersections are well preserved. It takes totally 15.3 seconds including 14.47 seconds used to compute $\lambda_{data}(\mathbf{x})$. Along with intersections, the noises near the intersected region are kept as well, in some situation this phenomenon is undesirable. To tackle this problem, the resulting curve in Figure 5.4(c) is again smoothed with constant similarity $\lambda_{data} = 1.0$ one more time. Note that the regularity parameter α now needs to be reduced since the input curve (i.e., Figure 5.4(c)) has much less noises. Figure 5.4(d) presented the final smoothing result by our algorithm with $\alpha = 0.01$, where the noises in the intersected regions are also eliminated and the intersection are satisfactorily preserved.

5.5.2 Reconstruction Examples

As described in Section 5.4, in reconstruction applications a *Crust* region with a proper width d_{th} must be first established, which separates the whole domain into

Figure 5.5: A Chinese character. (a) gives points set; (b) is the result with $\alpha_{data} = 0$ and 2.08 seconds is used. (b) is the result with $\alpha_{data} = 0.02$ and 3.33 seconds is used. From above figures, it can be observed that the larger α_{data} is, the smoother curve is and the longer computation time is needed.

several disconnected subregions, and u_{data} assigns different labels to each of the subregions. For the *Crust* region, u_{data} is always given by the smallest label in our implementation. and the initial guess u_0 is simply equal to u_{data} . For example, in the case of two-phase problems the interior region has label 1 and the exterior region has label 2, while the *Crust* has label 1 as well. It is worth pointing out that the width of the crust is crucial in reconstruction applications. Based on the definition of the *Crust*, the *Crust* may not be watertight for too small d_{th} . On the other hand, too large crust will result in bad initial guesses, and hence longer iteration time.

Figure 5.5 shows reconstruction of a Chinese character “WANG”. The input data take the form of noisy points set as shown in Figure 5.5(a) and separate the plane into two parts. The points are distributed in a 1000×1000 mesh. For this example, since the sampling is not so dense the width of the crust is taken to be $d_{th} = 10h$. We define λ_{data} according to (6.14) with $C_{crust}=1e6$ and $\beta = 60$. The reconstructed curve using our GEWCVT-based method algorithm with $\alpha = 0$ is presented in Figure 5.5(b) and that with $\alpha = 0.02$ in Figure 5.5(c). With extra regularization, the resulting curve in Figure 5.5(c) is clearly smoother than the one in Figure 5.5(b) as expected, but at the same time it also requires longer computation time.

Figure 5.6 shows reconstruction of another Chinese character “LI” that has three disconnected components. This example can be viewed as a four-phase reconstruction

Figure 5.6: A Chinese character having three components. It takes 3.03 seconds on 1000^2 grids to obtain the presented result. In this example $\alpha_{data} = 0.02$ and $\beta = 60$.

problem, which still can be automatically handled by the proposed GEWCVT-based method. The points set is shown in Figure 5.6(a) and the reconstructed curve is presented in Figure 5.6(b). The GEWCVT algorithms took 3.03 seconds on a 1000×1000 grids, while the width of the crust is $d_{th} = 10h$, $\alpha = 0.02$, and $\beta = 60$ this time.

The example given in Figure 5.7 concerns surface reconstruction in the three dimensional space. The initial noisy points set presents the shape of a horse as shown in Figure 5.7(a) which includes 494,195 sampling points (data were obtained from Large Geometric Models Archive of Georgia Institute of Technology http://www.cc.gatech.edu/projects/large_models/). We set $\alpha = 0.0$, $\beta = 60$ and again define λ_{data} according to (6.14) with $C_{crust}=1e6$. The width of the crust is selected as $d_{th} = 2h$. We reconstruct the target surface using our GEWCVT-based algorithm on two different meshes, one is $400 \times 400 \times 400$ and the other is $800 \times 800 \times 800$, see Figures 5.7(b) and 5.7(c) for the results. As mentioned before, a few times of the Laplacian smoothing are performed on the reconstructed surfaces to eliminate the stair-cased artifacts. Compared with Figure 5.7(a), the surface in Figure 5.7(b) has more details in the face and presents clear grains on muscles. Thus, higher resolution mesh results in better quality, but also more computation cost.

The example shown in Figure 5.8 is about reconstruction of a girl model called “Bimba”, whose data set has 74,764 points (Figure 5.8(a)) and is obtained from

Figure 5.7: Reconstruction of a 3D horse on different meshes by our GEWCVT-based algorithm with $\alpha = 0.0$, $\beta = 60$, and $d_{th} = 2h$. (a) The given noisy points set; (b) The reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 52.6 seconds); (c) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time = 397.1 seconds).

Figure 5.8: Reconstruction of a “Bimda” model by our GEWCVT-based algorithm with $\alpha = 0.0$, $\beta = 60$, and $d_{th} = 4h$. (a) The given noisy points set; (b) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time = 464.1 seconds).

Aim@Shape Shape Repository (<http://shapes.aim-at-shape.net/>). All parameters are set to be the same values used by the example in Figure 5.7 except the width of the *Crust* $d_{th} = 4h$. In order to catch the fine features such as the plaits of the girl, the surface reconstruction is carried on $800 \times 800 \times 800$ mesh which consumes 464.09 seconds. From Figure 5.8(b), it can be observed that all the fine features such as plaits and facial expressions are faithfully reconstructed. The following example in Figure 5.9 demonstrates that the proposed GEWCVT-based method is capable of tackling non-uniform sampling. As shown in Figure 5.9(a), the points set (196278 points) of the frog model has a hole at the bottom. To deal with such a non-uniform sampling example using uniform meshes, normally a very coarse mesh has to be used so that the hole is contained in one grid. However, such a strategy clearly cannot produce a satisfactory surface provided that the hole is large and fine features exist. Actually, since the sampling of this example is quite dense except the hole, by specifying a relative thick crust region, the proposed GEWCVT-based method is able to resolve fine features with high resolution as well as overcome the difficulties caused by non-uniform sampling. We use the same parameters λ_{data} , α_{adta} , and β as that in the former example in Figure 5.7. Figure 5.9(b) shows the reconstructed surface on a $200 \times 200 \times 200$ mesh, where the width of the *Crust* region $d_{th} = 10h$ which is large

enough to envelop the hole. It is obvious that the hole is filled as shown in Figure 5.9(b). Due to the existence of the hole, as the mesh size decreases the width of the *Crust* region has to be kept; it means that $d_{th} = 20h$ on a $400 \times 400 \times 400$ mesh and $40h$ on a $800 \times 800 \times 800$ mesh. This leads to considerable long computation time since the solution depends on the choice of the initial *Crust* region. In order to efficiently obtain good initial guesses in presence of the non-uniform sampling, a hierarchy implementation is used. The problem is first solved on a coarse mesh, then the result on the coarse mesh will be mapped on a fine mesh, and the *Crust* region on the fine mesh is built based on the result from the coarse mesh. For simplicity, we always halve the mesh size, and denote $2h$ and h by the mesh size of the coarse mesh and fine mesh respectively. The detailed idea is given as follows. Let $u_{i,j,k}^{2h}$ be the solution on the coarse mesh, and $\tilde{u}_{i,j,k}^h$ be a labeling function on the fine mesh. If $u_{i,j,k}^{2h}$ has a adjacent node which has a different label, $\tilde{u}_{2i,2j,2k}^h$ and $\tilde{u}_{2i\pm1,2j\pm1,2k\pm1}^h$ are marked as *Crust*, otherwise $\tilde{u}_{2i,2j,2k}^h$ and $\tilde{u}_{2i\pm1,2j\pm1,2k\pm1}^h$ equal to $u_{i,j,k}^{2h}$. By the above technique every node in the fine mesh has an initial label by setting $u_{data}^h = \tilde{u}^h$, the problem then can be solved on the fine mesh. By such implementation the computation cost on the fine mesh can be considerably reduced. The reconstructed surface from the frog model using our GEWCVT-based algorithm on finer meshes are presented in Figures 5.9(c) and 5.9(d) respectively.

Figure 5.10 presents the surface reconstruction of a statue called “Fertility”. The points set consisting of 241,607 points and is obtained from Aim@Shape Shape Repository as well. This model has several regions which are under-sampled as shown in Figure 5.10(a), and moreover it has thin and elongated parts which require high resolution to preserve correct topology. Use of an uniform mesh at least 400×400 resolution is needed, otherwise the arms could be disconnected. The result shown in Figure 5.10(b) is reconstructed using a $800 \times 800 \times 800$ mesh by our GEWCVT-based algorithm with same parameters as the example in 5.9(d). It costs totally 1859.9 seconds.

Figure 5.11 illustrates a multi-phase (four phases) example of two intersected spheres. Figure 5.11(a) shows the points set which consists of 3,442 points. The $400 \times$ mesh is used to resolve the intersection parts. The width of the *Crust* is

Figure 5.9: An example has non-uniform sampling. From the second row to the last row, results obtained from 200^3 grids to 800^3 grids are given respectively. The hierarchy implementation is utilized to accelerate the speed on the fine mesh: the computation time on 200^3 grids is 73.5 seconds; By using the initial solution from previous mesh, GEWCVT spends 194.74 seconds on 400^3 grids; the iteration time on 800^3 grids is 916.73 seconds, and without the hierarchy implementation it will take 4231.2 seconds.

Figure 5.10: Reconstruction of the "Fertility" model by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 40h$. (a) The given noisy points set; (b) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time = 1859.9 seconds).

$d_{th} = 8h$, which is relatively large since the sampling points is very coarse. Figure 5.11(b) presents the reconstructed surface by our algorithm viewed from outside, and Figure 5.11(c) gives the decomposition of the surface, from which the topology of the surface can be easily observed.

Figure 5.12 presents another multi-phase (three phases) reconstruction example, in which two cubic frames are entangled with each other. The input data have 15344 points as shown in 5.12(a). Although this example can be modeled as a two-phase problem, by means of the multi-phase implementation the phase numbers are automatically determined and the surface can be extracted from the partitioned voxel data without any artificial intervention. The computation is carried on a $400 \times 400 \times 400$ mesh and cost 313.75s, see Figure 5.12(b) for the final reconstructed surface.

5.6 Concluding Remarks

In this chapter, we propose a new edge-weighted centroidal Voronoi tessellation model, which is an important generalization of the edge-weighted centroidal Voronoi tessellation model developed in [110]. By this new model, the general variational framework proposed in the last chapter can be interpreted in CVT or clustering language. Several solution algorithms are developed and applied on applications of geometry

Figure 5.11: Reconstruction of the surface of two intersected spheres by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 8h$. (a) The given noisy points set; (b) the reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 294.7 seconds); (c) the decomposition of the surface for visualization.

Figure 5.12: Reconstruction of two entangled cubes by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 8h$. (a) The given unorganized noisy points set; (b) the reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 313.7 seconds)

process such as curve/surface smoothing and reconstruction. Moreover, the challenging multi-phase surface reconstruction problems are also considered, and the corner measure proposed in Chapter 3 is extended to multi-phase problems as well. Through various numerical examples, the proposed GEWCVT-based method is shown to be an effective and efficient tool for such applications. It is worth pointing out that as a new clustering method the GEWCVT method can not only be applied to image segmentation, surface reconstruction but also some other problems in computer vision, computer graphics, data analysis and quantization.

Comparing with the graph-cuts method, the discrete GEWCVT-based method also uses discrete functions i.e. piecewise constant level set functions or labeling functions to represent curves/surfaces. However, there are some essential differences between graph-cuts methods and discrete GEWCVT-based methods. First, in the discrete GEWCVT-based methods the energy functional is discretized through a CVT-based way. Therefore, unlike graph-cuts methods requiring memory space to store both nodes' and edges' information, the GEWCVT-based methods only need store the information of nodes. For multi-phase problems, discrete GEWCVT-based methods need much less memory than graph-cuts methods, since graph-cuts methods need one graph for each phase as phase number increases the memory request increases considerably, while the memory request of discrete GEWCVT-based methods for multi-phase problems is as same as two-phase problems. Second, the graph-cuts methods solve the optimization problem by min-cut/max-flow algorithms, while GEWCVT-based methods minimize the corresponding energy functional by computing the variation of the generalized edge-weighted clustering energy and doing the corresponding transferring. To some aspects, GEWCVT-based methods are similar to the traditional PDE-based methods such as level-set methods: (1) GEWCVT-based methods are also guaranteed to converge to local minimum; (2) the efficiency of GEWCVT-based methods highly depend on initial solution as well. Hence, generally GEWCVT-based methods are slower than graph-cuts methods. But discrete GEWCVT-based methods are basically integer programming, their efficiencies will be much better than level-set methods.

Chapter 6

Graph-cuts on Tetrahedron Meshes

6.1 Introduction

Reconstruction of a surface from a given unorganized point set is an important yet challenging problem in computer graphics. As the first step of creating computer graphics, surface reconstruction plays a significant role in converting discrete scanned data to a continuous surface model. In real applications, however, it is inevitable to encounter difficulties such as non-uniform sampling, incomplete scanning, noises as well as complicated geometry and topologies, which often make reconstruction problems ill-posed. To address these issues, extensive researches have been conducted and various methods have been proposed, most of which can be categorized into two groups: explicit methods [46, 5, 6, 7, 2, 11, 31, 22, 115, 10, 83, 27, 106], and implicit methods [59, 29, 4, 82, 120]. In view of these existing methods, some are only dedicated to special applications; some are not efficient enough, and some involve complicated post-processing. Therefore, a robust and efficient reconstruction method, which can incorporate most advantages of explicit and implicit methods and takes all aforementioned difficulties into account, is of much necessity and will be the objective of this paper.

Most explicit methods are based on Voronoi diagrams and dual Delaunay triangulations. One advantage of these Delaunay based approaches is that given a sufficient sampling density, i.e. ϵ -sampling [5], the restricted Delaunay triangulation of data set

points is homeomorphic to the original surface [7], which suggests a sound approximation of the original surface. Another advantage is that the resulting homeomorphic sub-complex, i.e. the restricted Delaunay triangulation, has a structure of triangular surface mesh, which is preferred in many applications due to its simplicity, its efficiency to approximate the original surface, and its convenience as the boundary input for further volumetric mesh generation or other processing [56, 114, 94, 64, 42, 43]. The Alpha Shape [46], the CRUST algorithm [5] and the COCONE algorithm [31] are good examples of explicit Delaunay based methods. However, undersampling, non-uniformity, and noises introduced during the data acquisition stage are likely to destroy the desirable homeomorphic sub-complex, hence make reconstruction problems ill-posed and cause various difficulties. Scanning insufficiency aside, complicated topologies of multiphases are challenges for explicit methods.

Considering difficulties faced by explicit methods, researchers turned to variational implicit methods in the last decade. Variational methods regularize the ill-posedness and eliminate the heuristics which is more or less required in explicit methods. By utilizing implicit surface representation such as level set terminology, some implicit methods [59, 29, 4, 82] were proposed to handle one or several difficulties mentioned above. Since [120], the weighted minimal surface model has proven to be a useful framework in various reconstruction problems. In this chapter, a generalized weighted minimal surface model is proposed, of which both the original model in [120] and the one in [61] turn out to be special cases. More details of this model will be given in Section 3.

Though implicit methods have several advantages over explicit ones, it would be even more ideal that the merits of both kinds of approaches can be exploited. For example, triangular surface meshes preferred by most applications can not be extracted in implicit methods as easily as explicit methods. Besides, the nice “ ϵ -sampling-homeomorphic” theoretic guarantee has seldom been exploited by implicit methods. Therefore, in this chapter the proposed variational model is minimized on a Delaunay-based tetrahedral mesh instead of a regular grid like [120, 61]. Since all data points have been inserted into the background mesh in a Delaunay way, the homeomorphic surface pursued by [46, 5, 6, 7, 2, 11, 31] exists given a sufficient sample

and a proper mesh. Even though the homeomorphic surface may not exist due to the impairment of noises or undersampling, the Delaunay triangulation could still provide a “partial homeomorphic” sub-complex. For the sufficiently sampled part, the corresponding surface patches are still homeomorphic; for the part suffering from noises or undersampling, the variational model can obtain a reasonable result though without the homeomorphic guarantee. By in turn solving the minimization problem the uncertainties caused by noises or undersampling are eliminated and complicated topologies can be handled as well. The merits of explicit and implicit methods are therefore beautifully integrated into one method.

Regarding to the minimization problem, one approach is to solve the Euler-Lagrange equation of the energy functional by gradient descent as in [120]. However, in this paper, the graph cut technique [72] is used to solve the minimization problem. Compared with the iterative gradient descent approach, the graph cut algorithm, with polynomial complexity, provides a much faster solver. Since [58], graph cut has been widely applied in low computer vision and graphics problems. Several graph based works have been accomplished in surface reconstruction [71, 103, 88, 61, 60, 95]. Many efficient min-cut/max-flow algorithms were developed during the study of combinatorial optimization. Especially, the algorithm in [19] has empirically good performance on grids and proves efficient on the graphs dual to Delaunay tetrahedral meshes in this chapter as well.

Furthermore, the proposed reconstruction method is extended from two-phase to multiphase problems, which arise in numerous applications like CAD, medical imaging and animation. The energy functional in this paper can model multiphase problems as well, but the corresponding min-cut/max-flow problems become multi-way cut ones, which have been challenges since the wide application of graph cut [19, 73, 76]. Our method adapts the multilayer idea in [8], which translates the original multi-way cut problem to a binary cut in a new multilayer graph. With region growing algorithms, both multiphase and two-phase cases can be solved in an automatic and unified way. To the best of our knowledge, this is the first time that multiphase surface reconstruction problem is solved by the graph cut technique.

The remainder of this chapter is organized as follows. In Section 6.2 the weighted

minimal surface model in [120] is reviewed and the level set approach is given for comparison. In Section 6.3, a generalized weighted minimal surface model is proposed and the graph cut approach is presented. One simple example illustrates the main stages. In Section 6.4, the proposed method is extended from two-phase to multiphase problems. The method in [8] is adapted to generate a higher dimensional graph, from which a hyper-surface is obtained and then translated to several surfaces in original dimensions. In Section 6.5, some interesting and challenging examples are provided. Section 6 concludes the article.

6.2 Weighted Minimal Surface Model and Level Set Approach

By applying implicit surface representation, the weighted minimal surface model in [120] can handle complicated topologies easily. A brief introduction of the level set approach utilized in [120] will be given in this section.

Let S be a data set, which contains points, curves and patches of surfaces. Define the distance function from a point x to S as $d(x) = d(x, S) = \inf_{y \in S} \text{dist}(x, y)$, where $\text{dist}(x, y)$ is the Euclidean distance between points x and y . The weighted minimal surface model defines a surface energy functional:

$$E(\Gamma) = \left[\int_{\Gamma} d^p(x) ds \right]^{1/p}, \quad 1 \leq p \leq \infty, \quad (6.1)$$

where Γ is an arbitrary surface and ds is the surface area. If $p < \infty$, $E(\Gamma)$ is based on an integral on surface area weighted by a certain power of the distance function. That is the reason why this model is called the weighted minimal surface model. When $p = \infty$, $E(\Gamma)$ becomes the Hausdorff distance between Γ and S :

$$\lim_{p \rightarrow \infty} E(\Gamma) = \sup_{x \in \Gamma} \inf_{y \in S} d(x, y). \quad (6.2)$$

To find a local minimizer of $E(\Gamma)$, the gradient flow of the energy functional $E(\Gamma)$ is

obtained,

$$\frac{d\Gamma}{dt} = - \left[\int_{\Gamma} d^p(\vec{x}) ds \right]^{\frac{1}{p}-1} d^{p-1}(x) \left[\nabla d(\vec{x}) \cdot \vec{N} + \frac{1}{p} d(\vec{x}) \kappa \right] \vec{N}, \quad (6.3)$$

and the corresponding Euler-Lagrange equation is as follows,

$$d^{p-1}(\vec{x}) \left[\nabla d(\vec{x}) \cdot \vec{N} + \frac{1}{p} d(\vec{x}) \kappa \right] = 0, \quad (6.4)$$

where $\vec{N}$ is the unit normal and κ is the mean curvature.

Since the Euler-Lagrange equation (6.4) and the gradient flow (6.3) involves surface related geometric properties, a consistent global parametrization of surface is required in explicit representations, which is very difficult for complex topologies and evolving surfaces. Instead, implicit representations such as level set are applied. The main idea is to represent surface by a certain level set of a scalar function, usually the zero level set.

In an R^d space X , define a level set function $\phi(x)$, by which we can represent a co-dimension one surface as, $\Gamma = \{x : \phi(x) = 0\}$. Geometric properties like area, unit normal $\vec{N}$, and mean curvature κ can be calculated in the level set formulation as follows.

$$Area(\Gamma) = \int_{\Gamma} ds = \int_X |\nabla H(\phi)| dx = \int_X \delta(\phi) |\nabla \phi| dx, \quad (6.5)$$

$$\vec{N} = \frac{\nabla \phi}{|\nabla \phi|}, \quad \kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right). \quad (6.6)$$

Similar to the surface area, the energy functional in (6.1) can be translated as follows,

$$\int_{\Gamma} d^p(x) ds = \int_X d^p(x) \delta(\phi(x)) |\nabla \phi| dx. \quad (6.7)$$

Since $\Gamma(t) = \{x : \phi(x, t) = 0\}$, by differentiating both sides of this equation with respect to t , we can obtain,

$$\frac{d\phi(\Gamma(t), t)}{dt} = \frac{\partial \phi}{\partial t} + \frac{d\Gamma(t)}{dt} \cdot \nabla \phi = 0. \quad (6.8)$$

Substituting $\frac{d\Gamma(t)}{dt}$ in (6.8) by (6.3),

$$\frac{\partial \phi}{\partial t} = -\frac{d\Gamma(t)}{dt} \cdot \nabla \phi = \frac{1}{p} |\nabla \phi| \left[\int d^p(x) \delta(\phi) |\nabla \phi| dx \right]^{\frac{1}{p}-1} \nabla \cdot \left[d^p(x) \frac{\nabla \phi}{|\nabla \phi|} \right]. \quad (6.9)$$

Thus the level set formulation translates the partial differential equation of Γ to that of ϕ , a scalar function defined on the entire domain X . The standard finite difference method can be used to discretize equation (6.9) on regular grids [120]. In this paper, the graph cut technique is instead applied to minimize a modified version of the above energy functional, which will be discussed in the next section.

6.3 Reconstruction Based on Delaunay Triangulation and Graph Cut

During the last decade, many researchers and works paid plenty of attention to the weighted minimal surface model, based on which we propose our variational model. In the proposed model, a regularization term concerning the surface area is added. Moreover, an indicator function as well as a confidence function is introduced to make the model more flexible and efficient. The graph cut technique is used to minimize the energy functional. Equivalence between the minimal cut and the energy minimization can be verified by the conclusion of [72]. In this section, the variational model is presented with a detailed explanation. Then, the process of graph construction is discussed. Finally the whole algorithm is presented with details.

6.3.1 The Variational model

Let S , Ω , Γ and $d(x)$ be the same as in Section 2. The following cost energy is proposed for surface reconstruction,

$$E(\Gamma) = \int_{\Omega} |\phi_{\Gamma}(x) - I(x)| \beta(x) dx + \int_{\Gamma} d(x) ds + \alpha \int_{\Gamma} ds, \quad (6.10)$$

where $\phi_\Gamma(x)$ is the level set function induced by the surface Γ , $I(x)$ the indicator function, $\beta(x)$ the confidence function and α the regularization coefficient. The discussions throughout this section are based on two phase cases though similar arguments are valid for multiphase cases.

Instead of using unsigned distance function in [120], a piecewise constant level set function $\phi_\Gamma(x)$ is used,

$$\phi_\Gamma(x) = \begin{cases} 1 & \text{if } x \text{ inside } \Gamma \\ 2 & \text{if } x \text{ outside } \Gamma \end{cases}, \quad (6.11)$$

which is more suitable for graph-representing, and hence the surface is now represented as discontinuities of the function rather than the zero level set. And the level values such as 1 and 2 can be arbitrarily selected as long as they are consistent with the indicator values.

The crust region around S is defined as

$$Crust = \{x : d(x, S) \leq d_{th}\}, \quad (6.12)$$

where d_{th} defines the thickness of the crust region. In $Crust$, a priori topology information is unreliable due to the influences of noises. The building of the crust takes account of the noise level: the higher the noise level is, the larger d_{th} is taken, which means a wider vicinity of the data set is included in the thicker crust. The watertight crust separates Ω into two disconnected subdomains, the interior and the exterior. The indicator function is used to identify these subdomains.

$$I(x) = \begin{cases} 1 & \text{if } x \text{ in the interior} \\ 2 & \text{if } x \text{ in the exterior} \\ 0 & \text{if } x \text{ is in the } Crust \end{cases}. \quad (6.13)$$

The indicator function serves as an initial estimation of level set function or a priori knowledge of the topology information, which can be identified interactively or accomplished automatically by the region growing algorithm as in this paper.

The confidence function suggests the extent to which this initial estimation is

faithful. The following confidence function is used in this chapter,

$$\beta(x) = \begin{cases} c & x \in Crust^c \\ 0 & x \in Crust \end{cases}, \quad (6.14)$$

where c is a relatively large constant.

Actually, since the confidence decreases down to zero in the crust region and the first term contributes nothing to (6.10), any indicator value is acceptable and zero is taken in this example. It is worth pointing out that the functional used in [60] is similar to (6.10) but without the first term which means it can not be solved by the graph cut method directly since its global minimizers correspond to isolated data set points which are trivial for surface reconstruction. Thus, in their work a narrow-band implementation is applied which is equivalent to taking the $I(x)$ and $\beta(x)$ in (6.10) as (6.13) and (6.14). In other words, the functional (6.10) is more general and flexible, and proper choices of $I(x)$ and $\beta(x)$ not only guarantee a meaningful global minimizer but also improve the efficiency of the max-flow/min-cut algorithm.

6.3.2 Mesh and Discretization

In this chapter, an unstructured tetrahedral mesh is used instead of structured grid to solve the minimization problem. Generally speaking, a three dimensional tetrahedral mesh $\mathcal{T}_h$ consists of tetrahedra $\{K_i\}_{i=1}^N$, the intersection of any two of which is either empty set or a lower dimensional face, i.e. triangle, edge and vertex [55]. A proper sized mesh reasonably discretizes the domain as well as any energy function defined on the domain. In this subsection, a brief introduction of the discrete versions of most definitions in Section 3.1 is given.

The surface reconstruction problem can be thought of as a segmentation problem. Once the domain is segmented into two partitions, the surface can be easily represented by the interface of two partitions. When the domain is discretized as tetrahedra $\{K_i\}_{i=1}^N$ by the mesh $\mathcal{T}_h$, the segmentation problem is equivalent to labelling all tetrahedra: $L(K_i) = l_i, l_i \in \{1, 2\}$. As a consequence, the surface, the interface of two partitions, is approximated by the union of those triangle faces shared by tetrahedra

of different labels: $\Gamma_h = \bigcup_{l_i \neq l_j} (K_i \cap K_j)$. The crust region notation is also modified to the union of the tetrahedra in a vicinity of S . Normally the n -ring neighborhood of S is a good choice for the crust region. All the functions involved in the $E(\Gamma)$ can also be modified for the mesh framework. Generally, the domain of those discrete functions is $\{K_i\}_{i=1}^N$ instead of Ω for the continuous functions. Take the level set function as an example, the continuous function $\phi_\Gamma : \Omega \rightarrow \{1, 2\}$, while the discrete function $\phi_{\Gamma_h} : \{K_i\}_{i=1}^N \rightarrow \{1, 2\}$.

$$\phi_{\Gamma_h}(K_i) = \begin{cases} 1 & \text{if } K_i \text{ inside } \Gamma_h \\ 2 & \text{if } K_i \text{ outside } \Gamma_h \end{cases}, \quad (6.15)$$

which is equivalent to the labelling function $L(K_i) = l_i$. For the remaining of this article, we refer $\phi_{\Gamma_h}(K_i)$ and l_i to the same thing.

Similarly, the crust region separates the tetrahedra $\{K_i\}_{i=1}^N$ into two subdomains, i.e. interior and exterior. And the modified indicator function and confidence function is as follows.

$$I(K_i) = \begin{cases} 1 & \text{if } K_i \text{ in the interior} \\ 2 & \text{if } K_i \text{ in the exterior} \\ 0 & \text{if } K_i \text{ is in the } Crust \end{cases}. \quad (6.16)$$

$$\beta(K_i) = \begin{cases} c & K_i \in Crust^c \\ 0 & K_i \in Crust \end{cases}. \quad (6.17)$$

Hence the energy functional proposed in (6.10) can be approximated as follows.

Figure 6.1: An illustration of the tetrahedral mesh and the crust around the data set.
 (a) Data points and mesh points; (b) A cut view with crust

$$\begin{aligned}
 E(\Gamma) &= \int_{\Omega} |\phi_{\Gamma}(x) - I(x)| \beta(x) dx + \int_{\Gamma} d(x) ds + \alpha \int_{\Gamma} ds \\
 &\approx \sum_{i=1}^N |\phi_{\Gamma_h}(K_i) - I(K_i)| \beta(K_i) + \sum_{i < j, l_i \neq l_j} \int_{K_i \cap K_j} (d(x) + \alpha) ds \\
 &\approx \sum_{i=1}^N |l_i - I(K_i)| \beta(K_i) + \sum_{i < j, l_i \neq l_j} (d_{ij} + \alpha) S_{ij} \\
 &= \sum_{i=1}^N |l_i - I(K_i)| \beta(K_i) + \sum_{i < j} (d_{ij} + \alpha) S_{ij} \mathbf{1}_{\{l_i \neq l_j\}} \\
 &= E_h(\Gamma_h),
 \end{aligned} \tag{6.18}$$

where

$$d_{ij} = \begin{cases} \frac{\int_{K_i \cap K_j} d(x) ds}{\int_{K_i \cap K_j} ds} & \text{if } K_i \cap K_j \neq \emptyset \\ 0 & \text{others} \end{cases}, \quad S_{ij} = \int_{K_i \cap K_j} ds. \tag{6.19}$$

A concrete example is given to illustrate the discretization of a tetrahedral mesh. Red points in Figure 6.1(a) are data points sampled from a sphere surface and blue points are mesh points. A proper tetrahedral mesh is generated on the union of data points and mesh points, a cut view of which is shown in Figure 6.1(b). The red crust

Figure 6.2: Illustration of graph construction

region separates all tetrahedra into two partitions, colored in blue and green.

6.3.3 Graph Construction

To minimize the approximated energy functional $E_h(\Gamma_h)$ in (6.18), a graph G dual to the tetrahedral mesh $\mathcal{T}_h$ is constructed. For each tetrahedron K_i in $\mathcal{T}_h$, a node x_i is added to the dual graph G . For each triangle face shared by two tetrahedra K_i and K_j in $\mathcal{T}_h$, an edge connecting x_i and x_j is added to G . This primal-dual relationship is illustrated by a two dimensional example in Figure 6.2(a), in which the blue solid lines stand for the primal mesh and the red dashed lines the dual graph.

Given a watertight surface triangulation mesh Γ_h in $\mathcal{T}_h$, all tetrahedra can be separated into two partitions, which can be represented by the level set function $\phi_{\Gamma_h}(K_i) = l_i, l_i \in \{1, 2\}$. This piecewise constant level set function is equivalent to a labelling function in the graph G , $L(x_i) = l_i, l_i \in \{1, 2\}$. Furthermore, the surface Γ_h in $\mathcal{T}_h$ is equivalent to a cut C in the dual graph G . The surface Γ_h is the union of all triangles shared by tetrahedra from different partitions, while the cut C is the union of all edges connecting the nodes of different labels. This primal-dual relationship and its derivatives are summarized in Table 6.1.

The first term in (6.18) corresponds to the t -links in the graph and the second term n -links. The sketch in Figure 6.2(b) and the following (6.20) illustrates the edge

Table 6.1: Primal-dual relationship

Primal Mesh $\mathcal{T}_h$	Dual Graph G
Tetrahedron K_i	Node x_i
Triangle $F_{ij} = K_i \cap K_j$	Edge $e_{ij} = (x_i, x_j)$
$\phi_{\Gamma_h}(K_i) = l_i, l_i \in \{1, 2\}$	$L(x_i) = l_i, l_i \in \{1, 2\}$
$\Gamma_h = \bigcup_{l_i \neq l_j} (K_i \cap K_j)$	$C = \bigcup_{l_i \neq l_j} (x_i, x_j)$

weight assigning procedure. The whole algorithm of constructing graph is given in details in the next subsection.

$$\begin{aligned}
 s_i &= |I(K_i) - 2| \beta(K_i), \quad s_j = |I(K_j) - 2| \beta(K_j), \\
 t_i &= |I(K_i) - 1| \beta(K_i), \quad t_j = |I(K_j) - 1| \beta(K_j), \\
 N_{ij} &= (d_{ij} + \alpha) S_{ij}, \quad N_{ji} = (d_{ij} + \alpha) S_{ij}.
 \end{aligned} \tag{6.20}$$

Furthermore, [72] proved that for an energy function having the form of

$$E(x_1, \dots, x_n) = \sum_i E^i(x_i) + \sum_{i < j} E^{i,j}(x_i, x_j), \tag{6.21}$$

the minimization of E can be approached by the graph cut technique, i.e. E is graph-representable, if and only if each $E^{i,j}$ satisfies the inequality

$$E^{i,j}(1, 1) + E^{i,j}(2, 2) \leq E^{i,j}(1, 2) + E^{i,j}(2, 1). \tag{6.22}$$

Comparing $E_h(\Gamma_h)$ to (6.21), it is not difficult to find,

$$E^i(x_i) = |l_i - I(K_i)| \beta(K_i), E(x_i, x_j) = (d_{ij} + \alpha) S_{ij} \mathbf{1}_{\{l_i \neq l_j\}}. \tag{6.23}$$

Hence it can be verified that $E(\Gamma)$ is graph representable by finding that

$$E(l_i = 1, l_j = 1) = E(l_i = 2, l_j = 2) = 0. \tag{6.24}$$

6.3.4 Algorithm and Discussions

The proposed algorithm consists of three stages. The first stage is the primal mesh generation based on the Delaunay triangulation. The second stage includes establishing the crust region and specifying functions in the variational model, i.e. the distance function, the indicator function and the confidence function. The third stage includes constructing the dual graph, finding the minimal cut and extracting the desired surface from the minimal cut.

In the first stage, the primal mesh can be generated in several ways since no specific property of the mesh is required in the discretization through Section 6.3.3. In most previous graph based surface reconstruction methods, structured grids are utilized, which also work well in our method. However, the direct surface extraction from a structured grid and graph cut would result in a staircase surface. Hence some preprocessing or postprocessing is required to obtain a triangular surface mesh without any undesirable “blockiness”. On the contrast, by utilizing a tetrahedral mesh, the desirable triangular surface mesh can be readily obtained from the minimal cut in the proposed method. Compared with the deliberate design of the octahedron subgraph [61] or generating a triangle fan [60], the method proposed in this paper is simple and elegant, though with extra cost in generating the tetrahedral mesh, which can be considerably reduced by means of the well-developed triangulation technique and the adaptive mesh generation. The implementation of the classical incremental Delaunay triangulation algorithm [15, 12] provided by the Computational Geometry Algorithms Library (CGAL) [1] is applied in the experiments.

Before the end of the first stage, some operation on mesh points distribution is required since mesh points too close to the data set may destroy the faces supposed to form the desired homeomorphic surface as in Figure 6.3(a). Centroidal Voronoi tessellation (CVT) can help move the mesh points away from the potential surface and more details about CVT are presented in [34, 41, 44]. The result after CVT optimization is shown in Figure 6.3(b). Simply deleting the points within a small vicinity of the data set is another way, in which a good estimation for the “safe distance” is required.

In the second stage, the crust region is first constructed as the n -ring neighborhood

Figure 6.3: This figure shows (a) an undesirable resulting surface caused by too close mesh points and (b) the desirable surface after CVT optimization.

of the data set S followed by the specification of both the indicator and the confidence functions. Next, the structure of the Delaunay triangulation is exploited to efficiently compute the distance function. This can be achieved because a vertex in Delaunay triangulation only connects the nearest vertices. In the graph construction, only the distance function of points in crust is of interest, which will be explained in the next paragraph. Therefore, instead of calculating distances from the mesh points to the data set, we calculate the distances from the data set to its nearby mesh points, i.e. points in the crust region. This fast algorithm utilizes the Delaunay triangulation data structure requiring no more storage and the distance calculation can be accomplished efficiently during the crust establishment.

In the third stage, the dual graph is constructed following the instructions in Section 6.3.3. In the implementation, some computation of edge weights can be saved and a fast graph construction algorithm is proposed as follows. For each K_i in the crust region, x_i does not have any valid t -links according to (6.20). For each K_i out of the crust region, either of x_i 's t -links is zero and the other is a relatively large value c . It is safe to assert $l_i = 1$ if $s_i = c, t_i = 0$ and vice versa. The minimal cut would not cross the edges out of crust region provided c is large enough. Hence it is not necessary to calculate the weight of n -link (x_i, x_j) when K_i and K_j are out of the crust. In other words only the n -links in the crust region is of interest. It is sufficient to calculate the distance function of points in the crust region. To keep the graph

Table 6.2: Graph Construction Procedure

Inputs	<ol style="list-style-type: none"> 1. Mesh topological information <ol style="list-style-type: none"> a. tetrahedra $\{K_i\}_{i=1}^N$ b. triangle faces $\{F_r\}_{r=1}^R$ 2. <i>Crust</i> 3. Indicator function $I(K_i)$ and Confidence function $\beta(K_i)$
Algorithm	<ol style="list-style-type: none"> 1. Add x_i to Graph G for every K_i %% N-links 2. Add edges (x_i, x_j) and (x_j, x_i) to G for every $F_r = K_i \cap K_j$ 3. If any of K_i and K_j is in <i>Crust</i> 4. Assign $(d_{ij} + \alpha)S_{ij}$ as weights of (x_i, x_j) and (x_j, x_i) 5. Else 6. Assign c as weights of (x_i, x_j) and (x_j, x_i) %% T-links 7. Assign $I(K_i) - 2 \beta(K_i)$ to s_i for every x_i 8. Assign $I(K_i) - 1 \beta(K_i)$ to t_i for every x_i
Outputs	Graph G

construction procedure logical and clear, we use c -weighted n-links to connect nodes out of the crust. Finally the whole algorithm of the graph construction is given in Table 6.3.

In the third stage, the dual graph is constructed following the instructions in Section 6.3.3. In the implementation, some computation of edge weights can be saved and a fast graph construction algorithm is proposed as follows. For each K_i in the crust region, x_i does not have any valid t -links according to (6.20). For each K_i out of the crust region, either of x_i 's t -links is zero and the other is a relatively large value c . It is safe to assert $l_i = 1$ if $s_i = c, t_i = 0$ and vice versa. The minimal cut would not cross the edges out of crust region provided c is large enough. Hence it is not necessary to calculate the weight of n-link (x_i, x_j) when K_i and K_j are out of the crust. In other words only the n-links in the crust region is of interest. It is sufficient to calculate the distance function of points in the crust region. To keep the graph

Table 6.3: Graph Construction Procedure

Inputs	
1.	Mesh topological information
a.	tetrahedra $\{K_i\}_{i=1}^N$
b.	triangle faces $\{F_r\}_{r=1}^R$
2.	$Crust$
3.	Indicator function $I(K_i)$ and Confidence function $\beta(K_i)$
Algorithm	
1.	Add x_i to Graph G for every K_i %% N-links
2.	Add edges (x_i, x_j) and (x_j, x_i) to G for every $F_r = K_i \cap K_j$
3.	If any of K_i and K_j is in $Crust$
4.	Assign $(d_{ij} + \alpha)S_{ij}$ as weights of (x_i, x_j) and (x_j, x_i)
5.	Else
6.	Assign c as weights of (x_i, x_j) and (x_j, x_i) %% T-links
7.	Assign $ I(K_i) - 2 \beta(K_i)$ to s_i for every x_i
8.	Assign $ I(K_i) - 1 \beta(K_i)$ to t_i for every x_i
Outputs	Graph G

construction procedure logical and clear, we use c -weighted n-links to connect nodes out of the crust. Finally the whole algorithm of the graph construction is given in Table 6.3.

Subsequently the min-cut/max-flow algorithm [18] is applied on the obtained graph. After the minimal cut is obtained, the reconstructed surface can be easily extracted from the primal mesh according to Table 6.1. Finally, a number of figures in Figure 6.4 illustrate all stages.

6.4 Multiphase Surface Reconstruction

In Section 6.3, two phase surface reconstruction is considered, in which the surface is obtained as the interface of two partitions. When the phase number increases up to more than two, the corresponding min-cut/max-flow problem becomes a multi-way

Figure 6.4: Main stages in our method

cut problem. Some approximation methods, such as α expansion [19], have been developed to solve multi-way cut problems. For more approximation methods, we refer to [73, 76]. Recently a new method to find the solution of a multi-way cut problem is presented in [8]. In this section, the surface reconstruction method is extended from two phase cases to multiphase cases by adapting the multilayer graph idea in [8]. To the best of our knowledge, this is the first time that multiphase surface reconstruction problems, which ubiquitously occur in CAD, animation and medical imaging applications, are solved by the graph cut technique. A brief review of the multilayer graph idea in [8] is provided in Section 6.4.1 for completeness and the method for multiphase surface reconstruction is presented in Section 6.4.2. An example illustrating the robustness of the proposed method concludes this section.

6.4.1 Multiphase Segmentation Problems

In [8], the segmentation problem is considered, aiming to find a partitioning of the domain Ω which satisfies

$$\bigcap_{m=1}^M \Omega_m = \emptyset, \quad \bigcup_{m=1}^M \Omega_m = \Omega. \quad (6.25)$$

The piecewise constant level set function $\phi(x) = m$ for $x \in \Omega_m$, $m = 1, 2, \dots, M$. The surfaces are interfaces between partitions presented by the discontinuities of ϕ . In multiphase cases, surfaces can be considered as a hyper-surface in a higher dimensional domain. The level set function

$$\phi(x) = m \text{ in } \Omega_m, \quad m = 1, 2, \dots, M, \quad x \in R^d \quad (6.26)$$

can be represented as

$$\phi(x) - z = 0, (x, z) \in R^d \times \{1, 2, \dots, M\}. \quad (6.27)$$

Therefore, one extra dimension is introduced to handle the multiphase issue. The multi-way cut problem in the original graph can be translated to a binary cut problem

in a new multilayer graph. A brief description of the translating algorithm is given as follows.

Assume the original graph contains N nodes $\{x_i, i = 1, \dots, N\}$ and the phase number is M . Then a new graph of $M - 1$ layers is constructed, which contains $N \times (M - 1)$ nodes $\{x_{i,k}, i = 1, \dots, N, k = 1, \dots, M - 1\}$. The nodes $\{x_{i,k}, 1 \leq i \leq N, k = 1, \dots, M - 1\}$ are $M - 1$ images of x_i . The subset of nodes $\{x_{i,k}, i = 1, \dots, N, 1 \leq k \leq M - 1\}$ forms the k -th layer in the graph.

The edges in the new graph can be classified into two groups: horizontal edges and vertical edges. Horizontal edges include the edges connecting nodes in the same layer while vertical edges include t-links and the edges connecting nodes in different layers. For every edge (x_i, x_j) in the original graph, there are $M - 1$ images in the new graph: $\{(x_{i,k}, x_{j,k}), 1 \leq k \leq M - 1\}$, whose weights are the same with that of (x_i, x_j) . Hence every layer is a replica of the original graph except the t-links. Vertical edges include $\{(x_{i,k}, x_{i,k+1}), i = 1, \dots, N, k = 1, \dots, M - 2\}$ and $\{(s, x_{i,1}), (x_{i,M-1}, t), i = 1, \dots, N\}$. The vertical edge weight assignment has something to do with the application and is left to be discussed in Section 4.2.

After applying the min-cut/max-flow algorithm to the new graph, each node's label in the original graph can be determined by the following equation based on the obtained minimal cut C ,

$$\phi_i = \begin{cases} 1 & (s, x_{i,1}) \in C \\ l + 1 & (x_{i,l}, x_{i,l+1}) \in C \\ M & (x_{i,M-1}, t) \in C \end{cases} . \quad (6.28)$$

The equation (6.28) converts the binary partitioning of the multilayer graph to a multi-partitioning of the original graph and hence the multi-way cut problem is solved.

We borrow the one dimensional example in [8] to illustrate the above algorithm. The original graph contains six nodes. Figure 6.5(a) and 6.5(c) show the binary cut and the segmentation result. When a four way cut is required, a three layer graph is built as in Figure 6.5(b), in which every node in the original graph has three images. The horizontal edges are replicas of n-links in the original graph while the vertical

Figure 6.5: One dimensional example to illustrate multilayer graph

edges connect the neighboring layers and terminals. After applying the min-cut/max-flow algorithm to the new graph, the converted result is obtained as shown in Figure 6.5(d).

6.4.2 The multiphase Method

In this chapter, the multilayer graph idea used in [8] is adapted to handle multiphase cases. Given a data set, the primal mesh is generated and the original graph is constructed in the same way to that of Section 6.3. Then the multilayer graph as well as the horizontal edge weights is established following the instructions in Section 6.4.1. Once the assignment of vertical edge weights is finished, the graph cut algorithm is applied and equation (6.28) converts the min-cut to the correct partitioning, from which the corresponding surfaces are subsequently extracted. The remaining of this subsection will describe how to determine vertical edge weights in the multilayer graph.

Let Ω and K be the same as in Section 6.3. The *Crust* separates Ω into several disconnected subdomains $\{\Omega'_i\}_{i=1}^M$, on which the indicator, confidence and level set

functions can be defined as follows,

$$I(K_i) = \begin{cases} m & \text{if } K_i \in \Omega'_m \\ 0 & \text{if } K_i \in Crust \end{cases}, \quad (6.29)$$

$$\beta(K_i) = \begin{cases} 0 & \text{if } K_i \in Crust \\ c & \text{others} \end{cases}, \quad (6.30)$$

and

$$\phi_\Gamma(K_i) = m, \quad K_i \in \Omega_m, \quad (6.31)$$

where $\{\Omega_m\}_{m=1}^M$ are partitions induced by the resulting surface Γ . Consider a tetrahedron $K_i \in \Omega'_m$. The indicator function $I(K_i) = m$, which results in that the level set function $\phi_\Gamma(K_i)$ induced by the reconstructed surface Γ has to be m . Otherwise, any deviation between the level set function and the indicator function, i.e. $\phi_\Gamma(K_i) = n \neq I(K_i) = m$, would keep the energy functional $E(\Gamma)$ from being minimized. By recalling (6.28), this assertion can be achieved by assigning vertical edge weights as shown in Table 6.4.

For the tetrahedron $K_i \in Crust$ and corresponding node x_i , no t-link or vertical edge is needed for $x_{i,j}$ since the min-cut is totally determined by the geodesic defined by $\int_\Gamma (d(x) + \alpha) ds$ which corresponds to n-links in the original graph or horizontal edges in the multilayer graph. Therefore, those vertical edges are simply assigned to be zero and this policy works well in all experiments. This is equivalent to imposing hard boundary conditions [16] or the narrow-band implementation [60]. An example of two intersecting spheres is illustrated in Figure 6.6, where the zero weighted edges are removed from the graph for clearer demonstration of vertical edge weight distribution. Thus far, two phase cases and multiphase cases are unified into a general graph framework, which can be implemented in a single algorithm shown in Table 6.5, where s is considered as the 0-th layer and t the M -th one, i.e. $x_{i,0}, x_{i,M}, 1 \leq i \leq K_N$ stands for s and t respectively. The multilayer graph such as Figure 6.5(b) automatically degrades to a single layer graph like Figure 6.5(a) as long as the phase number decreases to two.

Concerning the phase number, it can be computed by region growing instead of

Figure 6.6: This figure illustrates the multilayer graph structure by two intersecting spheres. (a) presents the cut view of the mesh, where red crust separates the domain into four regions marked with different colours. The corresponding multilayer graph is shown in (b), in which $I(K_i) = 1, 2, 3, 4$ when K_i is blue, green, purple, or brown. The nodes in the graph correspond to the tetrahedra with the same color. The weights of vertical edges depend on $I(K_i)$. Therefore, some vertical edges vanish as shown in (b) and the red nodes do not have vertical edges at all.

Table 6.4: Weights for vertical edges

	$I(K_i) = 0$	$I(K_i) = 1$	$I(K_i) = 2$	$\dots$	$I(K_i) = k$	$\dots$	$I(K_i) = M$
$w(s, x_{i,1})$	0	0	c	$\ddots$	c	$\ddots$	c
$w(x_{i,1}, x_{i,2})$	0	c	0	$\ddots$	c	$\ddots$	c
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$
$w(x_{i,k-1}, x_{i,k})$	0	c	c	$\ddots$	0	$\ddots$	c
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$
$w(x_{i,M-1}, t)$	0	c	c	$\ddots$	c	$\ddots$	0

Table 6.5: Graph Construction for Multiphase Problem

Inputs	<ol style="list-style-type: none"> 1. Mesh topological information <ol style="list-style-type: none"> a. tetrahedra $\{K_i\}_{i=1}^N$ b. triangle faces $\{F_r\}_{r=1}^R$ 2. <i>Crust</i> 3. Phase number M
Algorithm	<ol style="list-style-type: none"> 1. For every K_i and every $k = 1, \dots, M - 1$ layer 2. Add $x_{i,k}$ to Graph G %% horizontal edges 3. For every $F_r = K_i \cap K_j$ and every $k = 1, \dots, M - 1$ layer 4. Add $(x_{i,k}, x_{j,k})$ and $(x_{j,k}, x_{i,k})$ to G 5. If any of K_i and K_j is in <i>Crust</i> 4. Assign $(d_{ij} + \alpha)S_{ij}$ as weights of $(x_{i,k}, x_{j,k})$ and $(x_{j,k}, x_{i,k})$ 5. Else 6. Assign c as weights of $(x_{i,k}, x_{j,k})$ and $(x_{j,k}, x_{i,k})$ %% vertical edges 7. For every $K_i \notin \text{Crust}$ and every $k = 0, \dots, M - 1$ layer 8. If $I(K_i) \neq k$ 9. Add a c-weighted edge $(x_{i,k}, x_{i,k+1})$
Outputs	Graph G

interactive specification. The computation is usually sensitive to the mesh, especially in some complex topologies, nevertheless the proposed method can handle this uncertainty without any trouble. One example is presented to show the robustness of our method and also to conclude this section on multiphase problems. The data set in Figure 6.7(a) is sampled from a femur surface. The original surface is a single watertight one. In an ideal mesh, the crust should separate the domain into two subdomains, the inner one of which is an integral cavity as in Figure 6.7(b), then a two phase partitioning can be obtained as shown in Figure 6.7(c). However, if the mesh points selection is not perfect, some topological change happens as establishing the crust. For example, in the inner part of the femur, the cavity may be split into two or more subdomains as shown in Figure 6.7(d) while the partitioning result goes as in Figure 6.7(e). Two different meshes lead to two different graphs, however, either of the reconstructed results is a reasonable solution to the same problem. Comparing two reconstructed results, it can be found that two results are almost the same except that the multiphase solution has one more internal surface, which is redundant however can be easily removed by merging the phases. The desired surface is shown in Figure 6.7(f). What is worth mentioning is that when the reconstruction problem becomes more ill-posed, such as the data set is more insufficiently sampled, and the noise level is higher, more unfaithfulness is likely to be observed in the result. One example is two disjoint surfaces separated by a small gap, whose size is comparable to the data density. The regularization term tends to fill that gap and a connected surface may be obtained. This topological unfaithfulness can not be as easily removed as in Figure 6.7(e). The ill-posedness of this degree means the information carried by the insufficient data set only suggests a rough profile instead of an exact solution. In this case, we consider both the separated surfaces and the connected surface reasonable results.

6.5 Examples

In this section, various examples are presented to illustrate the effectiveness, efficiency and robustness of the proposed method. All experiments were conducted on a PC

Figure 6.7: The femur example shows robustness of multiphase surface reconstruction

Figure 6.8: A typical multiphase model without intersection, which consists of four concatenating tori.

with Intel Pentium 4 CPU of 3.2GHz and 4GB memory. The Dragon, the Buddha and the armadillo data were obtained from the Stanford 3D Scanning Repository (<http://graphics.stanford.edu/data/3Dscanrep/>). The hand skeleton and the horse data were obtained from Large Geometric Models Archive of Georgia Institute of Technology (http://www.cc.gatech.edu/projects/large_models/). Those intersecting objects were synthesized by ourselves and other models were obtained from Digital Shape Workbench (<http://dsw.aimatshape.net/>). Only points locations were used. The reconstruction surfaces were rendered by MeshLab (<http://meshlab.sourceforge.net/>).

Figure 6.10 presents a femur nailed by a stick, which is a case widely occurring in the femur surgery. In previous simulations, two objects have to be reconstructed

Figure 6.9: Several multiphase examples with intersections. (a) is one cube intersecting one sphere. The phase number is four. (b) is eight cubes intersecting one sphere. The phase number is eighteen. (c) is four intersecting spheres, any subset of which has a non-empty intersection. The phase number is sixteen.

separately. But in our method two objects are reconstructed simultaneously and intersections are automatically generated. Figure 6.10(a)-(d) show the details from different angles and Figure 6.10(e) illustrates the cut view of the intersection part.

Figure 6.11 shows several classic examples in the surface reconstruction field: the hand skeleton, the armadillo and the horse. Figure 6.12 and 6.13 show another two classic examples in different resolutions: Happy Buddha and Dragon. In high resolutions, surface features are clear and details are well preserved. In low resolutions, profiles are roughly but faithfully maintained with a few topological distortions.

Figure 6.14 shows a group of sculpture reconstruction examples. The patterns on Pegasus and Gargoyle wings provide more examples of delicate details. Furthermore, both Ramesses and Neptune sculptures contain smooth human bodies and sharp rigid features like Ramesses' stony seat and Neptune's trident. These mixed style surfaces are not challenges to our method. The example shown in Figure 6.15(a) is a raptor with an elongated tail and sharp claws. The details of one front claw are illustrated in Figure 6.15(b).

Figure 6.16 shows a real example. The statue 'Khmer's smile' was scanned by our 3D scanner and a data set of 20,663 points was obtained. To remove noises introduced by the scanner, the surface has been smoothened by increasing the regularization

Figure 6.10: This model is the identical femur in Figure 6.7 but nailed with a cylindrical object from the top. The intersecting part is presented and the interface is zoomed for details. A cut-view is provided as well.

Figure 6.11: This figure shows several classical examples in the surface reconstruction field: (a) Hand skeleton, (b) Armadillo and (c) Horse.

coefficient. The result keeps most features and one can still read the subtle smile from the reconstructed surface.

The noise level in Khmer’s smile example is relatively low. To study the performance of the proposed method in highly noisy cases, two data sets with higher noise levels are investigated as shown in Figure 6.17. One is with noises around one point and the other along the sphere equator. Figure 6.17 shows the results with different regularization coefficient α in (6.10). When the regularization coefficient α decreases to zero, our model becomes Zhao’s model in [120]. It can be seen from Figure 6.17(b) and (f) that few noises have been removed in these cases. By increasing the regularization coefficient most noises have been removed as shown in (c) and (g). However there still exist some remaining noise points forming the ‘scar’ surfaces shown in (d) and (h). The denoising becomes extremely difficult because the noise level is comparable to the data density.

In Figure 6.18, a technique called Feature Swap for preserving features such as ridges and corners is illustrated. This technique requires some a priori knowledge such as some critical points presenting the features. The proposed variational model tends to smoothen the features with large curvatures as a side effect of the regularization term whose original purpose is for denoising. Though being a good solution for the

Figure 6.12: This figure shows another classical example in different resolutions: Dragon. (a) shows the triangular surface mesh of the low resolution model and (b) shows the rendered surface. (c) shows the rendered surface in the high resolution.

Figure 6.13: This figure shows another classic example in different resolutions: Happy Buddha. (a) shows the triangular surface mesh of the low resolution model and (b) shows the rendered surface. (c) shows the rendered surface in the high resolution.

Figure 6.14: This figure shows a group of mythologic and ancient sculptures. The first two models, Pegasus in (a) and Gargoyle in (b), are used to show the ability of handling shallow patterns. The last two models, Ramesses in (c) and Neptune in (d), are used to show the performance on one single model containing both rigid sharp features and a smooth human body.

Figure 6.15: This figure shows an example of a raptor model. (a) Rendered reconstructed surface; (b) details of the front claw. The long tail and sharp claws are both faithfully reconstructed.

Figure 6.16: This figure shows a real example of a Cambodia statuette named ‘Khmer’s Smile’, in which the data set is acquired by our 3D scanner. The picture of the statuette is shown in (a) for comparison. (b) presents the data set points. The reconstructed surface is shown in (c), in which the subtle smile in the Buddha’s face still can be read.

Figure 6.17: This figure demonstrates two groups of noise removal experiments. The data set in this series of examples is sampled from a sphere surface into which two different types of artificial noises are added. In (a) noise points (red points) are around one location while in (e) noise points are along the sphere equator. (b) and (f) show the reconstructed surfaces with the regularization coefficient $\alpha = 0$. Results obtained with $\alpha = 0.001$ are shown in (c) and (g). The remaining noise points and these ‘scar’ surfaces are zoomed in (d) and (h).

general surface reconstruction problem, the smoothed surface is undesirable in the CAD industry where the rigid features are important. The fact that all data points are a part of the mesh framework makes possible the existence of a surface triangulation homeomorphic to the ground truth, which is a prerequisite of a feature preserving reconstruction method. The Feature Swap method is proposed as a post-processing to recover all features described by some critical points.

The main idea of Feature Swap is to ensure the presence of every critical point in the reconstructed surface. The detailed algorithm is described in Table 6.6. The first half part of the algorithm intends to recover the presence of all critical points in the reconstructed surface. After executing this part, the smoothened surface in Figure 6.18(a) is turned to (b), in which most features are recovered while three corners still suffer from distortions. The second half part of the algorithm aims to regularize these undesirable distortions by some measures such as area and curvature. In this study, the area of the reconstructed surface is used. The result in (b) will be turned to the surface in (c) which has all features preserved. Details on the feature are shown in (d) and (e). It is worth pointing out that this Feature Swap method is strongly dependant on a proper mesh. As a heuristic but empirically useful post-processing to recover features of a surface, this method is still under investigation and an important aspect of our future work.

Finally, an example of a chameleon model concludes this section, which contains most difficulties in surface reconstruction problems: undersampling, non-uniformity, noises and thin parts. The elongated tongue contains less than 200 sample points. Four or less points are sampled in a contour of tongue. This poorly sampled data set presents a challenge to all reconstruction methods. With the properly generated mesh, careful selection of the parameter and the Feature Swap technique the surface is reconstructed as shown in Figure 6.19(a) and the tongue details are shown in (b) and (c). The reconstruction from this poorly sampled points proves the robustness and effectiveness of the proposed method.

Table 6.7 gives the sizes of the data sets involved in our examples and CPU time counted in seconds. The first column of Table 6.7 gives the names of examples, where ‘Low Res’ means ‘low resolution’ and ‘High Res’ means ‘high resolution’. The second

Table 6.6: Feature Swap Algorithm

Inputs	<ol style="list-style-type: none"> 1. Mesh information <ol style="list-style-type: none"> a. tetrahedra $\{K_i\}_{i=1}^{N_K}$ b. triangle faces $\{F_i\}_{i=1}^{N_F}$ 2. Critical Points, $\{P_i\}_{i=1}^{N_P}$ 3. The partitioning, $L : \{K_i\}_{i=1}^{N_K} \rightarrow \{1, 2\}$ 4. The surface $\Gamma_h = \bigcup_{L(K_i) \neq L(K_j)} (K_i \cap K_j)$. 5. The presence number $PR(\Gamma_h) = \sum_{i=1}^{P_N} \mathbf{1}_{\{P_i \in \Gamma_h\}}$.
Algorithm	<ol style="list-style-type: none"> 1 For $i = 1 : N_P$ 2 If $P_i \notin \Gamma_h$ 3 Find K_m incident to P_i, $l = L(K_m)$ 4 Find the minimal N, $\exists F_j$ in N-ring neighborhood of P_i <ol style="list-style-type: none"> such that $F_j \subset \Gamma_h$ 5 Find the segment seg between P_i and the centroid of F_j 6 Find all M tetrahedra intersecting seg, $\{K_{N_j}\}_{j=1}^M$ 7 $L(K_{N_j}) = \{1, 2\} \setminus l, j = 1, \dots, M$ 8 End If 9 End For 10 For $i = 1 : N_K$ 11 If K_i has a neighbor K_j such that $L(K_i) = l_1 \neq L(K_j) = l_2$ <ol style="list-style-type: none"> 12 $L(K_i) = l_2$ and obtain the new surface Γ'_h 13 If $Area(\Gamma'_h) > Area(\Gamma_h)$ or $PR(\Gamma'_h) < PR(\Gamma_h)$ 14 $L(K_i) = l_1$ 15 End If 16 End If 17 End If
Outputs	<p>The new surface $\Gamma_h = \bigcup_{L(K_i) \neq L(K_j)} (K_i \cap K_j)$</p>

Figure 6.18: This figure demonstrates a technique called Feature Swap used to recover important features like edges and corners. The data set is sampled from a cut-an-eighth sphere surface. The initial reconstruction surface is shown in (a), in which sharp parts have been smoothed due to the regularization term in the proposed variational model. As shown in (b), most of edges have been recovered by the first half part of Feature Swap operation except the corners. These corners are recovered as in (c) after the Feature Swap method is finished. Details of (a) and (c) are shown in (d) and (e).

column contains the number of data points for each example and the corresponding Delaunay insertion time is given in the third column. The fourth and the fifth columns give the numbers of background mesh points and mesh elements. The CPU time given in the sixth column is the time consumed by graph construction while the CPU time in the last column is the graph cut time. It can be observed that the most time-consuming procedure is the mesh generation in two phase examples while the graph cut time dominates in multiphase examples. The overhead caused by the mesh generation is worthwhile since the surface conversion is avoided and the feature preserved technique can be applied. In Table 6.7, the femur example spent $0.07s$ on data inserting, $0.025s$ on graph construction and $0.005s$ on graph cut. The large model raptor spent $38.82s$, $25.33s$ and $21.77s$ on data inserting, graph construction and graph cut respectively. Above examples have demonstrated the efficiency of the proposed method in complicated and realistic applications. It should be noticed that the data inserting time is roughly proportional to the data size, and the similar relation exists between the mesh element number and the graph construction time. The graph cut time strongly depends on element number or graph size, and interestingly, it also depends on the surface shape in a more complicated way, which is still under investigation.

6.6 Concluding Remarks

In this chapter, a variational method for surface reconstruction is proposed based on Delaunay triangulation and graph cut. The variational model is based on the weighted minimal surface model [120] with an additional regularization term and special fidelity terms as boundary conditions. The graph cut technique is used to minimize the energy functional in the variational model. Different from previous graph-cut based method, the Delaunay based mesh is applied.

The multilayer graph is applied to solving multiphase problems by converting the multi-way cut problem to a binary cut one. It is the first attempt to solve multiphase surface reconstruction problems by the graph cut technique. In some complex cases whose phase numbers are ambiguous, different meshes and parameters may lead to

Figure 6.19: This figure shows an example of chameleon. Thin part, i.e. chameleon's tongue, consists of very sparse sample points as shown in (b). Due to the properly generated mesh, carefully selection of the parameter and the Local Swap technique, the surface is successfully reconstructed as shown in (a). The most difficult part tongue is shown in (c).

Table 6.7: Statistics of surface reconstruction examples

Example	Data Set	Data Insert Time	Mesh Points	Mesh Elements	Graph Built Time	Graph Cut Time
Femur	2020	0.07	1330	17735	0.025	0.005
Perforated Cube	7672	1.83	38160	244935	0.52	0.24
Two Perforated Cubes	15344	3.09	19693	204989	0.55	1.53
Knot	478704	17.63	185245	3655240	9.59	4.83
Four Tangling Tori	32768	1.78	133686	1007425	9.23	26.79
A Cube and a Sphere	11692	0.43	4924	83557	0.25	2.39
8 Cubes and a Sphere	54714	2.99	132584	928907	11.86	381.546
Four Spheres	22184	0.67	21176	248504	2.78	54.56
Nailed Femur	36968	1.72	102850	884783	4.64	29.85
Golf	35984	2.35	6573	138440	0.29	0.12
Hand Skeleton	327323	15.6	340940	4025863	12.98	24.91
Horse	494195	20.03	471649	5173576	13.52	6.96
Armadillo	172974	6.36	86828	1455076	3.63	1.65
Happy Buddha Low Res	7108	0.24	9628	95334	0.19	0.07
Happy Buddha High Res	543652	21.19	328651	5006848	12.76	10.88
Dragon Low Res	5205	0.17	4962	63347	0.12	0.07
Dragon High Res	437645	17.29	238633	3840493	10.08	6.55
Kid Bust	255358	9.52	124399	2234355	5.89	3.09
Kitten	137098	4.86	58058	1099154	2.93	1.35
Bimba	74764	2.65	31747	560483	1.4	0.6
Pegasus	63544	2.29	27278	522159	1.21	0.67
Gargoyle	863210	32.74	456579	7651135	20.18	10.68
Ramesses	193252	7.08	75835	1481239	3.88	1.84
Neptune	205835	8.03	113146	1733813	4.49	2.06
Raptor	1000080	38.82	785813	10340012	25.33	21.77
Khmer's Smile	20664	0.73	16930	216104	0.47	0.24
Chameleon	4594	0.16	308636	1737455	8.15	1.19

different reconstructed surfaces, all of which, however, are reasonable. The example in Section 6.4.2 has already shown the robustness and the flexibility in multiphase problems. Limitation of the proposed method on insufficient data set is also discussed.

The Delaunay triangulation not only makes possible the existence of a homeomorphic sub-complex, but also accelerates the calculation of the distance function and save the overload of cut-surface conversion. Therefore, our new method does not require any post-processing, such as Laplacian smoothing used in the last chapter, to eliminate the artifacts caused by discrete representation. Moreover, the Delaunay-based mesh needs much fewer nodes than the uniform mesh, which dramatically saves the storage for huge graph in 3D. This is extremely important for 3D multiphase reconstruction, since the memory request of using uniform meshes will be unaffordable.

What is worth mentioning is that the faithfulness of the reconstructed surface is strongly dependant on the sufficiency of the sampled data set. To tackle problems brought by the insufficient data sampling, a post-processing Feature Swap is introduced in order to recover those features which are beyond the presenting ability of the data set. This Feature Swap will help present sharper and clearer features as in Figure 6.18(c), which is desirable in the CAD industry. However this technique is still in the experimental period and some heuristics is required. More robust and efficient method is expected in our future work.

Chapter 7

Conclusion

In this chapter, various examples are used to demonstrate the advantages and disadvantages of the different methods used in the previous chapters. Also, a qualitative comparison will be given at the end of this chapter.

First, the comparison is conducted between the level-set modeling and the phase-field modeling. Figure 7.1 depicts an example of a noisy rabbit which has been used in Chapter 2. Now, this example is used to demonstrate the efficiency of the proposed PFMCS method. The level set method and the PFMCS method is used to smooth the noisy curve as shown in Figure 7.1(a) respectively. Both methods are solved on the same uniform 400×400 rectangular mesh. In order to measure the smoothing effects of the different methods, as before, the normalized symmetric area difference (NSAD) is computed for the result produced by each method,

$$NSAD(C_{noisy}, C_{smooth}) = \frac{A_{diff}}{A_{noisy}},$$

where A_{diff} is the symmetric area difference of the regions enclosed by the noisy curve C_{noisy} and the smoothed curve C_{smooth} , and A_{noisy} is the area of the region surrounded by the the noisy curve C_{noisy} . To fairly compare the efficiency of the level-set method and PFMCS method, the noisy curve in Figure 7.1(a), denoted as C_{noisy} will be smoothed to the curve with a fixed NSAD. For instance, the curve illustrated in Figure 7.1(b) is produced by the PFMCS method, denoted as C_{pf} , and

	Phase Field	Level Set
CPU time	5.03s	3124.35s
NSAD	0.00623	0.00684

Table 7.1: CPU time Comparison

the corresponding $NSAD(C_{noisy}, C_{pf}) = 0.00623$. The curve presented in Figure 7.1(c) is produced by the standard level-set implementation, denoted as C_{ls} , and its $NSAD(C_{noisy}, C_{ls}) = 0.00684$. Thus, two curves C_{pf} and C_{ls} have almost the same NSAD to the initial noisy curve, and also, it can be observed from the figures that these two curves are almost visually identical. However, the computation times of two methods are quite different, which can be seen clearly from the Table 7.1. The level-set implementation spends 3124 seconds to obtained the smoothed curve, which is almost the same as the result produced by the PFMCS method in merely 5 seconds.

Since the nonlinearity of the phase-field modeling is lower than the level-set modeling, the corresponding solution algorithms are generally faster and more robust than the level-set method. The reason that phase-field method has lower nonlinearity is because it uses diffuse interfaces in the modeling, and the phase-field model only converges to sharp interface case as the transitional thickness goes to zero. In another words, some model errors are introduced in phase-field modeling. A deeper analysis can be found in [112] and references cited therein. The following example visually shows how the transitional parameter affects the solution in curve smoothing application.

Figure 7.2(a) depicts a noisy outline of a dog. The PFMCS is used to smooth the noisy curve. As usual the phase function assumes the form $\tanh(d(x)/\sqrt{2}\epsilon)$, where $d(x)$ is the signed distance function, and a 400×400 rectangular mesh is used. Figure 7.2(b) shows the result with $\epsilon = 0.005$. Figure 7.2(c) shows the result with exact the same setting except that $\epsilon = 0.01$. It is clear that the curve becomes smoother. Figure 7.2(d) presents the result with $\epsilon = 0.02$, and it is even smoother than the one in Figure 7.2(c). Thus, as transitional layer becomes thicker the diffusive effects become more and more evident. The above phenomenon also occurs in other applications [112].

(a) Noisy rabbit

(b) Smoothed rabbit processed by phase field method

(c) Smoothed rabbit processed by level set method

Figure 7.1: Examples of the noisy rabbit

Figure 7.2: Examples of the noisy dog. (a) Noisy curve; (b) Smoothed curve with $\epsilon = 0.005$; (c) Smoothed curve with $\epsilon = 0.01$; (d) Smoothed curve with $\epsilon = 0.02$.

The previous two examples shows the characteristic of the phase-field method, and the next two examples will reveal some traits of the graph-cuts method. As before, the uniform mesh is used in all example. In chapter 4, we have seen that graph-cuts methods are much more efficient than the level-set method. Another advantage of graph-cuts method is that it is able to produce global minimizers in the give discrete space.

We consider an example of a curve which is the digitized outline of a cat with lots of zig-zags, as shown in Figure 7.3(a). The smoothed curve is the minimizer of the functional (7.1),

$$\mathcal{E}_1(C, \alpha, \beta) = \int_0^1 (\alpha + \beta \varphi^0(C(\tau))) |C_\tau| d\tau \quad (7.1)$$

where φ^0 is the unsigned distance function induced by the noisy curve. This is a non-convex functional and there is no unique global minimizer for it. The second row in Figure 7.3 shows the results of level-set implementations while the third row presents the results of graph-cuts implementations of the optimizations of the functionals we consider. Figure 7.3(b) and 7.3(c) present the results with different curve initialization. In Figure 7.3(b), the initial curve was a small circle inside the cat, and after 200,000 iterations the evolution is stuck at a local minimum, as shown. In Figure 7.3(c), the initial solution is chosen as a large circle around the cat and after 500,000 iterations the evolution stops. It is obvious that with a large circle as the initial guess the final solution is much better than taking the small circle as the initial solution. However, both of them missed the concave part of the curve. There is no initialization issue for graph-cuts implementation, however using graph-cut to minimize functional (7.1), one has to resort to narrowband implementations. Figure 7.3(d) and 7.3(e) present the results with different widths of the narrowband, both of which are solved by graph-cuts. With a very large bandwidth, the result (Figure 7.3(d)) misses all elongated parts. Comparing to Figure 7.3(d), the result in Figure 7.3(e) is much better due to a smaller bandwidth. The efficiency aside, the results generated by the graph-cuts method are better than the ones produced by the level-set implementation.

Figure 7.3: A cat with a lot of zig-zags. (a) The noisy curve; (b) the level-set solution of with a small circle as the initial solution; (c) the level-set solution with a large circle as the initial solution; (e) the graph-cuts solution of with a large narrowband; (f) the graph-cuts solution with a relatively small narrowband. 400*400 resolution is used for level-set implementations and 800*800 resolution is used for graph-cuts implementations.

However, the graph-cuts method also has its own drawback as shown in the following examples. Figure 7.4 gives an example of a noisy square which is processed by graph-cuts and level set methods. In this example the 64-neighbourhood system is used for graph cuts method to reduce metrication errors. However, since the result obtained by graph cut method is actually only the labellings and the curve is obtained by connecting discrete girds, the resulting curve in Figure 7.4-(b) is still not as smooth as the one in Figure 7.4-(c) obtained by the level set method, which is the interpolation of zero level-set of the signed distance function. This is a typical drawback for discrete methods such as graph-cuts or GEWCVT methods.

The next example shows the differences between the multi-phase graph-cuts method and the GEWCVT method. Although both methods work for discrete valued function and inevitably introduce some model errors for continuous models, the magnitude of the model errors of two methods are different. Graph-cuts method has notorious metric errors [16, 17] caused by using anisotropic total variation approximation and neighborhood systems. On the other hand, the GEWCVT method introduces the error by approximating the curve by straight line in a small neighborhood. The following triple junction example will qualitatively show that the model error of GEWCVT method is smaller than the one of graph-cuts method. As shown in Figure 7.5 we want to recover a triple junction by filling in the circle region through minimizing the following energy,

$$\mathcal{E}_2(\Gamma) = \int_{\Omega} \lambda |u_{\Gamma} - u_{data}|^2 d\mathbf{x} + \alpha \oint_{\Gamma} ds \quad (7.2)$$

where the similarity parameter λ is 0 inside the circle region. The global minimizer of this problem should fill in the circle region with the minimum total length of the boundaries between the labels, i.e. the boundaries meet with the 120 degree angle in the center.

Figure 7.6 presents the solutions of the graph-cuts method with different neighborhood systems. Figure 7.6(a) gives the result using 4-neighborhood system. It is clear that the result is not the global minimizer in the continuous setting, but it is the correct solution under the give graph metrics. As the neighborhood size increases

(a) Noisy rectangle

(b) The result of the graph-cuts method

(c) The result of the level-set method

Figure 7.4: In figure (b) and (c), the first column contains the smoothed curve and the second column includes the mixture of the smoothed curves and noisy curves. Parameters in functional (4.3) are $\lambda = 0.05$ and $\beta = 8$ and 64 neighbor system is used

Figure 7.5: Look for triple junction

to 8, the result becomes quite different as shown in Figure 7.6(b). However, again this is not the correct solution for the problem. Continuing to increase neighborhood size to 16, the result goes like the one given in Figure 7.6(c). Figure 7.6(d) shows the result with 32-neighborhood system, which only has minute differences from the one in Figure 7.6(c). This suggests that as neighborhood size increases, the multi-phase graph-cuts method gives convergent result. In fact, the result using 64-neighborhood system is almost the same with the ones shown in Figure 7.6(c) and 7.6(d). Obviously, the junction produced by the graph-cuts method is far from the 120 degree triple junction, not even close to the center, albeit very large neighborhood has been used. This is mainly caused by the metrication error from anisotropic total variation approximation. Figure 7.7 gives the solutions of the GEWCVT method with different neighborhood size. Figure 7.7(a) gives the result using a disk neighborhood with radius $\omega = h$, where h is the mesh size. The result is not so good since the radius is too small to accurately approximate arc length. The result in Figure 7.7(b) is much better due to a larger neighborhood, $\omega = 2h$ in this case. However, It still can be observed that the junction shifts horizontally to right somehow. By increasing the radius of the neighborhood to $3h$, the result is improved further as presented in Figure 7.7(c). The junction is drawn back to the center in this case, although there are

Figure 7.6: Results of the Graph-cuts method for triple junction problem. (a)-(d) show the results with different neighborhoods: (a)4-neighborhood; (b)8-neighborhood; (c)16-neighborhood; (d)32-neighborhood.

Table 7.2: CPU Time for Triple Junction Problem

Neighborhood	4-nb	8-nb	16-nb	32-nb	64-nb
Graph-cuts	0.13s	0.3s	0.91s	2.4s	7.36s
Neighborhood	$\omega = h$	$\omega = 2h$	$\omega = 3h$	$\omega = 4h$	$\omega = 5h$
GEWCVT	0.32s	1.57s	1.8s	2.49s	2.61s

server pixels' downward shifting. When the the radius of the neighborhood becomes $4h$, this shifting is reduced to 2 pixels as presented in Figure 7.7(d). However, the larger neighborhood, e.g. $\omega = 5h$, will not change the result anymore, which implies that we already obtained a convergent solution. In contrast to the result presented in Figure 7.6(d), the result in Figure 7.7(d) is very close to the exact solution, which clearly illustrates that GEWCVT method has smaller model error than multi-phase graph-cuts method.

Table 7.2 records the CPU times for the triple junction problem. The image size used for previous examples is 200×200 . The first row lists the 5 different neighborhood systems used by multi-phase graph-cuts method. The second row gives the corresponding CPU times the graph-cuts method consumes. The third row shows the different neighborhood size GEWCVT method uses, where the statistics represent the radius of the neighborhood. The last row lists the corresponding CPU times the GEWCVT method takes. When the neighborhood is relative small, for example 4-neighborhood for graph-cuts and $\omega = h$ for GEWCVT, the graph-cuts performs much faster than the GEWCVT. However, as the size of the neighborhood increases, the GEWCVT becomes more efficient. For example, using 64-neighborhood the graph-cuts needs 7.36 seconds to solve the triple junction problem whereas GEWCVT method takes only 2.61 seconds with the neighborhood having radius $5h$. This is even more true if one takes the accuracy into account—the result produced by the GEWCVT method with $5h$ neighborhood is much more accurate than the one obtained by the graph-cuts method with 64-neighborhood. To conclude, the graph-cuts method is more adapted for the applications in which only small neighborhoods are used, and the GEWCVT method is more suitable for the applications where large neighborhoods are needed.

Finally, to conclude this chapter as well as the main part of the thesis, we use Table

Figure 7.7: Results of the GEWCVT method for triple junction problem. (a)-(d) show the results with the neighborhoods having different radius ω : (a) $\omega = h$; (b) $\omega = 2h$; (c) $\omega = 3h$; (d) $\omega = 4h$.

Table 7.3: Features of different methods

	Graph Cuts	Phase Field	GEWCVT	Adaptive Level Set	Level Set (Explicit)
Efficiency	Very Fast	Fast	Very Fast	Fast	Slow
Memory Request	High	High	Low	Low	Low
Model Error	Very Large	Medium	Large	Small	Small
Global Optimization	Yes	No	No	No	No

7.3 to summarize five different methods discussed in this thesis. The qualitative comparison is conducted from four aspects including efficiency, memory request, model error and whether the corresponding method is of global optimization. Generally, the graph-cuts is the fastest optimization technique among these methods, and it also has the largest memory requests as well as the biggest model error. Level-set method is the slowest method but it has the smallest model error and very low memory requests. The phase-field method is faster than the level-set method, and its model errors and memory requests are less than graph-cuts method. GEWCVT is normally faster than the phase-field method and even faster than the graph-cuts method in some applications and it consumes the least memory among these methods especially for multiphase problems.

Chapter 8

Future Works

Various examples have substantiated that appropriate data-dependent meshes could improve the efficacy and efficiency of the proposed methods in smoothing and reconstruction applications. Especially in 3D situation, adaptive meshes will gain much more efficiency than uniform meshes. Therefore, possible future works include extending adaptive curve smoothing method to surface smoothing. In particular, adaptive finite element solutions could overcome the limitation of the mesh resulting from the small transition layer in the phase field model. A posteriori error estimate based refinement strategy for PDE-based methods will be one of the future working directions, and appropriate metrics to measure the effectiveness of the adaptation procedure will be developed in the future as well. A feature preserving variational model will help preserve and present sharp features in smoothing and reconstruction applications. Integrating the CVT-based methods with Delaunay mesh generation techniques to obtain more efficient reconstruction methods will be investigated in the near future. A thorough study of feature preserving regularizations is an interesting subject for future investigations. Curvature related high order terms [47] and anisotropic diffusion [24, 98] method are currently under consideration. Moreover, a thorough study of the possible links between TV-regularization [84] and the corner detector used in this thesis is also of great interest. Finally, applying the technique developed in this thesis to other more specific reconstruction problems, such as reconstructing range

images, is also under investigation. Another direction of our future work is to develop a quantitative quality measure for surface reconstruction, surface smoothing, and surface enhancement. This measure should take into account the smoothness of surface, the rigid sharp features, and the deviation from the ground truth if available.

Bibliography

- [1] CGAL, *Computational Geometry Algorithms Library*. <http://www.cgal.org>.
- [2] U. ADAMY, J. GIESEN, AND M. JOHN, *New techniques for topologically correct surface reconstruction*, in IEEE Visualization, 2000, pp. 373–380.
- [3] M. ALEXA, *Wiener filtering of meshes*, in Proceedings of the Shape Modeling International, 2002, pp. 51–57.
- [4] M. ALEXA, J. BEHR, D. COHEN-OR, S. FLEISHMAN, D. LEVIN, AND C. SILVA, *Point set surfaces*, in IEEE Visualization, 2001, pp. 21–28.
- [5] N. AMENTA AND M. BERN, *Surface reconstruction by Voronoi filtering*, Discrete and Computational Geometry, 22 (1999), pp. 481–504.
- [6] N. AMENTA, M. BERN, AND M. KAMVYSSELIS, *A new Voronoi-based surface reconstruction algorithm*, in SIGGRAPH '98: Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques, 1998, pp. 415–421.
- [7] N. AMENTA, S. CHOI, T. DEY, AND N. LEEKHA, *A simple algorithm for homeomorphic surface reconstruction*, in Proceedings of the sixteenth annual symposium on Computational geometry, 2000, pp. 213–222.
- [8] E. BAE AND X.-C. TAI, *Graph cut optimization for the piecewise constant level set method applied to multiphase image segmentation*, Lecture Notes in Computer Science, 5567 (2009), pp. 1–13.

- [9] T. J. BARTH AND J. A. SETHIAN, *Numerical schemes for the hamilton-jacobi and level set equations on triangulated domains*, Journal of Computational Physics, 145 (1998), pp. 1–40.
- [10] J. BOISSONNAT, *Shape reconstruction from planar cross sections*, Computer Vision, Graphics, and Image Processing, 44 (1988), pp. 1–29.
- [11] J. BOISSONNAT AND F. CAZALS, *Smooth surface reconstruction via natural neighbour interpolation of distance functions*, in Proceedings of the sixteenth annual symposium on Computational geometry, 2000, pp. 223–232.
- [12] H. BOROUCAKI, P. GEORGE, AND S. LO, *Optimal Delaunay point insertion*, International Journal for Numerical Methods in Engineering, 39 (1996), pp. 3407–3437.
- [13] H. BOROUCAKI AND P. L. GEORGE, *Parametric surface meshing using a combined advancing-front generalized delaunay approach*, International Journal for Numerical Methods in Engineering, 49 (2000), pp. 233–259.
- [14] H. BOROUCAKI, F. HECHT, AND P. J. FREY, *Mesh gradation control*, International Journal for Numerical Methods in Engineering, 43 (1998), pp. 1143–1157.
- [15] H. BOROUCAKI AND S. LO, *Fast Delaunay triangulation in three dimensions*, Computer Methods in Applied Mechanics and Engineering, 128 (1995), pp. 153–167.
- [16] Y. BOYKOV AND M. JOLLY, *Interactive graph cuts for optimal boundary and region segmentation of objects in ND images*, in IEEE International Conference on Computer Vision, 2001, pp. 105–112.
- [17] Y. BOYKOV AND V. KOLMOGOROV, *Computing geodesic and minimal surfaces via graph cuts*, in IEEE International Conference on Computer Vision, 2003, pp. 26–33.

- [18] —, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 1124–1137.
- [19] Y. BOYKOV, O. VEKSLER, AND R. ZABIH, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239.
- [20] S. BRENNER AND L. SCOTT, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, 1996.
- [21] V. CASELLES, R. KIMMEL, AND G. SAPIRO, *Geodesic active contours*, International Journal of Computer Vision, 22 (1997), pp. 61–79.
- [22] J. CEBRAL AND R. LOHNER, *From medical images to anatomically accurate finite element grids*, International Journal for Numerical Methods in Engineering, 51 (2001), pp. 985–1008.
- [23] T. F. CHAN, S. ESEDOĞLU, AND M. NIKOLOVA, *Algorithms for finding global minimizers of image segmentation and denoising models*, SIAM Journal on Applied Mathematics, 66 (2006), pp. 1632–1648.
- [24] U. CLARENZ, U. DIEWALD, AND M. RUMPF, *Anisotropic geometric diffusion in surface processing*, in IEEE Visualization, 2000, pp. 397–405.
- [25] L. D. COHEN AND I. COHEN, *Finite element methods for active contour models and balloons for 2d and 3d images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 1132–1147.
- [26] L. D. COHEN AND R. KIMMEL, *Global minimum for active contour models : A minimal path approach*, International Journal of Computer Vision, 24 (1997), pp. 57–78.
- [27] G. CONG AND B. PARVIN, *Robust and efficient surface reconstruction from contours*, The Visual Computer, 17 (2001), pp. 199–208.

- [28] D. CREMERS AND S. SOATTO, *A pseudo-distance for shape priors in level set segmentation*, in 2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision, 2003, pp. 169–176.
- [29] B. CURLESS AND M. LEVOY, *A volumetric method for building complex models from range images*, in SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques, 1996, pp. 303–312.
- [30] M. DESBRUN, M. MEYER, P. SCHRÖDER, AND A. H. BARRA, *Implicit fairing of irregular meshes using diffusion and curvature flow*, in SIGGRAPH '99: Proceedings of the 26th annual conference on Computer Graphics and Interactive techniques, 1999, pp. 317–324.
- [31] T. K. DEY AND S. GOSWAMI, *Tight cocone: a water-tight surface reconstructor*, in SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications, 2003, pp. 127–134.
- [32] E. A. DINIC, *Algorithm for solution of a problem of maximum flow in networks with power estimation*, Soviet Mathematics Doklady, 11 (1970), pp. 1277–1280.
- [33] Q. DU, M. EMELIANENKO, AND L. JU, *Convergence of the lloyd algorithm for computing centroidal voronoi tessellations*, SIAM Journal on Numerical Analysis, 44 (2006), pp. 102–119.
- [34] Q. DU, V. FABER, AND M. GUNZBURGER, *Centroidal voronoi tessellations: Applications and algorithms*, SIAM Review, 41 (1999), pp. 637–676.
- [35] Q. DU, M. GUNZBURGER, AND L. JU, *Advances in studies and applications of centroidal voronoi tessellations*, Numerical Mathematics: Theory, Methods and Applications, 3 (2010), pp. 119–142.
- [36] Q. DU, M. GUNZBURGER, L. JU, AND X. WANG, *Voronoi tessellation algorithms for image compression and segmentation*, Journal of Mathematical Imaging and Vision, 24 (2006), pp. 177–194.

- [37] Q. DU, C. LIU, R. RYHAM, AND X. WANG, *Phase field modeling of the spontaneous curvature effect in cell membranes*, Communications on Pure and Applied Analysis, 4 (2005), pp. 537–548.
- [38] Q. DU, C. LIU, AND X. WANG, *A phase field approach in the numerical study of the elastic bending energy for vesicle membranes*, Journal of Computational Physics, 198 (2004), pp. 450–468.
- [39] Q. DU, C. LIU, AND X. WANG, *Retrieving topological information for phase field models*, SIAM Journal on Applied Mathematics, 65 (2005), pp. 1913–1932.
- [40] Q. DU, C. LIU, AND X. WANG, *Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions*, Journal of Computational Physics, 212 (2006), pp. 757–777.
- [41] Q. DU AND D. WANG, *Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations*, International Journal for Numerical Methods in Engineering, 56 (2002), pp. 1355–1373.
- [42] Q. DU AND D. WANG, *Boundary recovery for three dimensional conforming delaunay triangulation*, Computer Methods in Applied Mechanics and Engineering, 193 (2004), pp. 2547–2563.
- [43] Q. DU AND D. WANG, *Constrained boundary recovery for three dimensional Delaunay triangulations*, International Journal for Numerical Methods in Engineering, 61 (2004), pp. 1471–1500.
- [44] —, *Anisotropic centroidal Voronoi tessellations and their applications*, SIAM Journal on Scientific Computing, 26 (2005), pp. 737–761.
- [45] Q. DU AND L. ZHU, *Analysis of a mixed finite element method for phase field elastic bending energy of vesicle membrane deformation*, Journal of Computational Mathematics, 24 (2006), pp. 265–280.
- [46] H. EDELSBRUNNER AND E. MUCKE, *Three-dimensional alpha shapes*, in Proceedings of the 1992 workshop on Volume visualization, 1992, pp. 75–82.

- [47] M. ELSEY AND S. ESEDOĞLU, *Analogue of the total variation denoising model in the context of geometry processing*, tech. report, University of California Berkeley Department of Mathematics, July 2007.
- [48] L. C. EVANS, *Partial Differential Equations*, American Mathematical Society, 1998.
- [49] S. FLEISHMAN, I. DRORI, AND D. COHEN-OR, *Bilateral mesh denoising*, ACM Transactions on Graphics, 22 (2003), pp. 950–953.
- [50] L. FORD AND D. FULKERSON, *Flows in Networks*, Princeton University Press, 1962.
- [51] P. FREY AND P. GEORGE, *Mesh generation: application to finite elements*, ISTE, 2007.
- [52] P. J. FREY, *About surface remeshing*, in Proceedings of the 9-th International Meshing Roundtable, 2000, pp. 123–156.
- [53] P. J. FREY AND H. BOROUCHAKI, *Geometric surface mesh optimization*, Computing and Visualization in Science, 1 (1998), pp. 113–121.
- [54] P. J. FREY AND P.-L. GEORGE, *Mesh Generation : Application to Finite Elements (Second Edition)*, ISTE Publishing Company, 2008.
- [55] P. GEORGE AND H. BOROUCHAKI, *Delaunay triangulation and meshing: application to finite elements*, Kogan Page, 1998.
- [56] P. GEORGE, F. HECHT, AND E. SALTEL, *Automatic mesh generator with specified boundary*, Computer Methods in Applied Mechanics and Engineering, 92 (1991), pp. 269–288.
- [57] A. V. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum-flow problem*, Journal of the Association for Computing Machinery, 35 (1988), pp. 921–940.

- [58] D. GREIG, B. PORTEOUS, AND A. SEHEULT, *Exact maximum a posteriori estimation for binary images*, Journal of the Royal Statistical Society. Series B (Methodological), (1989), pp. 271–279.
- [59] H. HOPPE, T. DEROSE, T. DUCHAMP, J. McDONALD, AND W. STUETZLE, *Surface reconstruction from unorganized points*, in SIGGRAPH '92: Proceedings of the 19th annual conference on Computer Graphics and Interactive Techniques, 1992, pp. 71–78.
- [60] A. HORNUNG AND L. KOBELT, *Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding*, in IEEE conference on Computer Vision and Pattern Recognition, 2006, pp. 503–510.
- [61] —, *Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information*, in Geometry Processing 2006: Fourth Eurographics Symposium on Geometry Processing, 2006, pp. 41–50.
- [62] C. Q. HU AND C. W. SHU, *Weighted essentially non-oscillatory schemes on triangular meshes*, Journal of Computational Physics, 150 (1999), pp. 97–127.
- [63] T. KANUNGO, D. MOUNT, N. NETANYAHU, C. PIATKO, R. SILVERMAN, AND A. WU, *An efficient k-means clustering algorithm: Analysis and implementation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 881–892.
- [64] B. KARAMETE, M. BEALL, AND M. SHEPHARD, *Triangulation of arbitrary polyhedra to support automatic mesh generators*, International Journal for Numerical Methods in Engineering, 49 (2000), pp. 167–191.
- [65] R. KIMMEL, *Numerical Geometry of Images: Theory, Algorithms, and Applications*, Springer-Verlag, 2004.
- [66] N. KIRYATI AND A. M. BRUCKSTEIN, *Heteroscedastic hough transform (htht): An efficient method for robust line fitting in the ‘errors in the variables’ problem*, Computer Vision and Image Understanding, 78 (2000), pp. 69–83.

- [67] K. ITO, K. KUNISCH, AND Z. LI., *Level-set function approach to an inverse interface problem*, Inverse Problems, 17 (2001), pp. 1225–1242.
- [68] P. M. KNUPP, *Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i – a framework for surface mesh optimization*, International Journal for Numerical Methods in Engineering, 48 (2000), pp. 401–420.
- [69] L. KOBBELT, T. HESSE, H. PRAUTZSCH, AND K. SCHWEIZERHOF, *Iterative mesh generation for fe-computations on free form surfaces*, Engineering Computations, 14 (1997), pp. 806–820.
- [70] V. KOLMOGOROV AND Y. BOYKOV, *What metrics can be approximated by geo-cuts, or global optimization of length/area and flux*, in IEEE International Conference on Computer Vision, 2005, pp. 564–571.
- [71] V. KOLMOGOROV, R. ZABIH, AND S. GORTLER, *Generalized multi-camera scene reconstruction using graph cuts*, Lecture notes in computer science, 2683 (2003), pp. 501–516.
- [72] V. KOLMOGOROV AND R. ZABIN, *What energy functions can be minimized via graph cuts?*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 147–159.
- [73] N. KOMODAKIS, G. TZIRITAS, AND N. PARAGIOS, *Fast, approximately optimal solutions for single and dynamic MRFs*, in IEEE conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
- [74] W. KÜHNEL, *Differential Geometry: Curves - Surfaces - Manifolds, Second Edition*, American Mathematical Society, 2005.
- [75] V. LEMPITSKY AND Y. BOYKOV, *Global optimization for shape fitting*, in IEEE conference on Computer Vision and Pattern Recognition, 2007, pp. 17–22.

- [76] V. LEMPITSKY, C. ROTHER, AND A. BLAKE, *LogCut-efficient graph cut optimization for Markov random fields*, in IEEE International Conference on Computer Vision, 2007.
- [77] R. J. LEVEQUE, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, 2002.
- [78] S. LLOYD, *Least squares quantization in pcm*, IEEE Transactions on Information Theory, 28 (1982), pp. 129–137.
- [79] H. LOMBAERT, Y. SUN, L. GRADY, AND C. XU, *A multilevel banded graph cuts method for fast image segmentation*, in IEEE International Conference on Computer Vision, 2005, pp. 259–265.
- [80] W. LORENSEN AND H. CLINE, *Marching cubes: A high resolution 3d surface construction algorithm*, in SIGGRAPH '87: Proceedings of the 14th annual conference on Computer Graphics and Interactive Techniques, vol. 21, 1987, pp. 163–169.
- [81] B. MILNE., *Adaptive level set methods interfaces*, tech. report, University of California Berkeley Department of Mathematics, 1995.
- [82] Y. OHTAKE, A. BELYAEV, M. ALEXA, G. TURK, AND H. SEIDEL, *Multilevel partition of unity implicits*, ACM Transactions on Graphics, 22 (2003), pp. 463–470.
- [83] J. OLIVA, M. PERRIN, AND S. COQUILLART, *3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram*, Computer Graphics Forum, 15 (1996), pp. 397–408.
- [84] S. OSHER AND R. FEDKIW, *The Level Set Method and Dynamic Implicit Surfaces*, Springer-Berlin, 2002.
- [85] S. OSHER AND F. SANTOSA, *Level set methods for optimization problems involving geometry and constraints. i. frequencies of a two-density inhomogeneous drum*, Journal of Computational Physics, 171 (2001), pp. 272–288.

- [86] S. OSHER AND J. A. SETHIAN, *Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.
- [87] N. PARAGIOS, M. ROUSSON, AND V. RAMESH, *Matching distance functions: A shape-to-area variational approach for global-to-local registration*, Lecture Notes in Computer Science, 2351 (2002), pp. 775–789.
- [88] S. PARIS, F. SILLION, AND L. QUAN, *A surface reconstruction method using global graph cut optimization*, International Journal of Computer Vision, 66 (2006), pp. 141–161.
- [89] P. PERONA AND J. MALIK, *Scale space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 629–639.
- [90] J. QIAN, Y.-T. ZHANG, AND H.-K. ZHAO, *Fast sweeping methods for eikonal equations on triangular meshes*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 83–107.
- [91] L. I. RUDIN, S. OSHER, AND E. FATEMI, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 256–268.
- [92] S. RUSINKIEWICZ, O. HALL-HOLT, AND M. LEVOY, *Real-time 3d model acquisition*, ACM Transactions on Graphics, 21 (2002), pp. 438–446.
- [93] I. SAZONOV, D. WANG, O. HASSAN, K. MORGAN, AND N. WEATHERILL, *A stitching method for the generation of unstructured meshes for use with co-volume solution techniques*, Computer Methods in Applied Mechanics and Engineering, 195 (2006), pp. 1826–1845.
- [94] J. SHEWCHUK, *Constrained delaunay tetrahedralizations and provably good boundary recovery*, in Proceedings of the 11th International Meshing Roundtable, 2002, pp. 193–204.

- [95] M. SORMANN, C. ZACH, J. BAUER, K. KARNER, AND H. BISHOF, *Water-tight multi-view reconstruction based on volumetric graph-cuts*, Lecture Notes in Computer Science, 4522 (2007), pp. 393–402.
- [96] X.-C. TAI AND T. F. CHAN, *A survey on multiple level set methods with applications for identifying piecewise constant functions*, International Journal of Numerical Analysis and Modeling, 1 (2004), pp. 25–48.
- [97] X.-C. TAI AND C.-H. YAO, *Image segmentation by piecewise constant mumford-shah model without estimating the constants*, Journal of Computational Mathematics, 24 (2006), pp. 435–443.
- [98] T. TASDIZEN, R. WHITAKER, P. BURCHARD, AND S. OSHER, *Geometric surface smoothing via anisotropic diffusion of normals*, in IEEE Visualization, 2002, pp. 125–132.
- [99] G. TAUBIN, *A signal processing approach to fair surface design*, in SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer Graphics and Interactive Techniques, 1995, pp. 351–358.
- [100] J. THOMPSON, B. K. SONI, AND N. P. WEATHERILL, *Handbook of Grid Generation*, CRC Press, 1999.
- [101] H. K. VERSTEEG AND W. MALALASEKRA, *An Introduction to Computational Fluid Dynamics: The Finit Volume Method (Second Edition)*, Pearson Education Limited, 2007.
- [102] L. A. VESE AND T. F. CHAN, *A new multiphase level set framework for image segmentation via the mumford and shah model*, International Journal of Computer Vision, 50 (2002), pp. 271–293.
- [103] G. VOGIATZIS, P. TORR, AND R. CIPOLLA, *Multi-view stereo via volumetric graph-cuts*, in IEEE conference on Computer Vision and Pattern Recognition, 2005, pp. 391–398.

- [104] M. WAN, Y. WANG, AND D. WANG, *Variational surface reconstruction based on delaunay triangulation and graph cut*, International Journal for Numerical Methods in Engineering, to appear.
- [105] D. WANG, *A new method for efficient generation of high quality surface triangular meshes*, Communications in Computational Physics, 1 (2006), pp. 716–735.
- [106] D. WANG, O. HASSAN, K. MORGAN, AND N. WEATHERILL, *Efficient surface reconstruction from contours based on two-dimensional Delaunay triangulation*, International Journal for Numerical Methods in Engineering, 65 (2006), pp. 734–751.
- [107] D. WANG, O. HASSAN, K. MORGAN, AND N. P. WEATHERILL, *Eqsm: An efficient high quality surface grid generation method based on remeshing*, Computer Methods in Applied Mechanics and Engineering, 195 (2006), pp. 5621–5633.
- [108] D. WANG, O. HASSAN, N. P. WEATHERILL, AND K. MORGAN, *Enhanced remeshing from stl files and its application to surface grid generation*, Communications in Numerical Methods in Engineering, 23 (2007), pp. 227–239.
- [109] D. WANG AND X. WANG, *A three-dimensional adaptive method based on the iterative grid redistribution*, Journal of Computational Physics, 199 (2004), pp. 423–436.
- [110] J. WANG, L. JU, AND X. WANG, *An edge-weighted centroidal voronoi tessellation model for image segmentation*, IEEE Transactions on Image Processing, 18 (2009), pp. 1844–1858.
- [111] J. WANG AND X. WANG, *Edge-weighted centroidal voronoi tessellations*, Numerical Mathematics: Theory, Methods and Applications, 3 (2010), pp. 223–244.
- [112] X. WANG, *Phase field models and simulations of vesicle bio-membranes*, PhD thesis, Department of Mathematics, Penn State University, 2005.

- [113] Y. WANG, S. SONG, Z. TAN, AND D. WANG, *Adaptive variational curve smoothing based on level set method*, Journal of Computational Physics, 228 (2009), pp. 6333–6348.
- [114] N. WEATHERILL AND O. HASSAN, *Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints*, International Journal for Numerical Methods in Engineering, 37 (1994), pp. 2005–2039.
- [115] Z. WU AND J. SULLIVAN, *Multiple material marching cubes algorithm*, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 189–207.
- [116] N. XU, R. BANSAL, AND N. AHUJA, *Object segmentation using graph cuts based active contours*, Computer Vision and Image Understanding, 107 (2007), pp. 210–224.
- [117] C. ZACH, T. POCK, AND H. BISCHOF, *A globally optimal algorithm for robust tv - l^1 range image integration*, in IEEE International Conference on Computer Vision, 2007, pp. 14–21.
- [118] H. ZHAO, *A fast sweeping method for eikonal equations*, Mathematics of Computation, 74 (2005), pp. 603–627.
- [119] —, *A fast sweeping method for eikonal equations*, Mathematics of Computation, 74 (2005), pp. 603–627.
- [120] H. ZHAO, S. OSHER, AND R. FEDKIW, *Fast surface reconstruction using the level set method*, in Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), 2001, pp. 194–201.
- [121] H. ZHAO, S. OSHER, B. MERRIMAN, AND M. KANG, *Implicit and nonparametric shape reconstruction from unorganized points using variational level set method*, Computer Vision and Image Understanding, 80 (2000), pp. 295–319.
- [122] L. ZHU, Y. WANG, L. JU, AND D. WANG, *A variational phase field method for curve smoothing*, Journal of Computational Physics, 229 (2010), pp. 2390–2400.