Computers and Mathematics with Applications 64 (2012) 2663–2681

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Generalized edge-weighted centroidal Voronoi tessellations for geometry processing☆

Yu Wang [a], Lili Ju [b,*], Desheng Wang [c], Xiaoqiang Wang [d]

[a] CGGVeritas Hub, 9 Serangoon North Ave 5, Singapore 554531, Singapore

[b] Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

[c] Division of Mathematical Sciences, School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore

[d] Department of Mathematics, Florida State University, Tallahassee, FL 32306, USA

ARTICLE INFO

Article history:
Received 5 March 2012
Received in revised form 9 July 2012
Accepted 25 July 2012

Keywords:
Edge-weighted centroidal Voronoi tessellations
Surface smoothing
Surface reconstruction
Feature preserving

ABSTRACT

In this paper, we propose a generalized edge-weighted centroidal Voronoi tessellation (GEWCVT) model and corresponding solution algorithms, then apply them for geometry processing such as curve/surface smoothing and reconstruction. The main idea of the method is to seek a good way to discretize the similarity and regularity measures of the objective functional in the context of centroidal Voronoi tessellation methodology, so that its minimization can be done by clustering-type algorithms. Through various numerical examples, the proposed GEWCVT-based method is shown to be an effective and robust tool for such applications.

1. Introduction

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

$$\mathcal{E}(\Gamma) = \int_{\Omega} \lambda |u_\Gamma - u_{data}|^2 \, d\mathbf{x} + \oint_{\Gamma} (\alpha + \beta \, \mathrm{dist}_{\Gamma_{data}}) \, ds \tag{1.1}$$

where Ω is the computing domain (occupied by the data), Γ is the interface (curve or surface) to be smoothed or reconstructed, $\mathrm{dist}_{\Gamma_{data}}$ 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, and α, β and λ in (1.1) are tuning parameters to be selected (could be constants or functions). The first term in the right hand side measures the similarity between the solution Γ and the input data, and the second term measures the regularity of Γ. In this paper this unified variational framework will be expressed in the centroidal Voronoi tessellation (CVT) languages as that in [1]. Basically in this approach, assuming the function u_Γ is a labeling function, a special distance between a point and a label/cluster which takes account of geometric information of clusters' boundaries needs to be defined; with this

☆ This work is partially supported by the US National Science Foundation under grant number DMS-0913491 and by the Singapore Ministry of Education under grant numbers ARC 29/07 T207B2202 and RG 59/08 M5211092 and the Singapore National Research Foundation under grant number 2007IDM-IDM 002-010 and in part at the Technion by a fellowship of the Israel Council for Higher Education.

* Corresponding author.
E-mail addresses: flimanadam@gmail.com (Y. Wang), ju@math.sc.edu (L. Ju), desheng@ntu.edu.sg (D. Wang), wwang3@fsu.edu (X. Wang).

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.07.011

distance function we then are able to generalize the notions of Voronoi diagrams and apply the Lloyd iteration to minimize the objective functional.

Currently popular methods used to solve the above minimization problem (1.1) include PDE-based methods [2–5] and graph-based methods [6–8]. The differences between these two approaches are mainly in two aspects. The first main difference is that the PDE-based methods use continuous functions to represent interfaces whereas the graph-based methods use discrete functions, the other one is that the PDE-based methods basically solve the Euler–Lagrangian equations derived from the proposed variational formulations, which usually gives local minimizers unless the functional is convex, while the graph-based methods always seek global minimizers in a certain finite space as long as the functional is graph-representable. Thus, graph-based methods are often much more efficient and robust while PDE-based methods are sometimes more general and accurate. The accuracy issues can be explained in two ways: first, PDE-base methods compute the interface by interpolating the function value (e.g., zero-level-set) and therefore normally have sub-pixel accuracy, on the other hand graph-based methods often only give partitions of the grids and sub-pixel accuracy is not so meaningful in this case; second, in the process of approximating a continuous model, graph-based methods introduce some model errors (see [9] for more details), and the errors of PDE-based methods basically come from the numerical solution not modeling itself. Due to more and more extensive research 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 in wide use. The interested reader can refer to [10] and the references cited therein. The basic k-means algorithm can be viewed as a special case of the centroidal Voronoi tessellation (CVT) methodology [11]. As a versatile methodology, the model of CVT [12,13] has been introduced to numerous fields and applications, such as image processing, data analysis and quantization, computational geometry, sensor networks, numerical partial differential equations [12,11], and so on. Recently, in [1], 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, an aspect which is analogous to PDE-based methods. EWCVTs for continuous domains in $\mathbb{R}^2$ are discussed in [14]. EWCVT-based algorithms are essentially clustering algorithms like k-means [10] 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-cut methods, the EWCVT-based methods converge to local minima but often require much less memory, especially for multi-phase cases. As a matter of fact, EWCVT-based methods provide a new way to approximate the well-known Mumford–Shah model [15].

The objective of this paper is to develop a generalized EWCVT (GEWCVT) model and corresponding algorithms, and apply them to geometry processing problems such as curve/surface smoothing and reconstruction. The first term (fitting) in the functional (1.1) can be naturally approximated by a discrete clustering energy functional, and the key part is how to interpret the second term (regularity) in the CVT context. Following the formation of the EWCVT model proposed in [1], 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 edge points and multiplying by some associated weight (constant or nonconstant) within the neighborhood. A generalized edge-weighted distance then can be derived for the discretized functional and, consequently, generalized edge-weighted Voronoi regions can be defined. With this new edge-weighted distance, we then develop a GEWCVT model and corresponding algorithms for its construction. It is worthwhile pointing out that when the edged-weighted function is constant the GEWCVT degenerates to the EWCVT. In other words, EWCVT is a special case of the GEWCVT and without ambiguity they are all called GEWCVT hereafter.

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

2. A discrete model and energy for geometry processing

As discussed in [16,17], the basic variational models for geometry processing problems will often have the following form:

$$\Gamma = \text{argmin}(\textit{similarity}(\Gamma, \text{data}) + \textit{regularity}(\Gamma)) \tag{2.1}$$

where "data" denotes the given data, which could be curves, surfaces or a point 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 (2.1): find Γ that minimizes the objective functional

$$\mathcal{F}(\Gamma) = \int_{\Omega} \lambda |u_{\Gamma} - u_{data}|^2 \, d\mathbf{x} + \oint_{\Gamma} r \, ds. \tag{2.2}$$

The first term in the right hand side of (2.2) 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 the whole

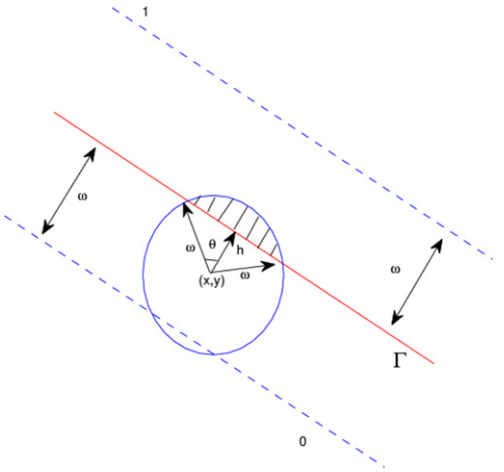

Fig. 2.1. Shadow area is the set of local edge points associated with (x, y).

domain Ω related to u_Γ. The first term in the right hand side of (2.2) can be easily expressed by the CVT language such as

$$\sum_{\mathbf{x}\in D} \lambda(\mathbf{x})|u(\mathbf{x}) - u_{data}(\mathbf{x})|^2, \tag{2.3}$$

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 are points within its neighborhood belonging to a different cluster. Let us define a local characteristic function as

$$\chi_{(x,y)}(x', y') = \begin{cases} 1 & \text{if } u_\Gamma(x', y') \neq u_\Gamma(x, y) \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

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 (see Fig. 2.1). Then

$$l(x, y) = \int_{N_\omega} \chi_{(x,y)}(x', y')\, dx'dy', \tag{2.5}$$

denotes the area inside the shadowed region in Fig. 2.1. We can show that

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

with the proof given in the following.

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

$$l(x, y) = \omega^2\theta - \omega^2 \sin\theta \cos\theta\,, \tag{2.6}$$

where θ is the angle shown in Fig. 2.1. 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

$$\begin{aligned}
\lim_{\omega\to 0} \frac{3}{4\omega^3} \int_\Omega rl\, dxdy &= \lim_{\omega\to 0} \frac{3}{2\omega^3} \int_0^\omega \oint_\Gamma r(s, h)(\omega^2\theta - \omega^2 \sin\theta\cos\theta)\, dsdh \\
&= \lim_{\omega\to 0} \frac{3}{2\omega} \int_0^{\pi/2} \oint_\Gamma r(s, \omega\cos\theta)(\theta - \sin\theta\cos\theta)\omega\sin\theta\, dsd\theta \\
&= \lim_{\omega\to 0} \frac{3}{2} \int_0^{\pi/2} \oint_\Gamma r(s, \omega\cos\theta)(\theta - \sin\theta\cos\theta)\sin\theta\, dsd\theta \\
&= \frac{3}{2} \int_0^{\pi/2} \oint_\Gamma \lim_{\omega\to 0} r(s, \omega\cos\theta)(\theta - \sin\theta\cos\theta)\sin\theta\, dsd\theta
\end{aligned}$$

$$= \frac{3}{2}\int_0^{\pi/2} \oint_{\Gamma} r(s,0)(\theta - \sin\theta\cos\theta)\sin\theta \; ds d\theta$$
$$= \frac{3}{2}\int_0^{\pi/2} (\theta - \sin\theta\cos\theta)\sin\theta \; d\theta \oint_{\Gamma} r(s,0)\; ds$$
$$= \oint_{\Gamma} r \; ds. \tag{2.7}$$

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 but the constant will be different from $\frac{3}{4\omega^3}$. The choice of neighborhood system, especially the parameter ω, depends on the application (see the remark in Section 5).

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

$$\hat{\mathcal{F}}(u) = \sum_{(x,y)\in D} \lambda(x,y)|u(x,y) - u_{data}(x,y)|^2 + \frac{1}{\omega^3}\left[\sum_{(x,y)\in D} r(x,y)l(x,y)\right]. \tag{2.8}$$

The second term in the right hand of (2.8) is called the *generalized edge energy*, thus we name the functional (2.8) 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 objective functional:

$$\hat{\mathcal{F}}(u) = \sum_{\mathbf{x}\in D} \lambda(\mathbf{x})|u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 + \frac{1}{\omega^{d+1}}\left[\sum_{\mathbf{x}\in D} r(\mathbf{x})l(\mathbf{x})\right]. \tag{2.9}$$

This is a generalization of the discrete functional developed in [1] in which $\lambda(\mathbf{x})$ does not appear and $r(\mathbf{x})$ is just a constant weighting function. A CVT-based minimization procedure for this functional will be derived in the next section.

3. Generalized edge-weighted centroidal Voronoi tessellations

The aim of this section is to build the generalized EWCVT model for the discrete functional (2.9) and develop fast methods for its minimization based on the CVT methodology studied in [12,1]. By computing the variation of the functional, the corresponding Euler–Lagrangian equation of (2.9) 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 (2.9) is merely a labeling function and can be obtained by classical clustering algorithms which are essentially integer programming problems.

3.1. Generalized edge-weighted Voronoi tessellation and its construction

The problem is to determine the variation of the generalized edge-weighted clustering energy (2.9) 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}. The energy (2.9) can be rewritten as

$$\hat{\mathcal{F}}(u) = \left(\sum_{\mathbf{x}'\in D, \mathbf{x}'\neq\mathbf{x}} \lambda(\mathbf{x}')|u(\mathbf{x}') - u_{data}(\mathbf{x}')|^2\right) + \lambda(\mathbf{x})|u(\mathbf{x}) - u_{data}(\mathbf{x})|^2$$
$$+ \frac{1}{\omega^{d+1}}\left[\left(\sum_{\mathbf{x}'\in D, \mathbf{x}'\neq\mathbf{x}} r(\mathbf{x}')l(\mathbf{x}')\right) + r(\mathbf{x})l(\mathbf{x})\right]. \tag{3.1}$$

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

$$\lambda(\mathbf{x})|c_{t_1} - u_{data}(\mathbf{x})|^2 - \lambda(\mathbf{x})|c_{t_2} - u_{data}(\mathbf{x})|^2. \tag{3.2}$$

The variation of edge energy (ignoring the scaler $\frac{1}{\omega^{d+1}}$ for a moment) for $\mathbf{x}$ caused by such change of the label is

$$r(\mathbf{x})[n_{t_2}(\mathbf{x}) - n_{t_1}(\mathbf{x})] \tag{3.3}$$

where n_t is the size of the set

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

Eq. (3.3) is the amount of the change of the size of neighborhood of $\mathbf{x}$, and can be also written as

$$\sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_2}} r(\mathbf{x}) - \sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_1}} r(\mathbf{x}), \tag{3.4}$$

since $n_{t_1} = \sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_1}} 1$ and $n_{t_2} = \sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_2}} 1$. The variation of edge energy (without the weight $\frac{1}{\omega^{d+1}}$) for $\mathbf{x}'$ is

$$\sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_2}} r(\mathbf{x}') - \sum_{\mathbf{x}'\in A_{\mathbf{x}}^{t_1}} r(\mathbf{x}'). \tag{3.5}$$

Summing (3.2), (3.4), (3.5), the variation of the total energy $\hat{\mathcal{F}}(u)$ due to transferring the point $\mathbf{x}$ from the label c_{t_1} to c_{t_2} is given by

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

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

$$\mathrm{dist}^*(\mathbf{x}, c_s) = \sqrt{\lambda(\mathbf{x})|u_{data}(\mathbf{x}) - c_s|^2 + \frac{1}{\omega^{d+1}} \sum_{\mathbf{x}'\in \mathcal{N}_\omega(\mathbf{x})\setminus A_{\mathbf{x}}^s} [r(\mathbf{x}') + r(\mathbf{x})]}. \tag{3.6}$$

We would particularly like to remark that this distance (3.6) is not defined in the usual mathematics sense, it is called a distance just because we follow the conventions used in CVT-based methods [1,14]. Note that $r(\mathbf{x})$ is non-negative to ensure that the above distance function is always meaningful, and when $r(\mathbf{x})$ is constant Eq. (3.6) naturally reduces to the change of the *edge-weighted distance* proposed in [1]. 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 (3.6), $\mathcal{D} = \{D_s\}_{s=1}^L$ of D can be defined as

$$D_s = \{\mathbf{x} \in D \mid \mathrm{dist}^*(\mathbf{x}, c_s) < \mathrm{dist}^*(\mathbf{x}, c_t),\ t = 1, \ldots, L,\ t \neq s\}. \tag{3.7}$$

We call such a set of $\{D_s\}_{s=1}^L$ a *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 calculation is given below:

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

1. For every $(x, y) \in D$,
   (a) calculate and compare the generalized edge-weighted distance defined in (3.6) 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 $\{\tilde{D}_s\}_{s=1}^L$.
2. If no point is moved, return $\{\tilde{D}_s\}_{s=1}^L$ and exit; otherwise go to Step 1.

Since Algorithm 1-GEWVT strictly decreases the objective functional $\hat{\mathcal{F}}(u)$ along the iterations and the (discrete) input data is 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 compared with the number of points in the whole domain. Such a strategy dramatically improves the efficiency of 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 $\{\tilde{D}_s\}_{s=1}^L$ 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 (3.6) 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 $\{\tilde{D}_s\}_{s=1}^L$.
3. If no point is moved, return $\{\tilde{D}_s\}_{s=1}^L$ and exit; otherwise go to Step 1.

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

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 (2.9) 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

$$u(\mathbf{x}) = c_s \quad \text{for some } s \text{ such that } \mathbf{x} \in D_s\,. \tag{3.8}$$

Then, seeking a minimizer of $\hat{\mathcal{F}}(u)$ is equivalent to finding 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 centroids of the Voronoi regions, which can be computed by

$$\tilde{c}_s = \frac{\sum\limits_{\mathbf{x}\in D_s} \lambda(\mathbf{x})u(\mathbf{x})}{\sum\limits_{\mathbf{x}\in D_s} \lambda(\mathbf{x})}, \quad i = 1, \ldots, L. \tag{3.9}$$

Based on the CVT methodology [12], the process of seeking minimizers of (2.9) is to find the generators $\mathcal{C} = \{c_s\}_{s=1}^L$ and the associated GEWVT $\mathcal{D} = \{D_s\}_{s=1}^L$ such that

$$c_s = \tilde{c}_s, \quad s = 1, \ldots, L. \tag{3.10}$$

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 [1].

Analogous to the classical CVT approach, the minimization is achieved by *Lloyd iteration* [27,28] – 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 $\tilde{D}_s$, $s = 1, \ldots, L$, determine its centroid $\tilde{c}_s$ by (3.9).
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 $s = 1, \ldots, L$ 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.

In Algorithm 3-GEWCVT, although the point transferring between clusters occurs immediately after the possible reduction 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 the efficiency of the GEWCVT-based algorithms in many applications [11,1].

4. Applications to geometry processing

In the last two sections we presented a GEWCVT-based approach for discretizing and minimizing the functional of type (2.2). 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.

4.1. Data representation

These two problems are closely related and can be addressed in the following discrete framework (letting $r(\mathbf{x}) = \alpha(\mathbf{x}) + \beta(\mathbf{x})\,\mathrm{dist}_{\Gamma_{data}}(\mathbf{x})$):

$$\hat{\mathcal{F}}(u) = \sum_{\mathbf{x}\in D} \lambda_{data}(\mathbf{x})|u(\mathbf{x}) - u_{data}(\mathbf{x})|^2 + \frac{1}{\omega^{d+1}}\left[\sum_{\mathbf{x}\in D}(\alpha(\mathbf{x}) + \beta(\mathbf{x})\,\mathrm{dist}_{\Gamma_{data}}(\mathbf{x}))l(\mathbf{x})\right] \tag{4.1}$$

where Γ is the curve (or surface) we seek and Γ_{data} the given data, which 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 the application. The unsigned distance function from a point $\mathbf{x}$ to the data set Γ_{data} is defined as usual, i.e.,

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

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, and in particular they can be *input-data-dependent*. In fact, if we treat u_Γ as a labeling function and let $r(x) = \alpha_{data} + \beta\,\mathrm{dist}_{\Gamma_{data}}$, then the above functional has exactly the same form as the functional (2.2). Thus, the curve/surface smoothing and reconstruction problems are optimization problems in this sense.

The curve/surface Γ is implicitly represented by the discontinuities of u_Γ. In our approach, the discrete domain D will be divided into several clusters by Γ and each cluster shares a unique value of u_Γ, which is called "label" of that cluster. Therefore it is also obviously different from the traditional level set method [18,19] or phase-field method [20,21,17]. When the data is 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 intersect the polygon (or polyhedron) boundary an odd number of times, otherwise an even number of 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. The interior region is composed of the points with odd crossings and the exterior region the points with even ray crossings. It is natural to give the interior subdomain a label and the exterior subdomain another label. When applied to multi-subdomain problems, the principles are essential similar.

When the data is 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 a two-cluster problem. First, suppose the unsigned distance function is available, then a fuzzy region is defined as

$$\Omega_{Fuzzy} = \{\mathbf{x} : \mathrm{dist}_{\Gamma_{data}}(\mathbf{x}) \le d_{th}\} \tag{4.2}$$

where the thickness of the fuzzy region d_{th} may depend on the noisy level and data resolution. If the data is dense and within a reasonably noise level, the Ω_{Fuzzy} will separate the domain into at least two subdomains. After the fuzzy region is established, the whole domain has a partitioning: inside, outside and fuzzy regions. Then we can give labels for the regions, i.e., the value of u_{data}. We also need to determine the topological information of the points in the fuzzy 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 (4.1) and the Ω_{Fuzzy} construction. 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 the fast sweeping method [22] 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 the selection of parameter functions λ_{data}, α, and β. Generally, α and β 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.

4.2. Determination of λ_{data}

4.2.1. 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 is 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 depending on noise level.

Recently, a better algorithm for determining λ_{data} (for two-phase problems) was proposed in [17] 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 $\lambda_{data}(\mathbf{x})$ should be large. Therefore a data-dependent (nonconstant) parameter function $\lambda_{data}(\mathbf{x})$ can be used to improve smoothing results. So far as the features such as corners and edges are concerned, a quantity that can indicate singularities of the noisy data is needed. To this purpose, we generalize the idea developed in [17] 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 noise 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, $\Gamma \cap S(\mathbf{x})$, 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 $|S(\mathbf{x})|$ respectively. The ratio defined by

$$\tau_r(\mathbf{x}) = \frac{\min(|A|, |B|)}{|S(\mathbf{x})|} \tag{4.3}$$

can be viewed as the singularity indicator. It is obvious that $0 \le \tau_r(x) \le \frac{1}{2}$, and $\tau_r(\mathbf{x})$ attains the maximum $\frac{1}{2}$ only when $\Gamma \cap S(\mathbf{x})$ is straight and passes $\mathbf{x}$. The smaller $\tau_r(\mathbf{x})$ 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 the noise. 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 $S(\mathbf{x})$ is divided into k parts $A_1, A_2, \ldots, A_k$. The definition of $\tau_r(\mathbf{x})$ becomes

$$\tau_r(\mathbf{x}) = \frac{\min_i(|A_i|)}{|S(\mathbf{x})|}. \tag{4.4}$$

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 $\mathcal{N}(\mathbf{x}, r)$, where N_i denotes the number of nodes with label i in $\mathcal{N}(\mathbf{x}, r)$. Set $N_{min} = \min\{N_1, N_2, \ldots, N_k\}$.
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

$$\lambda_{data}(\mathbf{x}) = \lambda_1 + \lambda_2(1 - 2\tau_r(\mathbf{x})) \tag{4.5}$$

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.

4.2.2. 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 [23,5,24] is commonly used for reconstruction, in which the distance function acts as a potential field attracting the initial surface to the local minimizers of

$$E_{GAC}(\Gamma) = \int_\Gamma \text{dist}_{\Gamma_{data}}\, ds. \tag{4.6}$$

It is well-known that global minimizers of the functional above are isolated points and its meaningful solutions correspond to the local minimizers [23]. This fact also implies that the final solution is sensitive to initial guesses. Therefore, using global optimization methods such as the graph-cut methods to minimize this functional needs some special techniques [25]. As pointed out in [26,8], the narrow-banded implementation is equivalent to minimizing the functional (2.2) with a non-constant function λ_{data} such as

$$\lambda_{data}(\mathbf{x}) = \begin{cases} C & \mathbf{x} \notin \Omega_{Fuzzy} \\ 0 & \mathbf{x} \in \Omega_{Fuzzy} \end{cases} \tag{4.7}$$

where Ω_{Fuzzy} is defined in (4.2), and C is a large constant. Here the width of the fuzzy region is the same as the one in the computation of the labeling function. Such a 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 fuzzy region discontinuity of the labeling function is forced to align with the data set. It has been numerically verified in [8] that by choosing an appropriate band width d_{th}, the computed minimizers of the functional (4.1) always give good results.

5. Numerical experiments

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 3.2 GHz and 4 GB memory. The first subsection presents results of curve/surface smoothing in the discrete setting, and the second subsection gives examples of curve/surface reconstruction. Since uniform meshes of the domain are used, the outputs of our GEWCVT-based algorithm are merely partitions of the mesh nodes. We also note that discretization artifacts are inevitable, especially for three-dimensional problems, thus the surface examples presented here are all post-processed by a Laplacian smoothing filter to effectively eliminate the discretization artifacts. "Marching Cube" [29] is used to extract surfaces from the partitioned voxel data.

The examples are presented in two categories: smoothing and reconstruction. For both categories, examples start with two-phase problems, i.e. $L = 2$ for a partition $\{D_s\}_{s=1}^L$, then some $L > 2$ examples are presented including 2-D and 3-D cases. To emphasize, two-phase and multi-phase problems are treated with no difference by the proposed algorithms, and in fact

the phase number is just a user-defined parameter in our code. In our experiments, we take $\mathcal{N}_\omega$ to be a circle centered at point (x, y) with radius ω. As to the stopping criteria the following two conditions are used: the first one is

$$\frac{P_{tran}}{P_{tot}} \leq c_1 \tag{5.1}$$

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

$$\frac{|E^{n+1} - E^n|}{E^{n+1}} \leq c_2 \tag{5.2}$$

where E^{n+1} denotes the value of the functional in the current step and E^n denotes the value of the functional in the previous step. In our experiments, we set $c_1 = 2.0\text{e} - 5$ and $c_2 = 1.0\text{e} - 4$. Once either of the above conditions is satisfied, our GEWCVT-based algorithm will be terminated. (In other words, if the change of the value of the functional 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 2 is taken to be a square centered at point (x, y) with side length $\omega_1 = 2\omega$. Based on our experiments, ω can be very small, for example just a few times the mesh size. This is partially because in the application of geometry processing, the initial guess can 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. Based on our experiences, ω typically ranges from $3h$ to $6h$, and a larger neighborhood does not produce significant visible differences and a too small neighborhood can not approximate the arc-length (surface area) of the boundaries very well [30].

5.1. Smoothing examples

For smoothing applications, the selection of parameters λ_{data}, α, β often depends on the noise level. A typical choice is to set $\beta = 0$ and $\lambda_{data} = 1.0$, then α can be computed according to the given noise level [31]. However, for a real application whose noise level is unknown, the automatic selection of regularization parameter is an issue requiring special attention. Efficiently choosing the parameter is still an active and interesting research topic beyond the scope of this paper. For a more detailed discussion, the interested reader can refer to [32–34] and references therein. Roughly speaking, a too small α results in insufficient noise removal and a too large α may yield a trivial constant solution. In [1,30] a simple way is provided to roughly determine the parameter α (again assume $\beta = 0$ and $\lambda_{data} = 1.0$): compute $\hat{\mathcal{F}}_1 = \left(\sum_{\mathbf{x}\in D} |\tilde{u}(\mathbf{x}) - u_{data}(\mathbf{x})|^2\right)$ and $\hat{\mathcal{F}}_2 = \frac{1}{\omega^{d+1}}\left[\sum_{\mathbf{x}\in D}(\alpha) l(\mathbf{x})\right]$, and then α is chosen such that

$$\frac{\max(\hat{\mathcal{F}}_1, \hat{\mathcal{F}}_2)}{\min(\hat{\mathcal{F}}_1, \hat{\mathcal{F}}_2)} = M \leq 10$$

is satisfied. In other words, we want to prevent either a similarity term or regularity term from dominating. This procedure can be automatically incorporated into the Lloyd iteration, as proposed in [30] for any given M. We also note that in order to preserve features, the ratio $\frac{\lambda_2}{\lambda_1}$ should be large (at least 1000). In our experiments λ_1 is fixed to be 1, and λ_2 is multiplied by 10 each time until the resulting objective function does not decrease. A more automatic way to determine the parameters λ_1 and λ_2 needs further investigations in the future.

In the following 2D examples the computation domain is $[-1, 1] \times [-1, 1]$ unless otherwise specified. The first two smoothing examples are two-phase problems, see Fig. 5.1 which presents smoothing of a noisy cat curve and Fig. 5.2 which shows smoothing of a noisy Air Jordan logo. The noisy curves are obtained from segmentation. Due to low resolution, the boundaries of the segmentation results have a lot of zig-zags and some outliers, which can be observed from Fig. 5.1(a) and Fig. 5.2(a). For both 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 both examples. We would like to show the effect of the parameters α and β through these two examples. For the cat example, we fix the parameter $\alpha = 0.03$. Fig. 5.1(b) presents the smoothing result obtained by our algorithm with $\beta = 9.0$ and Fig. 5.1(c) does that with $\beta = 0.0$. It is obvious that smaller β produces a smoother result, since the curve closer to the noisy curve results in smaller cost for larger β. For the Air Jordan logo example, we fix the parameter $\beta = 0$. The smoothing results with $\alpha = 0.02$ and $\alpha = 0.04$ are given in Fig. 5.2(b) and (c) respectively. It can be easily observed from this example that as α increases, the result becomes smoother. This is because α controls the regularization effect and larger α means curves with shorter length are expected. Overall, under the specified resolution all zig-zags are effectively eliminated for both examples.

Fig. 5.3 illustrates a multi-phase example. In this example, the target curves separate the plane into 4 disconnected domains; see Fig. 5.3(a), which presents the 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. Fig. 5.3(b) and (c) show the smoothing results with $\omega = 4h$ and $\omega = 5h$, respectively. It is easy to see from

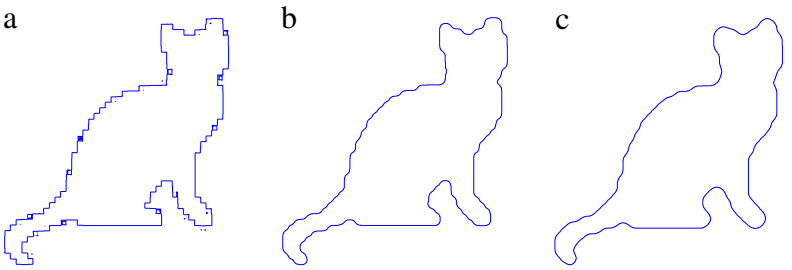

Fig. 5.1. Smoothing of a noisy cat by our GEWCVT-based algorithm with $\omega = 4h$, $\lambda_{data} = 1.0$ and $\alpha = 0.03$. (a) The noisy curve; (b) the smoothed curve using $\beta = 9.0$ (CPU time = 1.16 s); (c) the smoothed curve using $\beta = 0$ (CPU time = 2.27 s).

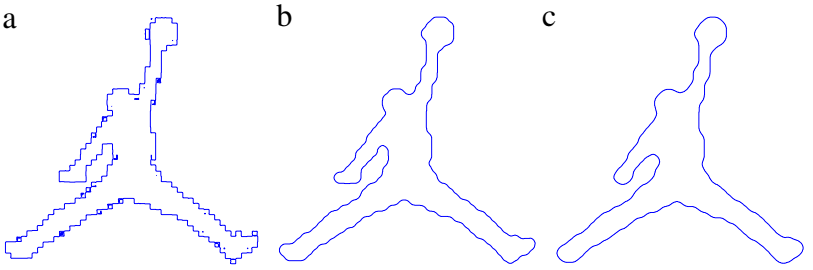

Fig. 5.2. Smoothing of a noisy Air Jordan logo by our GEWCVT-based algorithm with $\omega = 4h$, $\lambda_{data} = 1.0$, and $\beta = 0$. (a) The noisy curve; (b) the smoothed curve using $\alpha = 0.02$ (CPU time = 0.93 s); (c) the smoothed curve using $\alpha = 0.04$ (CPU time = 1.42 s).

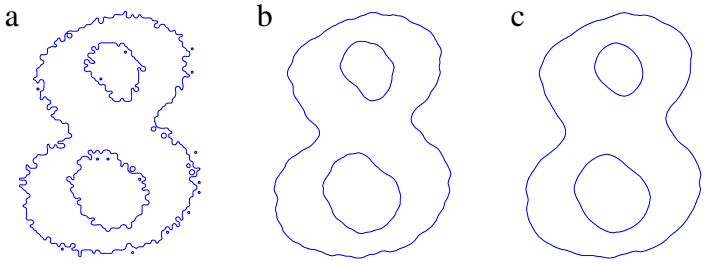

Fig. 5.3. Smoothing of a noisy number eight by our GEWCVT-based algorithm with $\lambda_{data} = 1.0$, $\alpha = 0.06$ and $\beta = 0$. (a) The noisy curves; (b) the smoothed curve using $\omega = 4h$ (CPU time = 1.2 s); (c) the smoothed curve using $\omega = 5h$ (CPU time = 2.5 s).

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 be 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.

Fig. 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 grid is used. The initial noisy curves (see Fig. 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 s to obtain the result as shown in Fig. 5.4(b), from which it can be seen that noise is removed and the resulting curves are smooth indeed. However, the intersections of the curves are shifted a lot, and the resulting curves have quite different topologies from the original ones. In order to preserve these intersections, we then change to a data-dependent similarity parameter function λ_{data}. 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 (4.5) with $\lambda_1 = 1.0$, $\lambda_2 = 1000.0$. Fig. 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 a total of 15.3 s including 14.47 s used to compute $\lambda_{data}(\mathbf{x})$. Along with intersections, the noise near the intersected region is kept as well, in some situation this phenomenon is undesirable. To tackle this problem, the resulting curve in Fig. 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., Fig. 5.4(c)) has much less noise. Fig. 5.4(d) presents the final smoothing result by our algorithm with $\alpha = 0.01$, where the noise in the intersected regions is also eliminated and the intersections are satisfactorily preserved.

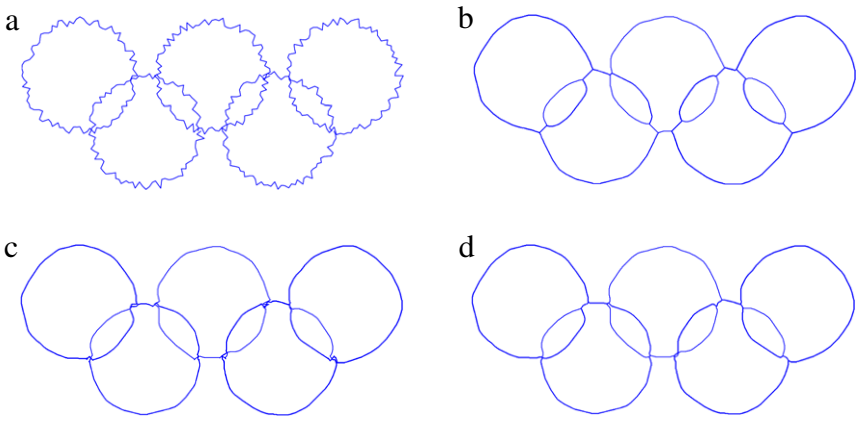

Fig. 5.4. Smoothing of curves formed by five noisy circles by our GEWCVT-based algorithm with $\omega = 4h$ and $\beta = 1.0$. (a) The noisy curves; (b) the smoothed curve using $\lambda_{data} = 1.0$ and $\alpha = 0.05$ (CPU time = 1.82 s). (c) the smoothed curve using $\alpha = 0.05$ and a nonconstant $\lambda_{data}(\mathbf{x})$ defined in (4.5) with $\lambda_1 = 1.0$, $\lambda_2 = 1000.0$ (CPU time = 15.3 s). (d) the final smoothed curve after one more processing using $\lambda_{data} = 1.0$ and $\alpha = 0.01$.

Fig. 5.5. Look for triple junction.

The next example shows that the GEWCVT model indeed approximates the arc-length very well, in particular, the difference between the multi-phase graph-cuts method and the GEWCVT-based method is manifested through this example. Although both methods work for discrete valued functions and inevitably introduce some model errors for continuous problems, the magnitude of model errors of the GEWCVT-based method is much smaller. The graph-cuts method often has serious metric errors [9,35] caused by using anisotropic total variation approximation and neighborhood systems. On the other hand, the GEWCVT method introduces errors when approximating the curves by straight lines just in a small neighborhood. The triple junction example will qualitatively demonstrate this issue. As shown in Fig. 5.5, we want to recover a triple junction by filling in the circle region by minimizing the following simple objective functional

$$\tilde{\mathcal{E}}(\Gamma) = \int_{\Omega} \lambda |u_{\Gamma} - u_{data}|^2 \, d\mathbf{x} + \oint_{\Gamma} ds \tag{5.3}$$

where the similarity parameter λ is 0 inside the circle region and 1 otherwise. 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 a 120° angle in the center.

Fig. 5.6 presents the solutions of the graph-cuts method [8] with different neighborhood systems. Fig. 5.6(a) gives the result using a 4-neighborhood system. It is clear that the result is not the global minimizer in the continuous setting, but it is the solution under the given graph metrics. As the neighborhood increases to 8, the result becomes quite different, as shown in Fig. 5.6(b). However, again this is not the correct solution for the problem. The result is still far from the 120° triple junction, not even close to the center, although a quite large neighborhood (32-neighborhood) has been used. This is mainly caused by the metrication error from the anisotropic total variation approximation.

Fig. 5.7 gives the solutions of the GEWCVT method with different neighborhood size. Fig. 5.7(a) gives the result using a neighborhood with $\omega = h$, where h is the mesh size. The result is not so good since the neighborhood size is too small to accurately approximate arc length. The result in Fig. 5.7(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 somewhat to the right. By increasing the size of the neighborhood to $3h$, the result is improved further, as presented in Fig. 5.7(c). The junction is drawn back to the center, although there are server pixels' shifting. When the size of the neighborhood becomes $4h$, this shifting is reduced to 2 pixels, as presented in Fig. 5.7(d). However, an even larger neighborhood, e.g. $\omega = 5h$, will not change the result further, which implies that we have already obtained a convergent solution. In contrast to the result presented in Fig. 5.6(d), the result in Fig. 5.7(d) is very close to the exact solution, which clearly illustrates that the GEWCVT method has a smaller model error than the multi-phase graph-cuts method and could be a better approximation to the Mumford–Shah model [15]. Fig. 5.8 illustrates evolution of the labels with the iteration for the case of $\omega = 4h$.

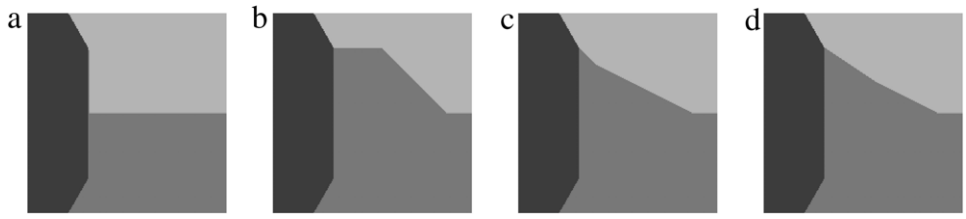

Fig. 5.6. Results of the graph-cuts method for the triple junction problem. (a)-(d) show the results with different neighborhoods: (a) 4-neighborhood; (b) 8-neighborhood; (c) 16-neighborhood; (d) 32-neighborhood.

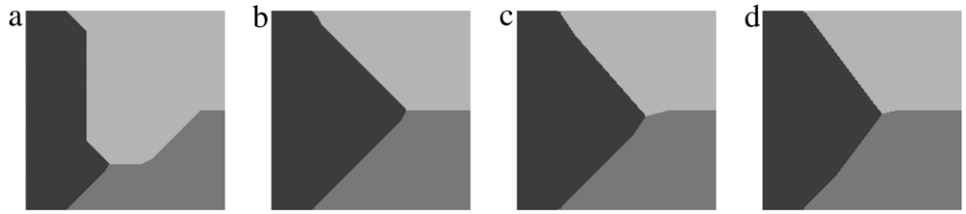

Fig. 5.7. Results of the GEWCVT method for the triple junction problem. (a)-(d) show the results with the neighborhoods having different radius ω: (a) $\omega = h$; (b) $\omega = 2h$; (c) $\omega = 3h$; (d) $\omega = 4h$.

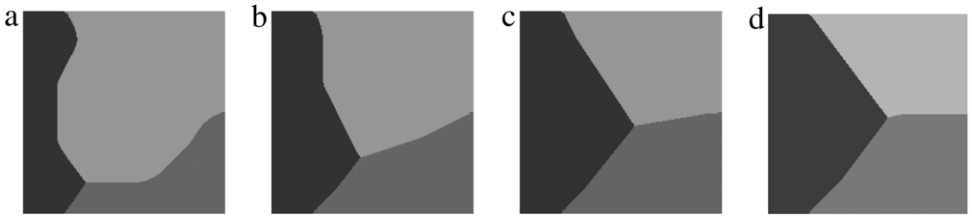

Fig. 5.8. Results of the GEWCVT method for the triple junction problem. (a)–(d) show the results after different iterations with the same neighborhood radius $\omega = 4h$: (a) after 10 iterations (b) after 100 iterations; (c) after 200 iterations; (d) final result.

Table 5.1
Computation times for the triple junction problem.

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

Table 5.1 records the CPU times for the triple junction problem. The image size used for previous examples is 200×200. 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 s to solve the triple junction problem whereas GEWCVT method takes only 2.61 s with the neighborhood size $\omega = 5h$; at the same time, the result produced by the GEWCVT method is much more accurate than the one obtained by the graph-cuts method. To conclude, the graph-cuts method could be more adapted for applications in which only small neighborhoods are used, and the GEWCVT method is often more suitable and accurate for applications where large neighborhoods are needed.

Fig. 5.9 presents a surface smoothing example, in which a noisy round-cube is smoothed by the proposed GEWCVT method. Since there are no sharp corners and edges, the similarity parameter λ_{data} is selected as the constant 1.0 in this example, and $\alpha = 0.6$, $\beta = 0$ and $\omega = 4h$. A $200 \times 200 \times 200$ rectangular mesh is used for this example. It takes 214.3 s to obtain the result as shown in Fig. 5.9(b), from which it can be seen that noise is removed.

Fig. 5.10 illustrates another surface smoothing example. For this example, a $400 \times 400 \times 400$ rectangular mesh and $\omega = 5h$ are used. This highly noisy surface has some obvious features, which are smeared out when the constant similarity parameter is used, as shown in Fig. 5.10(b). In order to preserve these features, the data-dependent similarity parameter function λ_{data} is again used. Algorithm 4 can be straightforwardly extended to the 3D case to detect corners and edges. In this example the reference ball in Algorithm 4 is chosen as a ball with radius $r = 2\omega$, and $\lambda_{data}(\mathbf{x})$ is computed by formula (4.5) with $\lambda_1 = 1.0$, $\lambda_2 = 20000.0$. Fig. 5.10(c) presents the smoothing result with this similarity parameter function, from which it can be observed that all important features are well preserved.

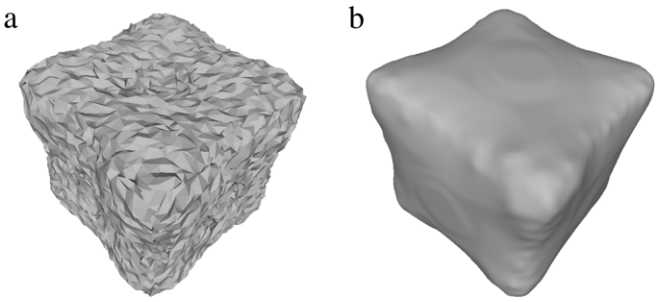

Fig. 5.9. Smoothing of a surface by our GEWCVT-based algorithm with $\omega = 4h$, $\lambda_{data} = 0.6$ and $\beta = 0.0$. (a) The noisy surfaces; (b) the smoothed surface (CPU time = 214.3 s).

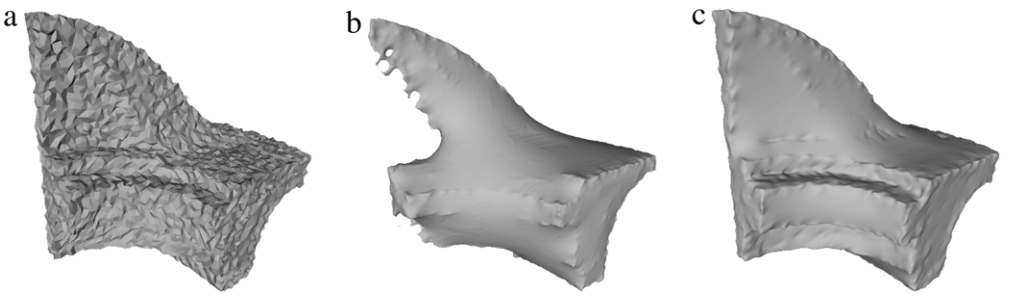

Fig. 5.10. Smoothing of a highly noisy surface by our GEWCVT-based algorithm with $\omega = 5h$ and $\beta = 0.0$. (a) The noisy surfaces; (b) the smoothed surfaces using $\lambda_{data} = 1.0$ and $\alpha = 1.0$ (CPU time = 1042.34 s). (c) The smoothed surfaces using $\alpha = 1.0$ and a nonconstant $\lambda_{data}(\mathbf{x})$ defined in (4.5) with $\lambda_1 = 1.0$, $\lambda_2 = 20000.0$ (CPU time = 3235.06 s).

5.2. Reconstruction examples

As described in Section 4, in the reconstruction applications a region Ω_{Fuzzy} with a proper width d_{th} must be first established, which divides the whole domain into several disconnected subregions, and u_{data} assigns different labels to each of the subregions. Inside the Ω_{Fuzzy} 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 Ω_{Fuzzy} has label 1 as well. It is worth pointing out that the width of the Ω_{Fuzzy} is crucial in reconstruction applications. By the definition of the Ω_{Fuzzy}, the Ω_{Fuzzy} may not be watertight for too small d_{th}. On the other hand, too large d_{th} will result in bad initial guesses, and hence longer iteration time. For a problem whose phase number is given, d_{th} can be easily determined as follows:(1) Set $d_{th} = 2h$; (2) Double d_{th} until our region growing method can detect the phase number correctly; (3) For the noisy example, continue increasing the d_{th} by the multiplication factor 2 as long as the phase number does not change. For a problem whose phase number is unknown, one can still use step (1) and (3) to determine d_{th}. For most reconstruction examples $\alpha = 0.0$ and we find that $\beta = 60$ works for most examples. Only for noisy cases, one needs to tune the parameter α to remove noise.

Fig. 5.11 shows reconstruction of a Chinese character "WANG". The noisy point set is shown in Fig. 5.11(a) and it roughly separates 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 fuzzy region is taken to be $d_{th} = 10h$ We define λ_{data} according to (4.7) with $C_{\Omega_{Fuzzy}}$=1e6 and $\beta = 60$. The reconstructed curve using our GEWCVT-based method algorithm with $\alpha = 0$ is presented in Fig. 5.11(b) and that with $\alpha = 0.02$ in Fig. 5.11(c). With extra regularization, the resulting curve in Fig. 5.11(c) is clearly smoother than the one in Fig. 5.11(b), as expected, however it also requires longer computation time.

Fig. 5.12 shows reconstruction of another Chinese character "LI" that has three disconnected components. This example can be viewed as a four-phase reconstruction problem, which still can be automatically handled by the proposed GEWCVT-based method. The point set is shown in Fig. 5.12(a) and the reconstructed curve is presented in Fig. 5.12(b). The GEWCVT algorithms took 3.03 s on a 1000×1000 grid, while the width of the fuzzy region is $d_{th} = 5h$, $\omega = 5h$, $\alpha = 0.02$, and $\beta = 60$ this time.

The following examples present results of surface reconstruction in three-dimensional space. In Fig. 5.13 a synthesized example by adding noise to a sphere is used to study the performance of the proposed GEWCVT method in highly noisy cases. For comparison, the reconstructed surfaces obtained by the Crust algorithm [36] and Poisson Surface Reconstruction (PSR) [37] are also presented. The noisy point set is shown in Fig. 5.13(a). Fig. 5.13(b) presents the reconstructed surface obtained by the Crust algorithm (the source code we used is provided by Luigi Giaccari http://www.mathworks.com/matlabcentral/fileexchange), from which it can be seen that the influence of noise is not eliminated. Fig. 5.13(c) shows the result of PSR (the source code is from http://www.cs.jhu.edu/~misha/Code/PoissonRecon/).

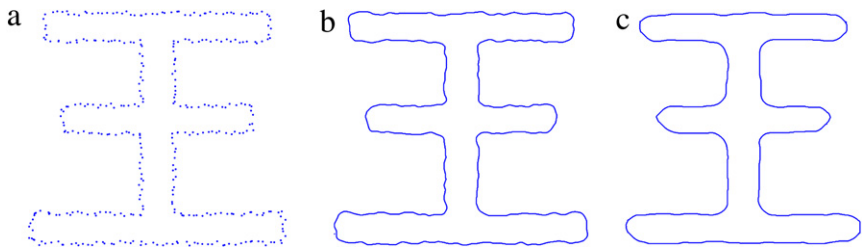

Fig. 5.11. Reconstruction of a Chinese character "WANG" by our GEWCVT-based algorithm with $beta = 60$ and $d_{th} = 10\max(h_x, h_y)$. (a) The given noisy point set; (b) the reconstructed curve using $\alpha = 0$ (CPU time = 2.08 s). (c) the reconstructed curve using $\alpha = 0.02$ (CPU time = 3.33 s).

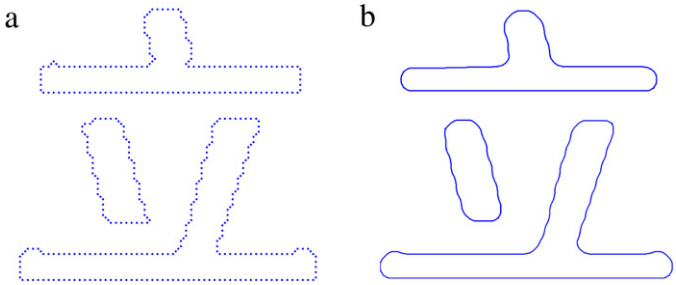

Fig. 5.12. Reconstruction of a Chinese character "LI" by our GEWCVT-based algorithm with $\alpha = 0.02$, $\beta = 60$ and $d_{th} = 5h$. (a) The given noisy point set; (b) the reconstructed curve (CPU time = 3.03 s).

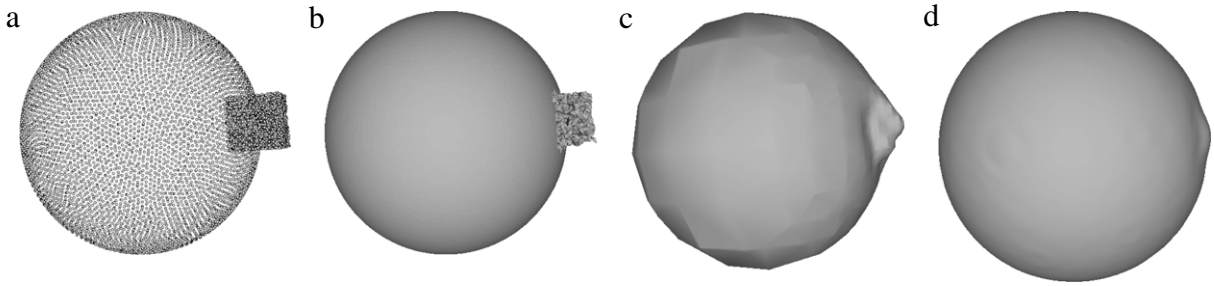

Fig. 5.13. Comparisons for an extremely noisy point set. (a) The given unorganized noisy point set; (b) the reconstructed surface generated by the Crust algorithm; (c)the reconstructed surface generated by PSR with *depth* = 10, *samplesPerNode* = 40; (d) the reconstructed surface generated by our GEWCVT method using a $200 \times 200 \times 200$ mesh, and $\alpha = 0.01$, $\beta = 1.0$.

Though the PSR can smooth out noise very well, in this extreme case the reconstruction by PSR still has a lot of undesirable patches caused by noise. In contrast, our method yields a much smoother reconstruction, as shown in Fig. 5.13(d). In order to remove the noise, we set the regularization parameter $\alpha = 0.01$ in this example. The computation times for Crust, Poisson Surface Reconstruction and GEWCVT are 10.47 s, 15.23 s and 34.44 s respectively. Although the current GEWCVT implementation is the slowest among the three methods, it yields the best reconstruction even in this extremely noisy case.

In Fig. 5.14, the initial noisy point set presents the shape of a horse, as shown in Fig. 5.14(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 (4.7) with $C_{\Omega_{Fuzzy}}$=1.0e6. The width of the fuzzy region 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 Fig. 5.14(b) and (c) for the results. As mentioned before, a few passes of the Laplacian smoothing are performed on the reconstructed surfaces to eliminate the stair-case artifacts. Compared with Fig. 5.14(b), the surface in Fig. 5.14(c) has more details in the face and presents clear grains on the muscles. Thus, a higher resolution mesh results in better quality, but also more computation cost.

The example shown in Fig. 5.15 is a reconstruction of a girl model called "Bimba", whose data set has 74,764 points (Fig. 5.15(a)) and was obtained from Aim@Shape Shape Repository (http://shapes.aim-at-shape.net/). All parameters are set to the same values as in the previous example except the width $\Omega_{Fuzzy} d_{th} = 4h$. In order to capture fine features such as the plaits of the girl, the surface reconstruction is carried on a $800 \times 800 \times 800$ mesh which consumes 464.09 s. From Fig. 5.15(b), it can be observed that all the fine features such as plaits and facial expressions are faithfully reconstructed.

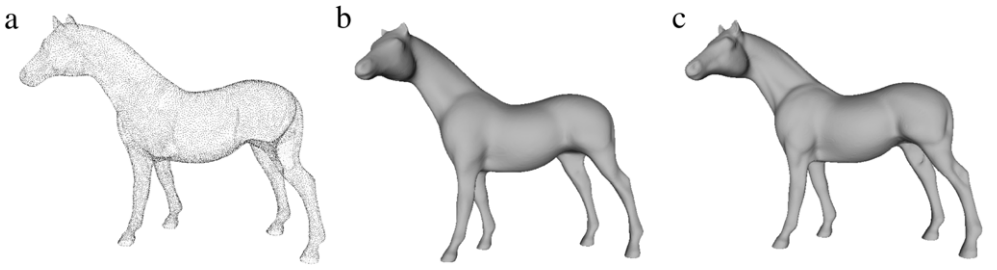

Fig. 5.14. Reconstruction of a 3D horse on different meshes by our GEWCVT-based algorithm with $\alpha = 0.0$, $\beta = 60$, and $d_{th} = 2h$. (a) The given noisy point set; (b) The reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 52.6 s); (c) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time = 397.1 s).

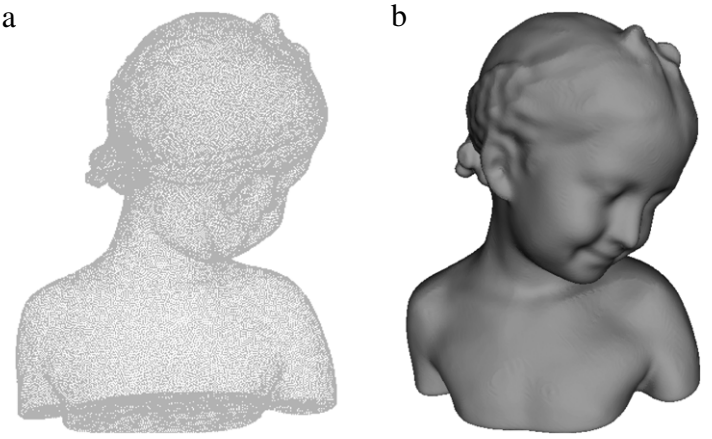

Fig. 5.15. Reconstruction of a "Bimba" model by our GEWCVT-based algorithm with $\alpha = 0.0$, $\beta = 60$, and $d_{th} = 4h$. (a) The given noisy point set; (b) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time = 464.1 s).

The following example in Fig. 5.16 demonstrates that the proposed GEWCVT-based method is capable of tackling non-uniform sampling. As shown in Fig. 5.16(a), the point set (196,278 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 if the hole is large and fine features exist. Actually, since the sampling of this example is quite dense, apart from the hole, by specifying a relative thick fuzzy 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 Fig. 5.14. Fig. 5.16(b) shows the reconstructed surface on a $200 \times 200 \times 200$ mesh, where the width of the Ω_{Fuzzy} region $d_{th} = 10h$, which is large enough to envelop the hole. It is obvious that the hole is filled, as shown in Fig. 5.16(b). Due to the existence of the hole, as the mesh size decreases the width of the Ω_{Fuzzy} region has to be maintained; 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 a considerably long computation time since the solution depends on the choice of the initial Ω_{Fuzzy} region. In order to efficiently obtain good initial guesses in the 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 *Curst* 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 as the mesh size of the coarse mesh and fine mesh respectively. The detailed idea is as follows. Let $u^{2h}_{i,j,k}$ be the solution on the coarse mesh, and $\tilde{u}^{h}_{i,j,k}$ be a labeling function on the fine mesh. If $u^{2h}_{i,j,k}$ has an adjacent node which has a different label, $\tilde{u}^{h}_{2i,2j,2k}$ and $\tilde{u}^{h}_{2i\pm1,2j\pm1,2k\pm1}$ are marked as Ω_{Fuzzy}, otherwise $\tilde{u}^{h}_{2i,2j,2k}$ and $\tilde{u}^{h}_{2i\pm1,2j\pm1,2k\pm1}$ are set equal to $u^{2h}_{i,j,k}$. By the above technique every node in the fine mesh has an initial label by setting $u^{h}_{data} = \tilde{u}^{h}$, 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 surfaces from the frog model using our GEWCVT-based algorithm on finer meshes are presented in Fig. 5.16(c) and (d) respectively.

Fig. 5.17 presents the surface reconstruction of a statue called "Fertility". The point set consists of 241,607 points and is obtained from the Aim@Shape Shape Repository as well. This model has several regions which are under-sampled, as shown in Fig. 5.17(a), and moreover it has thin and elongated parts which require high resolution to preserve the correct topology. Use of a uniform mesh with at least 400×400 resolution is needed, otherwise the arms could be be disconnected. The result shown in Fig. 5.17(b) is reconstructed using a $800 \times 800 \times 800$ mesh by our GEWCVT-based algorithm with the same parameters as the example in Fig. 5.16(d). It takes in total 1859.9 s.

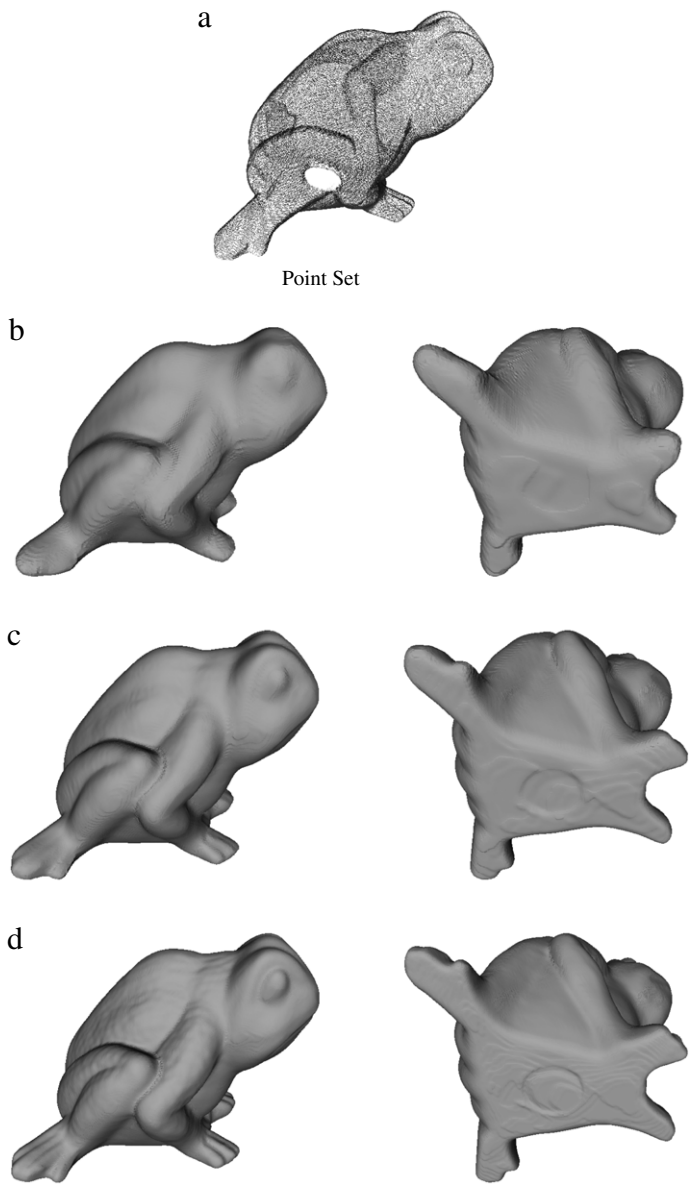

Fig. 5.16. Reconstruction of a frog model by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$. (a) The given noisy point set; (b) the reconstructed surface using a $200 \times 200 \times 200$ mesh with $d_{th} = 10h$ (CPU time $= 73.5$ s); (c) the reconstructed surface using a $400 \times 400 \times 400$ mesh with $d_{th} = 20h$ (CPU time $= 194.7$ s); (d) the reconstructed surface using a $800 \times 800 \times 800$ mesh with $d_{th} = 40h$ (CPU time $= 916.7$ s).

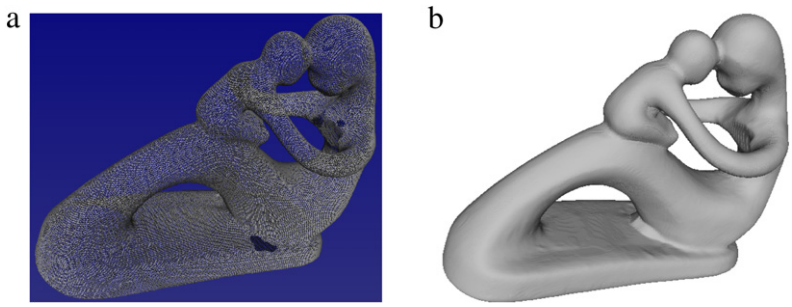

Fig. 5.17. Reconstruction of the "Fertility" model by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 40h$. (a) The given noisy point set; (b) the reconstructed surface using a $800 \times 800 \times 800$ mesh (CPU time $= 1859.9$ s).

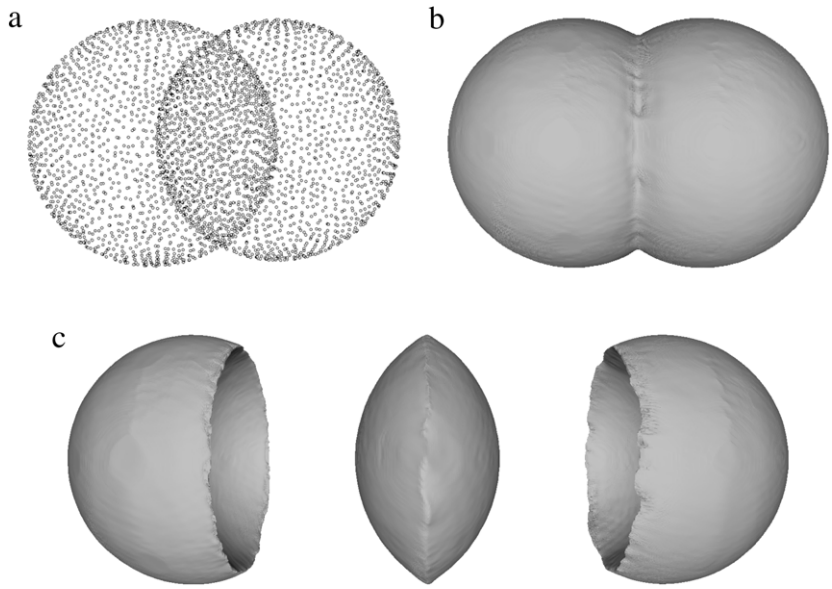

Fig. 5.18. Reconstruction of the surface of two intersecting spheres by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 8h$. (a) The given noisy point set; (b) the reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 294.7 s); (c) the decomposition of the surface for visualization.

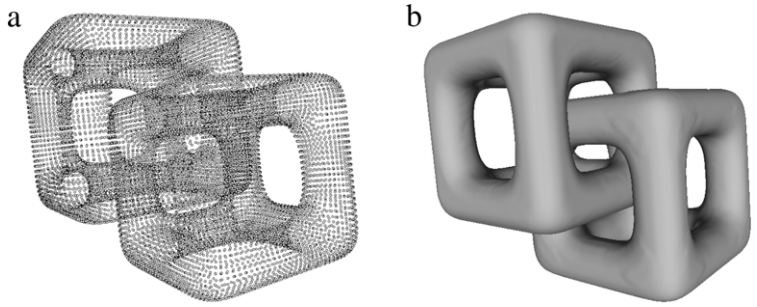

Fig. 5.19. Reconstruction of two entangled cubes by our GEWCVT-based algorithm with $\alpha = 0.0$ and $\beta = 60$ and $d_{th} = 8h$. (a) The given unorganized noisy point set; (b) the reconstructed surface using a $400 \times 400 \times 400$ mesh (CPU time = 313.7 s).

Fig. 5.18 illustrates a multi-phase (four phases) example of two intersected spheres. Fig. 5.18(a) shows the point set, which consists of 3,442 points. The $400 \times 400 \times 400$ mesh is used to resolve the intersection parts. The width of the Ω_{Fuzzy} is $d_{th} = 8h$, which is relatively large since the sampling points are very coarse. Fig. 5.18(b) presents the reconstructed surface by our algorithm viewed from the outside, and Fig. 5.18(c) gives the decomposition of the surface, from which the topology of the surface can be easily observed.

Fig. 5.19 presents another multi-phase (two phases) reconstruction example, in which two cubic frames are entangled with each other. The input data has 15,344 points, as shown in Fig. 5.19(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 takes 313.75 s, see Fig. 5.19(b) for the final reconstructed surface.

6. Conclusions and future Works

In this paper, we propose an important generalization of the edge-weighted centroidal Voronoi tessellation model developed in [1]. By this new model, a large range of variational models can be interpreted in CVT or clustering language. A unified variational framework is exposed in this paper, and solution algorithms are developed and applied to geometry processing such as curve/surface smoothing and reconstruction. Moreover, compared to all existing methods, the proposed model is able to address challenging multi-phase problems in an accurate and easy-to-implement way. Through various numerical examples, the proposed GEWCVT-based method is shown to be an effective and robust tool for such applications. As a new clustering method, the GEWCVT method can not only be applied to image segmentation [1] and geometry processing, but also some problems in data analysis and quantization, which is one of our future projects. On the other hand, there is still much scope for improving the efficiency of the GEWCVT-based methods in 3D applications, such as adopting

the octree grids as in PSR [37], quasi-Newton type iterations [38] or adaptive tetrahedron meshes as in [8]. In addition, since all the computations in the GEWCVT method are done in discrete space, using a GPU (graphics processing unit) to accelerate the GEWCVT method as [39] is also an interesting and important research topic for us. The alternating convex minimization techniques developed in [40,41] prove very efficient for solving L^1 regularized functionals but the nonconvexity of our functional makes it difficult to directly apply these fast minimization techniques. One promising research direction is to study the feasibility of applying the alternating minimization on the convexified functional, as done in [42,43].

Acknowledgments

The authors would like to thank the anonymous referees whose suggestions considerably contributed the improvement of this work.

References

[1] J. Wang, L. Ju, X. Wang, An edge-weighted centroidal Voronoi tessellation model for image segmentation, IEEE Transactions on Image Processing 18 (2009) 1844–1858.
[2] M. Desbrun, M. Meyer, P. Schröder, 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.
[3] S. Rusinkiewicz, O. Hall-Holt, M. Levoy, Real-time 3D model acquisition, ACM Transactions on Graphics 21 (2002) 438–446.
[4] 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.
[5] H. Zhao, S. Osher, 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.
[6] V. Kolmogorov, R. Zabin, What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 147–159.
[7] E. Bae, 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) 1–13.
[8] M. Wan, Y. Wang, D. Wang, Variational surface reconstruction based on Delaunay triangulation and graph cut, International Journal for Numerical Methods in Engineering 85 (2011) 206–229.
[9] Y. Boykov, 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.
[10] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, A. Wu, An efficient *k*-means clustering algorithm: analysis and implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 881–892.
[11] Q. Du, M. Gunzburger, L. Ju, X. Wang, Voronoi tessellation algorithms for image compression and segmentation, Journal of Mathematical Imaging and Vision 24 (2006) 177–194.
[12] Q. Du, V.F aber, M. Gunzburger, Centroidal Voronoi tessellations: applications and algorithms, SIAM Review 41 (1999) 637–676.
[13] Q. Du, M. Gunzburger, L. Ju, Advances in studies and applications of centroidal Voronoi tessellations, Numerical Mathematics: Theory, Methods and Applications 3 (2010) 119–142.
[14] J. Wang, X. Wang, Edge-weighted centroidal Voronoi tessellations, Numerical Mathematics: Theory, Methods and Applications 3 (2010) 223–244.
[15] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, Communications on Pure and Applied Mathematics 42 (1989) 577–685.
[16] Y. Wang, S. Song, Z. Tan, D. Wang, Adaptive variational curve smoothing based on level set method, Journal of Computational Physics 228 (2009) 6333–6348.
[17] L. Zhu, Y. Wang, L. Ju, D. Wang, A variational phase field method for curve smoothing, Journal of Computational Physics 229 (2010) 2390–2400.
[18] S. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[19] T. Chan, J. Shen, L. Vese, Variational PDE models in image processing, Notices of AMS 50 (2003) 14–26.
[20] Q. Du, C. Liu, X. Wang, A phase field approach in the numerical study of the elastic bending energy for vesicle membranes, Journal of Computational Physics 198 (2004) 450–468.
[21] Q. Du, C. Liu, X. Wang, Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions, Journal of Computational Physics 212 (2006) 757–777.
[22] H. Zhao, A fast sweeping method for eikonal equations, Mathematics of Computation 74 (2005) 603–627.
[23] V. Caselles, R. Kimmel, Sapiro G, Geodesic active contours, International Journal of Computer Vision 22 (1997) 61–79.
[24] L. Cohen, E. Bardinet, N. Ayache, Surface reconstruction using active contour models, in: Proceedings of SPIE Conference on Geometric Methods in Computer Vision, SPIE, Bellingham, 1993.
[25] A. Hornung, L. Kobbelt, 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.
[26] Y. Wang, D. Wang, A.M. Bruckstein, On variational curve smoothing and reconstruction, Journal of Mathematical Imaging and Vision 37 (2010) 183–203.
[27] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (1982) 129–137.
[28] Q. Du, M. Emelianenko, L. Ju, Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations, SIAM Journal on Numerical Analysis 44 (2006) 102–119.
[29] W. Lorensen, 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.
[30] J. Wang, L. Ju, X. Wang, Image segmentation using local variation and edge-weighted centroidal Voronoi tessellations, IEEE Transactions on Image Processing 20 (2011) 3242–3256.
[31] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 256–268.
[32] A.M. Yip, F. Park, Solution dynamics, causality, and critical behavior of the regularization parameter in total variation denoising problems, Tech. Report, UCLA CAM Report, 2003.
[33] D.M. Strong, J.-F. Aujol, T.F. Chan, Scale recognition, regularization parameter selection, and Meyers g norm in total variation regularization, Tech. Report, UCLA CAM Report, 2005.
[34] H. Liao, F. Li, M.K. Ng, Selection of regularization parameter in total variation image restoration, Journal of the Optical Society of America A 26 (2009) 2311–2320.
[35] Y. Boykov, V. Kolmogorov, Computing geodesic and minimal surfaces via graph cuts, in: IEEE International Conference on Computer Vision, 2003, pp. 26–33.

[36] N. Amenta, M. Bern, Surface reconstruction by Voronoi filtering, in: SCG '98: Proceedings of the Fourteenth Annual Symposium on Computational Geometry, 1998, pp. 39–48.
[37] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, 2006, pp. 61–70.
[38] Y. Liu, W. Wang, B. Levy, F. Sung, D.-M. Yan, On centroidal Voronoi tessellation - energy smoothness and fast computation, ACM Transaction on Graphics 28 (2009) 1–17.
[39] G. Rong, Y. Liu, W. Wang, X. Yin, X. Gu, X. Guo, GPU-assisted computation of centroidal Voronoi tessellation, IEEE Transactions on Visualization and Computer Graphics 17 (2011) 345–356.
[40] W. Yin, S. Osher, D. Goldfarb, J. Darbon, Bregman iterative algorithms for l_1-minimization with applications to compressed sensing, SIAM Journal on Imaging Sciences 1 (2008) 143–168.
[41] Y. Wang, J. Yang, W. Yin, Y. Zhang, A new alternating minimization algorithm for total variation image reconstruction, SIAM Journal on Imaging Sciences 1 (2008) 248–272.
[42] T. Goldstein, X. Bresson, S. Osher, Geometric application of the split Bregman method: segmentation and surface reconstruction, Journal of Scientific Computing 45 (2010) 272–293.
[43] J. Hahn, G. Chung, Y. Wang, X.-C. Tai, Fast algorithms for p-elastica energy with the application to image inpainting and curve reconstruction, in: 3rd International Conference on Scale Space and Variational Methods in Computer Vision, Ein-Gedi, Israel, 2011, pp. 169–182.