On Centroidal Voronoi Tessellation—Energy Smoothness and Fast Computation

YANG LIU
Project ALICE, INRIA and The University of Hong Kong
WENPING WANG
The University of Hong Kong
BRUNO LÉVY
Project ALICE, INRIA
FENG SUN, DONG-MING YAN, and LIN LU
The University of Hong Kong
and
CHENGLEI YANG
Shandong University

Centroidal Voronoi tessellation (CVT) is a particular type of Voronoi tessellation that has many applications in computational sciences and engineering, including computer graphics. The prevailing method for computing CVT is Lloyd's method, which has linear convergence and is inefficient in practice. We develop new efficient methods for CVT computation and demonstrate the fast convergence of these methods. Specifically, we show that the CVT energy function has 2nd order smoothness for convex domains with smooth density, as well as in most situations encountered in optimization. Due to the 2nd order smoothness, it is possible to minimize the CVT energy functions using Newton-like optimization methods and expect fast convergence. We propose a quasi-Newton method to compute CVT and demonstrate its faster convergence than Lloyd's method with various numerical examples. It is also significantly faster and more robust than the Lloyd-Newton method, a previous attempt to accelerate CVT. We also demonstrate surface remeshing as a possible application.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**] Computational Geometry and Object Modeling; G.1.6 [**Numerical Analysis**] Optimization; I.5.3 [**Pattern Recognition**] Clustering

General Terms: Algorithms

Additional Key Words and Phrases: Centroidal Voronoi tessellation, constrained CVT, Lloyd's method, remeshing, numerical optimization, quasi-Newton methods

ACM Reference Format:

Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Lu, L., and Yang, C. 2009. On centroidal Voronoi tessellation—Energy smoothness and fast computation. ACM Trans. Graph. 28, 4, Article 101 (August 2009), 17 pages. DOI = 10.1145/1559755.1559758 http://doi.acm.org/10.1145/1559755.1559758

1. INTRODUCTION

A *centroidal Voronoi tessellation (CVT)* is a particular Voronoi tessellation of a compact domain in Euclidean space yielded by a set of samples (also called sites or generators) such that each site coincides with the centroid of its Voronoi cell. For example, Figure 1(a) shows a Voronoi tessellation of a circular domain where the sites do not coincide with the centroids of the Voronoi cells (not a CVT), and Figure 1(b) shows a CVT of the same domain. CVT generates an evenly-spaced distribution of sites in the domain with respect to a given density function, and is therefore very useful in many fields, such as optimal quantization, clustering, data compression,

The work of W. Wang is partially supported by the Research Grant Council of Hong Kong (project no.: 718209), the National Key Basic Research Project of China under 2004C318000, and a Hong Kong General Research Fund (project no.: 717808). B. Lévy and Y. Liu are supported by the European Research Council (GOODSHAPE FP7-ERC-StG-205693). The work of C. Yang is supported by National Natural Research Council of China (60703028). The Rocker and Lion models are courtesy of AIM@SHAPE Project.

Authors' addresses: Y. Liu and B. Lévy, Project ALICE, INRIA, Villers les Nancy, France; email:{liuyang, levy}@loria.fr; W. Wang, F. Sun, D.-M. Yan, L. Lu, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong, China; email: {wenping, fsun, dmyan, llu}@cs.hku.hk; C. Yang, School of Computer Science and Technology, Shandong University, China; email: chl_yang@sdu.edu.cn.

DOI 10.1145/1559755.1559758 http://doi.acm.org/10.1145/1559755.1559758

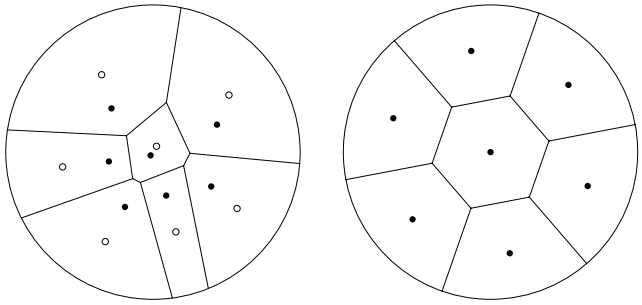


Fig. 1. (a) An ordinary Voronoi tessellation of a circular domain with seven sites marked with black dots and the the centroids of the Voronoi cells marked with small circles; (b) a CVT of the same domain with seven sites.

optimal mesh generation, cellular biology, optimal quadrature, coverage control, and geographical optimization. An excellent introduction to the theory and applications of CVT is given in Du et al. [1999] and Okabe et al. [2000].

CVT was recently applied to mesh generation and geometry processing [Du and Gunzburger 2002; Du and Wang 2003, Alliez et al. 2003, 2005, Valette and Chassery 2004] and vector field visualization [Du and Wang 2004; McKenzie et al. 2005]. Peyré and Cohen [2004] extend CVT using geodesic metric on mesh surfaces. Anisotropic CVT on surfaces is also considered by Du et al. [2003], Du and Wang [2005a] and Valette et al. [2008].

Equivalently, CVT can be defined by the critical points (gradient-vanishing points) of a certain CVT energy function, F, which we will discuss in detail shortly. There are several outstanding problems with CVT, such as computing a CVT with the globally minimal CVT energy and shape characterization of this globally optimal CVT, as stipulated by Gersho [1979]. The globally minimal CVT is difficult to obtain because the CVT energy function is nonlinear and nonconvex. Gersho's conjecture in 2D has been proved by Tòth [2001], asserting that in a globally optimal CVT, the shape of the Voronoi cells that are far away from the boundary converge to regular hexagons as the number of sites tends to infinity [Gruber 2004]. Gersho's conjecture is still open in nD, $n \geq 3$, though partial empirical results are available in 3D [Du and Wang 2005b].

In the present article, we are interested in efficient CVT computation. The most popular method for computing CVT is Lloyd's method [1982]. The popularity of Lloyd's method is due to its simplicity and robustness—it decreases the CVT energy value monotonically. However, Lloyd's method is not optimally efficient—it has only linear convergence and is therefore slow for practical applications with a large number of sites.

The probabilistic method by MacQueen [1966] is another method for CVT computation, but is not widely used in practice because of its lack of computational advantage. For acceleration of CVT computation, Lloyd's method has been implemented in a multigrid framework [Du and Emelianenko 2006] and MacQueen's method on a parallel platform [Ju et al. 2002].

Although CVT is naturally formulated as the solution of an optimization problem, there has been little progress in efficient CVT computation beyond Lloyd's method, from the optimization point of view. This is probably due to the complicated piecewise nature of the CVT energy function, F [Iri et al. 1984; Asami 1991; Du et al. 1999] (more on this follows). It is known that F has C^1 smoothness [Cortés et al. 2005]. In this article, we gain more knowledge about the properties of F and prove that it has C^2 smoothness, thus enabling us to propose a Newton-type optimization algorithm.

One notable previous attempt at accelerating CVT computation is the Lloyd-Newton method [Du and Emelianenko 2006]. Since this method minimizes a function that is different from the CVT energy function F, it often produces undesirable CVTs corresponding to unstable critical points of the CVT energy function. We will analyze this in detail in Section 5.2.

We make the following contributions in both theory and applications for efficient CVT computation:

—we show that the piecewise CVT function is C^2 for a convex compact domain in 2D and 3D as well as other commonly encountered domains with a sufficiently smooth (C^2) density function;

—we accelerate CVT computation by applying quasi-Newton methods and show that these methods are more efficient than both Lloyd's method and the Lloyd-Newton method [Du and Emelianenko 2006], as expected due to the newly established C^2 smoothness of the CVT energy function;

—we develop an efficient quasi-Newton method for computing the constrained CVT on polyhedral surfaces for surface remeshing.

The organization of the article is as follows: in Section 2 we present the formulation of the CVT problem and review the existing work on CVT computation. We show in Section 3 that the CVT energy function is C^2 for a convex compact domain with sufficiently smooth density in 2D/3D space, and propose in Section 4 to use quasi-Newton methods to accelerate CVT computation. We demonstrate the efficiency of the method in Section 5. In Section 6, we apply our smoothness analysis and fast method to the more general context of CVT computation on mesh surfaces in 3D, and show how it can be used for quality surface remeshing. We conclude the article in Section 7.

2. BACKGROUND AND PREVIOUS WORK

2.1 CVT Formulation

We will first briefly review CVT. A good overview can be found in Du et al. [1999]. Let $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ be an *ordered set* of n sites in a connected compact region $\Omega \subset \mathbb{R}^N$. The Voronoi region Ω_i of $\mathbf{x}_i$ is defined as:

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \forall j \neq i\}.$$

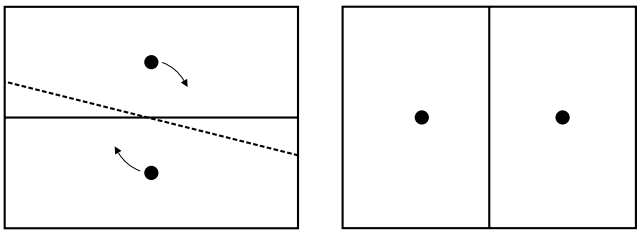
Here $\|\cdot\|$ denotes Euclidean norm in $\mathbb{R}^N$. The Voronoi regions, Ω_i, of all the sites form the Voronoi diagram (VD) of $\mathbf{X}$. A natural assumption is that any two sites are distinct: $\mathbf{x}_i \neq \mathbf{x}_j$, $\forall i \neq j$, because the Voronoi boundary consists of bisecting lines of pairs of sites and a bisecting line is not well-defined for two identical sites. Hence, the space of $\mathbf{X}$ that we will consider is $\mathbf{\Gamma} := \{\mathbf{X} \in \mathbb{R}^{nN} | \mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j; \mathbf{x}_i \in \Omega, \forall i\}$.

Let the domain Ω be endowed with a density function $\rho(\mathbf{x}) > 0$, which is assumed to be C^2; therefore $\rho(\mathbf{x})$ is bounded, since Ω is closed; that is, $\sup_{\mathbf{x}\in\Omega} |\rho(\mathbf{x})| \leq \gamma$ for some finite value $\gamma > 0$. Then the centroid of Ω_i is given by:

$$\mathbf{c}_i = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}\, d\sigma}{\int_{\Omega_i} \rho(\mathbf{x})\, d\sigma},$$

where $d\sigma$ is the area differential.

Definition 1. The *Voronoi tessellation* $\{\Omega_i\}_{i=1}^n$ is a centroidal Voronoi tessellation if $\mathbf{x}_i = \mathbf{c}_i$, $i = 1, 2, \ldots, n$, that is, each site $\mathbf{x}_i$ coincides with the centroid of its Voronoi cell Ω_i.



(a) (b)

Fig. 2. (a) An unstable CVT of two sites in a rectangular domain; (b) a stable CVT, which is also an optimal CVT of the same domain. Initialized with slightly perturbed positions of the sites from the unstable CVT, as shown in (a), Lloyd's iteration will converge to the stable CVT in (b).

A CVT can also be defined from a variational point of view. Define $F_i(\mathbf{X}) = \int_{\mathbf{x}\in\Omega_i} \rho(\mathbf{x})\|\mathbf{x}-\mathbf{x}_i\|^2\,\mathrm{d}\sigma$ for each site $\mathbf{x}_i$. Then the *CVT energy function* $F : \mathbf{\Gamma} \to \mathbb{R}$ of the Voronoi tessellation $\{\Omega_i\}_{i=1}^n$ is defined as [Du et al. 1999]:

$$F(\mathbf{X}) = \sum_{i=1}^{n} F_i(\mathbf{X}) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(\mathbf{x})\|\mathbf{x}-\mathbf{x}_i\|^2\,\mathrm{d}\sigma. \tag{1}$$

The term F_i expresses the compactness (or inertia momentum) of the Voronoi cell Ω_i. This requirement for compactness is desirable in many applications. For instance, in sampling theory, we can imagine that Ω represents a space that needs to be approximated (e.g., the color space of an image) and that the sites $\mathbf{x}_i$ correspond to samples (e.g., the elements of a colormap). In this case, minimizing the energy, F, ensures that each sample, $\mathbf{x}_i$, is a representative of approximately the same amount of information in Ω. Hence, from a quantization point of view, F corresponds to the quantization noise power [Lloyd 1982].

The gradient of $F(\mathbf{X})$ is [Iri et al. 1984; Asami 1991; Du et al. 1999]:

$$\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i(\mathbf{x}_i - \mathbf{c}_i), \tag{2}$$

where

$$m_i = \int_{\mathbf{x}\in\Omega_i} \rho(\mathbf{x})\,\mathrm{d}\sigma \tag{3}$$

is the mass and $\mathbf{c}_i$ the centroid of Ω_i. With this expression for the gradient, it is easy to see that a CVT corresponds to a critical point of the CVT function, $F(\mathbf{X})$, a point $\mathbf{X}_0$ where the gradient of $F(\mathbf{X})$ is zero. However, a critical point of $F(\mathbf{X})$ may be unstable, that is, when it is a saddle point characterized by an indefinite Hessian. In practice, we prefer a CVT that corresponds to a local minimizer of the CVT energy function, since it represents a more compact Voronoi tessellation than a CVT given by an unstable critical point. Hence, we have the following equivalent definition of CVT:

Definition 2. A *Voronoi tessellation* $\{\Omega_i\}_{i=1}^n$ *of a compact domain* Ω *with n sites* $\mathbf{X}_0 = (\mathbf{x}_i)_{i=1}^n$ is a centroidal Voronoi tessellation if $\mathbf{X}_0$ is a critical point of the CVT energy function $F(\mathbf{X})$. Furthermore, a CVT is called a stable CVT if $\mathbf{X}_0$ is a local minimizer of $F(\mathbf{X})$, and it is called an optimal CVT if $\mathbf{X}_0$ is a global minimizer of $F(\mathbf{X})$.

For instance, the CVT of the rectangle shown in Figure 2(a) is unstable, as it corresponds to a saddle point of F, where the Hessian is indefinite. A stable CVT, whose Hessian is positive-definite, is shown in Figure 2(b).

Since in a CVT, each site $\mathbf{x}_i$ coincides with the centroid of its Voronoi cell Ω_i, we have:

$$\forall i, \quad \mathbf{x}_i = \mathbf{c}_i = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}\,\mathrm{d}\sigma}{\int_{\Omega_i} \rho(\mathbf{x})\,\mathrm{d}\sigma}. \tag{4}$$

This is a system of nonlinear equations, since the boundaries of any Voronoi cell Ω_i are determined by all the sites $\mathbf{x}_i$.

The fact that a CVT is a critical point of F naturally leads us to Lloyd's method for computing CVT [Lloyd 1982]. Lloyd's method operates by iteratively moving all the sites to the centroids of their Voronoi cells. Its convergence to a CVT was proved in some particular cases by Du et al. [1999, 2006]. They showed that besides being a fixed-point iteration that solves Equation (4), Lloyd's method can be understood as a gradient descent method that always decreases the energy F with no need for step-size control [Du et al. 1999]. A similar analysis can be applied to the discrete k-means algorithm for clustering [Ostrovsky et al. 2006].

So far, we have seen two equivalent definitions of CVT, from two different points of view.

—*Variational characterization*: CVT is a critical point of the energy F in Equation (1). In many applications a stable CVT is desirable, rather than just a CVT given by an unstable critical point, since a stable CVT provides more compact Voronoi cells. It is these stable CVTs that we aim to compute in the present article.

—*Geometric characterization*: CVT is a solution of a system of nonlinear equations expressing that each site $\mathbf{x}_i$ coincides with the centroid $\mathbf{c}_i$ of its Voronoi cell (Equation (4)). The Lloyd-Newton method [Du and Emelianenko 2006] is based on directly solving this system of equations; as a consequence, it often produces an unstable CVT. We will give more details about this in Section 5.2.

The distinction between these two points of view is a key aspect of our approach. From the variational point of view, Lloyd's algorithm is a gradient descent method with linear rate of convergence. We will show how studying CVT computation from the variational point of view and studying the smoothness of the CVT energy F leads to methods that are more efficient than Lloyd's relaxation.

2.2 Variational Point of View

To improve the speed of convergence, we consider computing CVT by minimizing the energy function F with a quasi-Newton method. It is well known that a faster convergence rate can be obtained by using higher-order methods (e.g., Newton's method and its variants). As will explained further, the piecewise nature of the CVT energy function F and the complexity of its expression are impediments to the development of fast CVT methods. For this reason, there were few successful attempts in the literature in this direction [Iri et al. 1984; Du and Emelianenko 2006].

Newton's iteration for nonlinear optimization uses a second-order approximation of F:

$$F(\mathbf{X}+\delta_{\mathbf{X}}) \simeq F^*(\delta_{\mathbf{X}}) = F(\mathbf{X}) + \delta_{\mathbf{X}}^T \nabla F + \tfrac{1}{2}\delta_{\mathbf{X}}^T \mathbf{H}\,\delta_{\mathbf{X}}\ ,$$

where $\delta_{\mathbf{X}}$ denotes a small displacement from $\mathbf{X}$, ∇F denotes the gradient of F, and $\mathbf{H}$ is the Hessian. Newton's iteration finds the step vector $\delta_{\mathbf{X}}$ that minimizes the model function $F^*(\delta_{\mathbf{X}})$ as follows:

$$\begin{aligned} \texttt{solve} \quad \mathbf{H}\,\delta_{\mathbf{X}} &= -\nabla F(\mathbf{X}) \\ \mathbf{X} &\leftarrow \mathbf{X} + \delta_{\mathbf{X}}. \end{aligned}$$

As can be seen, since it uses the second-order derivatives, Newton's method requires F to be at least C^2 [Nocedal and Wright 2006].

Furthermore, Newton's method is not suitable for large-scale problems for which the computation of full Hessian is costly or when the Hessian matrix is not sparse. Therefore, in practice, one often uses quasi-Newton methods to deal with large-scale problems.

An impediment to the development of fast Newton-like methods for CVT computation is the lack of understanding about the smoothness of F. It was conjectured [Iri et al. 1984] that F is non-differentiable, based on the intuition that the structural change of the Voronoi diagram would make the CVT energy function non-smooth. This conjecture was later infirmed by Cortés et al. [2005], who showed that the CVT function F is C^1. We go one step further in the study of F's smoothness, and prove that it is almost always C^2. More specifically, F is C^2 for any convex domain in 2D and 3D, with C^2 smooth density. When the domain is non-convex, it is still C^2 in most situations encountered in optimization but it can become C^1 in some rare cases. Furthermore, these results carry over to the constrained CVT problem on mesh surfaces. This newly established smoothness of F provides justification for applying quasi-Newton methods to efficient CVT computation.

Before entering the details of the smoothness analysis, we need to explain why studying the continuity of F is a difficult problem. If one wants to evaluate F and its derivatives for a specific value of the set of variables $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$, it is necessary to first construct the Delaunay triangulation of the vertices defined by $\mathbf{X}$ (all the sites), then evaluate the integrals over each cell of its dual Voronoi diagram (see Equation (1)) and differentiate them. The expression of these integrals is complicated, since the vertices of the domains of integration Ω_i are the circumcenters of the Delaunay simplices.

Now suppose that we move one of the vertices to change the combinatorial structure of the Delaunay triangulation, then the expression of the CVT energy function F changes as well, since it will be based on a different triangulation. In other words, F is piecewise defined in $\mathbf{\Gamma}$ (the space of $\mathbf{X}$) and the pieces correspond to the subsets of $\mathbf{\Gamma}$ where the combinatorial structure of the Delaunay triangulation remains the same. A combinatorial change occurs in 2D when four or more sites become co-circular (or five or more sites become co-spherical in 3D). While these degenerate configurations are often considered as nuisances in mesh generation, they define gateways, or common faces, that connect the pieces of $\mathbf{\Gamma}$. Crossing such a gateway results in changes in the expression of F, hence possible discontinuity. We will show in Section 3 that F is C^2 at such transitions.

3. ENERGY SMOOTHNESS

For CVT in a 2D convex domain, we have the following result.

THEOREM 1. *The 2D CVT energy function is C^2 in $\mathbf{\Gamma}$ if Ω is convex and compact and if the density function $\rho(\mathbf{x})$ is C^2 in Ω.*

The proof is given in Appendix A. Here we will only give an intuitive idea of the proof. The CVT energy function F is C^∞-smooth as a function of the coordinates of sites as long as the combinatorial structure of Delaunay triangulation (as well as the Voronoi diagram) of the sites does not change. So we only need to analyze how the energy function changes when the combinatorial structure changes at a degenerate configuration. The Delaunay triangulation of such a degenerate configuration is non-unique and its multiple Delaunay triangulations give rise to different expressions of F. Therefore, analyzing the smoothness of F means proving that these expressions have C^2 contact at the degenerated configuration. To do so, we consider the Taylor expansions of these expressions of F at such a degenerate configuration, and show that they match up to the second-order term.

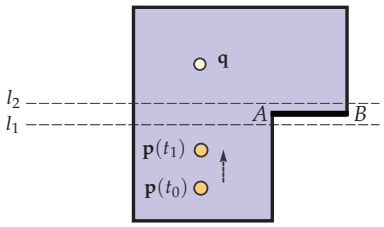


Fig. 3. A configuration where the CVT function is C^1.

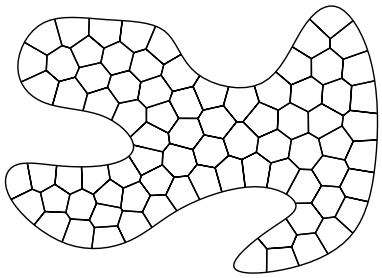


Fig. 4. A 2D non-convex CVT after several Lloyd's iterations. The CVT energy is C^2 around this configuration.

Following the same idea of the proof for the 2D case, we can also prove the following result for the 3D case. The proof will be given elsewhere, due to space limitation.

THEOREM 2. *The 3D CVT energy function is C^2 in $\mathbf{\Gamma}$ if Ω is convex and compact, and if the density function $\rho(\mathbf{x})$ is C^2 in Ω.*

Remark 1. In the more general locational optimization problem [Okabe et al. 2000], the distance function in Equation (1) is a smooth and strictly increasing function $W(\|\mathbf{x} - \mathbf{x}_i\|)$. Then the energy function becomes $F(\mathbf{X}) = \sum_{i=1}^n \int_{\mathbf{x}\in\Omega_i} \rho(\mathbf{x}) W(\|\mathbf{x} - \mathbf{x}_i\|)\, \mathrm{d}\sigma$. Theorem 1 and 2 still hold in this case.

In general, Theorems 1 and 2 do not hold for a non-convex domain Ω. The C^2 smoothness of the CVT function F is lost when a continuous part of the boundary $\partial\Omega$, of Ω, is contained in a face of a Voronoi cell or when a smooth part of $\partial\Omega$ is tangent to some face of a Voronoi cell—the CVT function is C^1 but not C^2 in these cases. This is illustrated in 2D in Figure 3. Consider the two points $\mathbf{p}$ and $\mathbf{q}$ in the 2D non-convex domain Ω. Fix the point $\mathbf{q}$ and let $\mathbf{p}$ move from $\mathbf{p}(t_0)$ up to $\mathbf{p}(t_1)$. When the bisector of $\mathbf{p}$ and $\mathbf{q}$ contains the thick horizontal edge AB, F is C^1 but not C^2.

A similar situation can occur in 3D as well. It can be shown that this is the only type of situation where C^2 smoothness of F is lost. Clearly, this situation rarely occurs in practice during optimization, since in most cases, the faces of the Voronoi cell are not parallel to the domain boundary when there are sufficiently many sites with a reasonable distribution (see Figure 4). That is to say, even if the function $F(\mathbf{X})$ is not everywhere C^2 in this case, the regions in which a method for computing CVT is applied almost always correspond to a subspace of $\mathbf{\Gamma}$ where F is C^2.

Remark 2. It seems possible that the requirement on the C^2 smoothness of the density function $\rho(\mathbf{X})$ can be relaxed. We conjecture that Theorems 1 and 2 still hold when $\rho(\mathbf{X})$ is C^0, as suggested by our empirical study.

We now use two examples to illustrate the smoothness of 2D and 3D CVT functions. We will see that, as asserted by Theorems 1

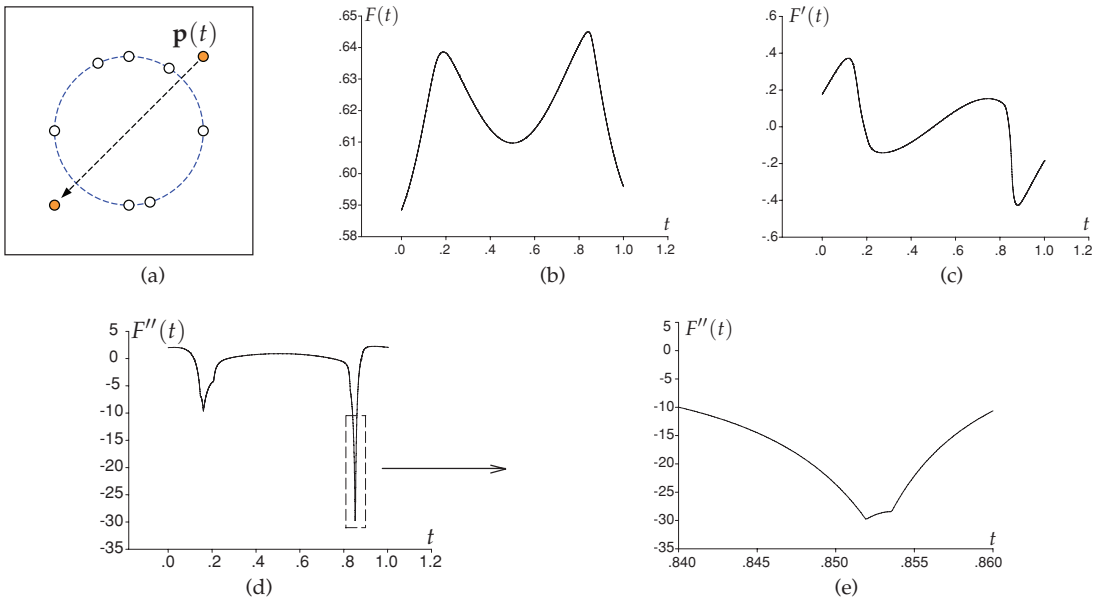


Fig. 5. Illustrations of C^2 smoothness of the 2D CVT function.

and 2, C^2 continuity of the CVT function is preserved even when the topological structure of the Voronoi diagram changes.

Example 1. 2D case: Figure 5(a) shows the domain $\Omega = [-1, 1]^2$ with eight sites. One site, $\mathbf{p}(t)$, moves along a straight path linearly parameterized by t, and the other seven sites are fixed and located on a circle. The structure of the Voronoi diagram of the eight sites changes at some points of the motion of $\mathbf{p}(t)$; for example, when $\mathbf{p}(t)$ crosses the circle. Figures 5(b) to (d) show the graphs of the CVT function $F(t)$ and its derivatives $F'(t)$, $F''(t)$ with respect to the motion parameter t. Figure 5(e) is a zoom-in view of part of the graph of $F''(t)$. We see that $F(t)$ and $F'(t)$ are smooth and $F''(t)$ is C^0. The kinks in the graph of $F''(t)$ correspond to the structural transitions of VD of the domain when the site $\mathbf{p}(t)$ crosses the circle.

Example 2. 3D case: Figure 6(a) shows the domain $\Omega = [-1, 1]^3$ with eight sites. One of the sites moves along a linear trajectory $\mathbf{p}(t)$ and it passes through spheres formed by several subsets of four of the other sites. The graphs of $F(t)$, $F'(t)$, and $F''(t)$ are shown in Figures 6(b)–(d), verifying that the CVT function F is C^2 for this domain Ω in 3D.

Note that, just for the purpose of simple illustration, we only considered the CVT energy as a function of one variable in these examples. However, our conclusion on the C^2 smoothness is about the general case where the CVT energy is the function of all the sites $\mathbf{X}$ being variables.

4. NUMERICAL OPTIMIZATION

In this section we will briefly discuss the efficiency issues related to Newton-type methods. We expand on quasi-Newton methods and describe the basic ideas of the two methods that we will use for efficient CVT computation—the L-BFGS method (*limited-memory* BFGS) and the P-L-BFGS method (*preconditioned* L-BFGS), which are two variants of the classical quasi-Newton BFGS method.

4.1 Newton's Method and Quasi-Newton Methods

Since the 2D and 3D CVT functions are almost always C^2, we may consider using Newton's method for nonlinear optimization [Nocedal and Wright 2006] to compute a local minimizer of these functions. Newton's method solves the linear system of equations $\mathbf{H}\,\delta_{\mathbf{X}} = -\nabla F$ to determine the search step $\delta_{\mathbf{X}}$ in each iteration, where $\mathbf{H}$ is the Hessian of the CVT function F. We note that the components of the gradient ∇F are given by $\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i\,(\mathbf{x}_i - \mathbf{c}_i)$ (cf. Equation (2)); the explicit formulae for the second-order derivatives, which are the elements of the Hessian, are given in Iri et al. [1984] and Asami [1991] and will be recalled in Section 4.3.

Although the Hessian matrix $\mathbf{H}$ is sparse, it is in general not positive-definite, therefore it needs to be modified to be so, in order to define a meaningful search direction. Iri et al. [1984] replaced the Hessian with the diagonal matrix $\mathbf{D}$ defined by the diagonal elements equal to $2m_i$ (cf. Equation (3)). This simplification in fact leads exactly to Lloyd's method, with non-optimal speed of convergence.

With more sophisticated modifications, such as modified Cholesky factorization [Schnabel and Eskow 1999; Nocedal and Wright 2006], Newton's method is applicable to small and median size optimization problems. Figure 7 shows a simple example in which Newton's method is applied to computing a CVT of a square with 20 sites. Due to the C^2 smoothness of the CVT function, as expected Newton's method converges quickly in this example (after 12 iterations, $\|\nabla F(\mathbf{X})\|$ goes to 10^{-16}), as shown by the plots in Figures 7 (c) and (d).

The main problem of Newton's method is the costly computation and modification of the Hessian matrix. Therefore it is normally not recommended for large-scale problems, that is when there is a large number of sites in CVT computation. For large-scale CVT computation, the *BFGS* (Broyden-Fletcher-Goldfarb-Shanno) method [Nocedal and Wright 2006], a classical quasi-Newton method, is more suitable, since it only requires at each step, to evaluate the function value and the gradient, and only involves

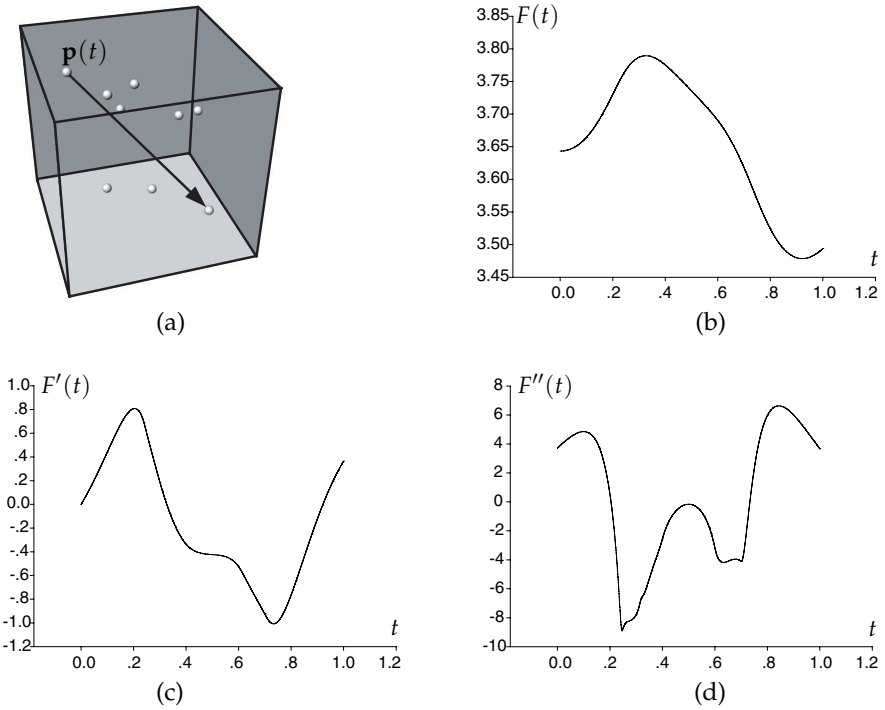


Fig. 6. Illustrations of C^2 smoothness of the 3D CVT function.

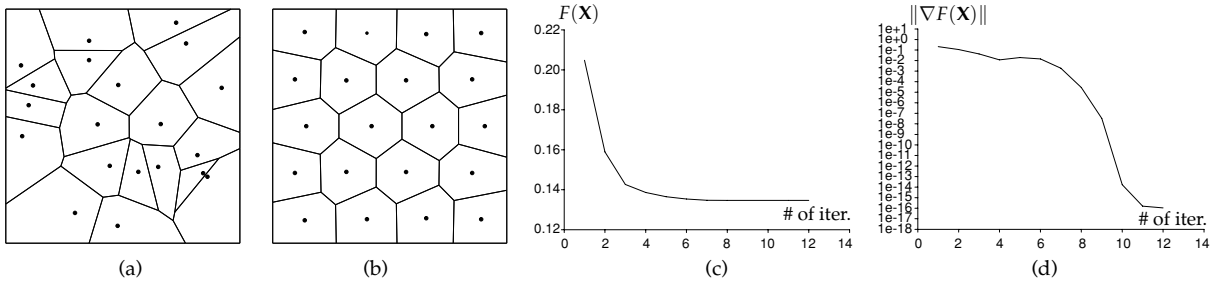


Fig. 7. (a) An initial Voronoi tessellation of 20 points; (b) The final Voronoi tessellation after optimization; (c) $F(\mathbf{X})$ against the number of iterations; (d) $\|\nabla F(\mathbf{X})\|$ against the number of iterations.

matrix-vector multiplications. However, the major drawback of the BFGS method is its large space requirement for storing the Hessian matrix, since the approximated inverse Hessian generated in each iteration is dense. In practice, this drawback is circumvented by the *limited memory BFGS method*, (*L-BFGS method*) [Nocedal 1980], to be discussed in the following.

4.2 L-BFGS Method

The L-BFGS method is similar to the classical inverse BFGS method in that the inverse Hessian is corrected by the BFGS formula. In the following we quote the concise description of the main idea of the L-BFGS method from [Liu and Nocedal 1989]:

"The user specifies the number M of BFGS corrections that are to be kept, and provides a sparse symmetric and positive definite $\widetilde{H}_0$, which approximates the inverse Hessian of f. During the first M iterations the method is identical to the BFGS method. For $k > M$, $\widetilde{H}_k$ is obtained by applying M BFGS updates to $\widetilde{H}_0$ using information from M previous iterations."

Here $f(x)$ is the objective function and g is the gradient of f. Let x_k denote the iterate at the k-th iteration and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$. The inverse BFGS formula is $\widetilde{H}_{k+1} = V_k^T \widetilde{H}_k V_k + \rho_k s_k s_k^T$, where $\rho_k = 1/(y_k^T s_k)$ and $V_k = \mathrm{I} - \rho_k y_k s_k^T$. Typically M is set as $3 \sim 20$. The explicit formula of $\widetilde{H}_{k+1}$ is:

$$
\begin{aligned}
\widetilde{H}_{k+1} \;=\; & \left(V_k^T \cdots V_{k-\widehat{M}}^T\right) \widetilde{H}_0 (V_{k-\widehat{M}} \cdots V_k) \\
+\; & \rho_{k-\widehat{M}} \left(V_k^T \cdots V_{k-\widehat{M}+1}^T\right) s_{k-\widehat{M}} s_{k-\widehat{M}}^T (V_{k-\widehat{M}+1} \cdots V_k) \\
+\; & \rho_{k-\widehat{M}+1} \left(V_k^T \cdots V_{k-\widehat{M}+2}^T\right) s_{k-\widehat{M}+1} s_{k-\widehat{M}+1}^T (V_{k-\widehat{M}+2} \cdots V_k) \\
& \vdots \\
+\; & \rho_k s_k s_k^T,
\end{aligned}
$$

where $\widehat{M} = \min\{k, M-1\}$. The product $-\widetilde{H}_{k+1}\, g_{k+1}$ is computed by term-wise product using the above expression of $\widetilde{H}_{k+1}$, with $O(Mn)$ operations, where n is the number of the sites. This is much faster than constructing $\widetilde{H}_{k+1}$ explicitly and computing $-\widetilde{H}_{k+1}\, g_{k+1}$ using matrix-vector multiplication, which would use $O(n^2)$ operations. This savings is the key to the efficiency improvement of the L-BFGS over the BFGS method.

Although only the gradient is required in the computation, the objective function needs to be C^2 to ensure the proper convergence of the BFGS and L-BFGS algorithms [Nocedal and Wright 2006; Liu and Nocedal 1989]. This requirement is indeed met by the CVT energy function in general (cf. Theorem 1 and Theorem 2 in Section 3).

What follows is the pseudo code for updating the approximate inverse Hessian in a L-BFGS iteration.

L-BFGS update at step k:

(1) initialization: $r = -g_k$;

(2) 1^{st} L-BFGS Update:
`for` $i = \min(M-1, k-1), \ldots, 0$
$\begin{cases} \gamma_i := \rho_i s_i^T r; \\ r := r - \gamma_i y_i; \end{cases}$

(3) $d_k := \widetilde{H}_k^0\, r$;

(4) 2^{nd} L-BFGS Update:
`for` $i = 0, \ldots, \min(M-1, k-1)$
$d_k := d_k + s_i(\gamma_i - \rho_i y_i^T d_k)$;

(5) update $x_{k+1} = x_k + \alpha_k d_k$ and let $k = k+1$.

A typical choice of $\widetilde{H}_k^0$ is the diagonal matrix $\frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}\mathrm{I}$ [Liu and Nocedal 1989].

4.3 Preconditioned L-BFGS Method

As we will shortly see in the experimental results, L-BFGS is significantly faster than Lloyd's method. When the Hessian is available, the convergence of L-BFGS can be further accelerated by frequently using the Hessian as the initial value of $\widetilde{H}_0$ — this is the idea of the preconditioned L-BFGS method (P-L-BFGS) by [Schlick 1992] and Jiang et al. [2004]. Normally the exact Hessian should not be used in every iteration, for otherwise the method becomes equivalent to Newton's method, which spends too much time on evaluating the Hessian.

There are two integer parameters, M and T in the P-L-BFGS method, denoted as P-L-BFGS(M, T). The parameter M means that the gradients of the previous M iterations are used to construct the approximate inverse Hessian, and T means that the initial Hessian estimate $\widetilde{H}_0$ is updated using the exact Hessian once every T iterations. Appropriate values of M and T can help achieve a balance between the accuracy of the approximate inverse Hessian and the average time-cost per iteration.

In the case of computing CVT in 2D and 3D, the exact Hessian can be constructed as follows. Let J_i denote the indices of those sites whose Voronoi cells are adjacent to Ω_i. Let $\Omega_i \bigcap \Omega_j$ be the common face shared by the Voronoi cells of $\mathbf{x}_i$ and $\mathbf{x}_j$, which is an edge in 2D or a polygon in 3D. Denote $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \ldots, x_{iN})$ and $\mathbf{x}^T = (x_1, x_2, \ldots, x_N)$. Then the second-order derivatives of the CVT function are given by the following explicit formulae [Iri et al. 1984; Asami 1991]:

$$\begin{cases} \dfrac{\partial^2 F}{\partial x_{ik}^2} &= 2m_i - \sum_{j\in \mathrm{J}_i} \int_{\Omega_i\cap\Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik}-x_k)^2\rho(\mathbf{x})\,\mathrm{d}\sigma, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{i\ell}} &= -\sum_{j\in \mathrm{J}_i} \int_{\Omega_i\cap\Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik}-x_k)(x_{i\ell}-x_\ell)\rho(\mathbf{x})\,\mathrm{d}\sigma, & k\neq\ell, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{j\ell}} &= \int_{\Omega_i\cap\Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik}-x_k)(x_{j\ell}-x_\ell)\rho(\mathbf{x})\,\mathrm{d}\sigma, & j\in \mathrm{J}_i, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{j\ell}} &= 0, & j\neq i,\ j\notin \mathrm{J}_i. \end{cases}$$

The combined pseudo code of the L-BFGS method and the P-L-BFGS method is as follows.

Preconditioned L-BFGS (P-L-BFGS) Algorithm:

(1) set $k := 0$ and choose the values of M, T. Set `flag` := *true* to indicate the exact Hessian is required; otherwise, the exact Hessian is not required;
(2) evaluate the CVT function and compute its gradient g_k. Construct the Hessian matrix H and apply modified Cholesky algorithm to obtain positive-definite $\widehat{H}$ if `flag` = `true` and ($T = 0$ or $k \bmod T = 0$);
(3) initialization: $r := -g_k$;
(4) call the 1^{st} L-BFGS Update and obtain the updated r;
(5) update d_k: if `flag` = `true`, solve $\widehat{H}\, d_k = r$; else $d_k := \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} r$ when $k > 0$;
(6) call the 2^{nd} L-BFGS update and obtain the updated d_k;
(7) apply the line-search algorithm to find $x_{k+1} := x_k + \alpha_k d_k$;
(8) check the convergence and stop criterion. If they are satisfied, stop the algorithm;
(9) set $k := k+1$. Go to step 2.

If `flag` is set to `false` in Step (1), the P-L-BFGS method reduces to the L-BFGS method. The interested reader can find our P-L-BFGS implementation online.[1]

5. PERFORMANCE EVALUATION

In this section, we will first compare the quasi-Newton methods with Lloyd's method to demonstrate the improved efficiency of the quasi-Newton method in CVT computation. Then we will analyze the Lloyd-Newton method, a previous attempt to compute CVT with second-order optimization.

5.1 Comparison with Lloyd's Method

Since Lloyd's method is currently the one most commonly used for computing CVT, we will compare it with our new methods based on the quasi-Newton methods. We will show that both the L-BFGS and P-L-BFGS methods are significantly faster than Lloyd's method.

We have implemented Newton's method, the L-BFGS method, and the P-L-BFGS method. Our tests were run on a desktop computer with a 2.33GHz Intel Xeon CPU and 4GB RAM in Windows XP 64-bit system and our C++ implementation uses CGAL [Fabri 2001] to compute Voronoi diagrams via Delaunay triangulation with hierarchy and QHULL [Barber et al. 1996] to obtain the boundary cells via half-space computation. To avoid handling the boundary constraints in our P-L-BFGS framework explicitly, we use step size

[1] http://www.loria.fr/∼liuyang/software/HLBFGS/.

control to reduce the increment of the sites if the P-L-BFGS iteration moves them outside the domain.

Our line-search routine in L-BFGS and P-L-BFGS is from More and Thuente [1994]. We set the maximum number of iterations to be 1000 and the iteration stops when $\|\nabla F(\mathbf{X})\| \leq 10^{-7}$ or as soon as there is no sufficient decrease during the line-search phase. The numerical integration method over simplices from Genz and Cools [2003] is used when $\rho(\mathbf{x}) \not\equiv$ `constant`. The modified Cholesky algorithm for Hessian correction is from Lin and Moré [1999].

In all the methods that we compare, a CVT function call is invoked in each iteration to construct the Voronoi diagram of the current sites for computing the CVT energy value and gradient. It might be invoked multiple times in each iteration due to line-search, such as in the P-L-BFGS method. Therefore, the total number of CVT function calls is more relevant than the total number of iterations when considering the total computation time of a particular method.

Example 3. 2D CVT with density $\rho(\mathbf{x}) \equiv 1$: The 2D domain Ω is a regular octagon with the bounding box $[-2, 2] \times [-2, 2]$. We sample 2000 points randomly in Ω as the initial sites (see Figure 8(a)). Table I provides detailed information of all the tested methods.

Before reaching the stopping criterion $\|\nabla F(\mathbf{X})\| \leq 10^{-7}$, Lloyd's method has used 1000 iterations and was terminated with a CVT energy higher than the values produced by the other methods. All the other methods have reached the stopping criterion $\|\nabla F(\mathbf{X})\| \leq 10^{-7}$ with less than 1000 iterations.

Table I shows that P-L-BFGS(20,20) is more efficient than P-L-BFGS(20,10) and P-L-BFGS(200,20), as also attested in Figures 8(f) and (i). Although the number of iterations used by P-L-BFGS(20,10) and Newton's method is less than that of P-L-BFGS(20,20), their more frequent use of costly Hessian computation and modification slows them down. Therefore we choose P-L-BFGS(20,20) as the representative P-L-BFGS method for further comparisons with the other methods, as shown in Figures 8(g), (h), (j), and (k) in terms of the CVT energy and the gradient norm with respect of the number of iterations and the computation time. Similarly, since L-BFGS(7) is more efficient than L-BFGS(20), we show only L-BFGS(7) in these figures.

Figures 8(b)–(e) show particular results computed by the P-L-BFGS(20,20) and Lloyd's method, starting from the same initialization. It is easy to see that P-L-BFGS(20,20) yields better quality with the same computation time by comparing the number of non-hexagonal Voronoi cells (with more or less than six sides, shown in blue). This visual evaluation makes sense due to Gersho [1979] observation (see the discussion in Section 1).

Because the CVT function has many local minimizers, different optimization methods may find different minimizers even if they start with the same initial set of sites. Therefore the convergence rates of different methods are better illustrated by the plots of their gradient norms $\|\nabla F(\mathbf{X})\|$ (see Figures 8(h) and (k)), than by the plots of the CVT energy alone.

The experimental data indicates that both L-BFGS and P-L-BFGS are significantly faster than Lloyd's method. We also note from Figure 8(j) that, as expected, Newton's method is not much more efficient than Lloyd's method, due to its costly computation and modification of the Hessian matrix in every iteration.

Example 4. 2D CVT with density $\rho(\mathbf{x}) \not\equiv$ `constant`: The 2D domain Ω is a regular hexagon with the bounding box $[-2, 2] \times [-1.732, 1.732]$. We sample 2000 points randomly in Ω as the initial sites (see Figure 9) according to the density function $\rho(\mathbf{x}) = e^{-20(x^2+y^2)} + 0.05 \sin^2(\pi x) \sin^2(\pi y)$. Figure 9 shows the comparisons of different methods in terms of $F(\mathbf{X})$ and $\|\nabla F(\mathbf{X})\|$. Table II shows other statistics for all the tested methods.

In this example we see again that L-BFGS and P-L-BFGS yield the best performance as compared to Lloyd's method and Newton's method.

Example 5. 3D CVT with density $\rho(\mathbf{x}) \equiv 1$: 2000 sites are sampled in a convex polyhedron with the bounding box $[-3.236, 3.236] \times [-2.427, 2.427] \times [-1.942, 1.942]$ (see Figure 10(a)). Figures 10(f) to (k) show the comparisons of different methods in terms $F(\mathbf{X})$ and $\|\nabla F(\mathbf{X})\|$. Table III shows other information of all the tested methods, in the same format as in the previous two test examples. Again we conclude from the data that the L-BFGS and P-L-BFGS are more efficient than the other methods, and that all these methods are more efficient than Lloyd's relaxation.

5.2 Lloyd-Newton Method

The Lloyd-Newton method [Du and Emelianenko 2006] is a previous attempt at speeding up CVT computation. The Lloyd-Newton method uses Newton's root-finding method to solve the system of nonlinear equations given by Equation (4). Note that Newton's root-finding method used in the Lloyd-Newton's method needs to be distinguished from the Newton-like minimization used by our method. We will show that Lloyd-Newton is essentially equivalent to the Gauss-Newton method applied to minimizing an objective function different from the CVT energy function in Equation (1). In contrast, the quasi-Newton method we have considered so far for CVT computation directly minimizes the CVT energy function. Hence, the two methods work by quite different principles. We will see that the Lloyd-Newton method often fails to find a stable CVT, unless many Lloyd iterations are first used to preinitialize the algorithm, but this makes the overall method behave like Lloyd's relaxation, which is inefficient.

Equation (4) can be rewritten as $\mathbf{G}(\mathbf{X}) \equiv \mathbf{X} - \mathbf{T}(\mathbf{X}) = 0$. Here, $\mathbf{T}(\mathbf{X}) = (\mathbf{c}_i(\mathbf{X}))_{i=1}^n$ is termed the *Lloyd map*, which maps the sites to their corresponding centroids. Newton's method for solving a system of nonlinear equations, used in Du and Emelianenko [2006], operates by using a first-order approximation of $\mathbf{G}$ around $\mathbf{X}$:

$$\mathbf{G}(\mathbf{X} + \delta_{\mathbf{X}}) \simeq G(\mathbf{X}) + J_{\mathbf{G}}\, \delta_{\mathbf{X}},$$

where $\delta_{\mathbf{X}}$ denotes a small displacement around $\mathbf{X}$ and $J_{\mathbf{G}} = [\frac{\partial G_i}{\partial x_j}]_{i,j}$ is the Jacobian matrix of $\mathbf{G}$. Then it repeatedly applies the following iteration until some convergence criterion is reached :

$$\begin{aligned} \text{solve} \quad J_{\mathbf{G}}\, \delta_{\mathbf{X}} &= -G(\mathbf{X}) \\ \mathbf{X} &\leftarrow \mathbf{X} + \delta_{\mathbf{X}}, \end{aligned} \tag{5}$$

where $G(\mathbf{X}) = \mathbf{X} - \mathbf{T}(\mathbf{X})$ and $J_{\mathbf{G}} = (\mathbf{I} - J_{\mathbf{T}})$.

We will show that the Lloyd-Newton method can be recast in a variational formulation, as it is essentially equivalent to applying the Gauss-Newton method to a least squares problem that minimizes the function:

$$\widehat{F} = \frac{1}{2} \sum \|\mathbf{x}_i - \mathbf{c}_i\|^2 = \frac{1}{2} \|G(\mathbf{X})\|^2. \tag{6}$$

The gradient $\nabla \widehat{F}$ and Hessian $\mathbf{H}'$ of $\widehat{F}$ are given by:

$$\begin{aligned} \nabla \widehat{F} &= J_{\mathbf{G}}^T\, \mathbf{G}, \\ \mathbf{H}' &= J_{\mathbf{G}}^T\, J_{\mathbf{G}} + \mathbf{Q} \simeq J_{\mathbf{G}}^T\, J_{\mathbf{G}}, \end{aligned}$$

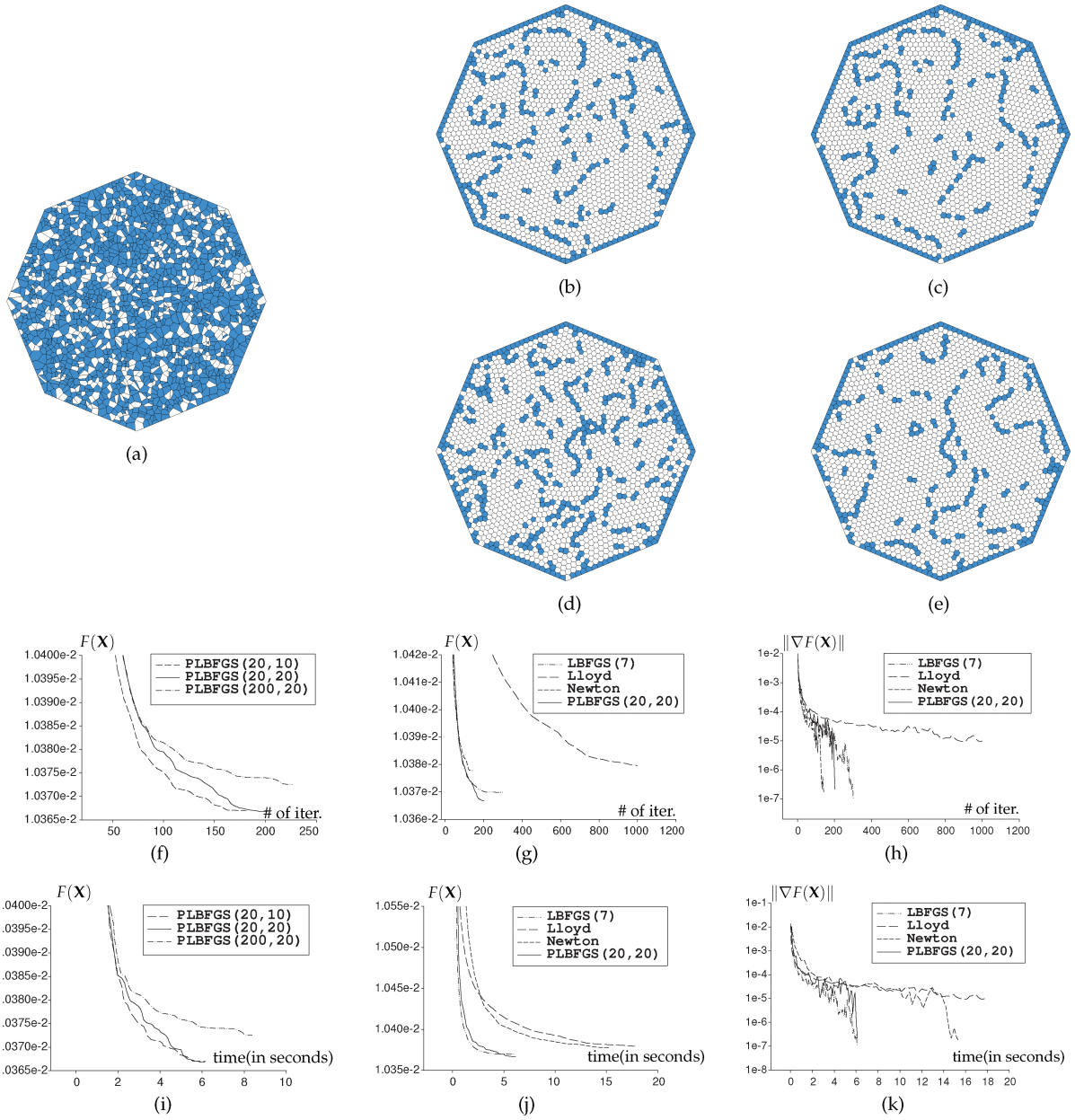


Fig. 8. *Example* 3. Non-hexagonal cells (with more or less than six edges) are marked in blue. (a) The initial Voronoi tessellation; (b) the result after 100 iterations by P-L-BFGS(20, 20); (c) the result after 203 iterations by P-L-BFGS(20, 20); (d) the result after 153 Lloyd's iterations (which takes the same time as (b)); (e) the result after 1000 Lloyd's iterations; (f) $F(\mathbf{X})$ against the number of iterations of the P-L-BFGS methods; (g) $F(\mathbf{X})$ against the number of iterations of some selected methods; (h) $\|\nabla F(\mathbf{X})\|$ against the number of iterations of some selected methods; (i) $F(\mathbf{X})$ against the computation time of the P-L-BFGS methods; (j) $F(\mathbf{X})$ against the computation time of some selected methods; (k) $\|\nabla F(\mathbf{X})\|$ against the computation time of some selected methods.

where $\mathbf{Q}$ contains some second-order terms that are neglected by the Gauss-Newton method. This gives the following expression to compute the step vector $\delta_{\mathbf{X}}$ in each Gauss-Newton iteration:

$$J_{\mathbf{G}}^T J_{\mathbf{G}}\, \delta_{\mathbf{X}} \;=\; -J_{\mathbf{G}}^T\, G(\mathbf{X}).$$

Since the system has as many equations as unknowns, $J_{\mathbf{G}}$ is a square matrix. When $J_{\mathbf{G}}$ is nonsingular, $J_{\mathbf{G}}^T$ can be removed from both sides, and this produces exactly the same update scheme as in the Lloyd-Newton method (cf. Equation (5)).

There are two aspects that make it difficult to compute CVT by minimizing $\widehat{F} = \frac{1}{2}\|\mathbf{X} - \mathbf{T}(\mathbf{X})\|^2 = \frac{1}{2}\|G(\mathbf{X})\|^2$ using a Newton-like method. First, the Lloyd map $\mathbf{T}(\mathbf{X})$ is only a C^1 function. Therefore, the quadratic convergence cannot be expected in general by either applying Newton's method to solving $G(\mathbf{X}) = 0$ or by applying the Gauss-Newton method to minimizing $\widehat{F} = \frac{1}{2}\|G(\mathbf{X})\|^2$. Second,

Table I. Comparisons in Example 3. # *of iter.* is the Number of Iterations and # *of CVT-func-calls* is the Total Number of CVT Function Calls

Method	# of iter.	# of CVT-func-calls	Time (seconds)	$F(\mathbf{X})_{\texttt{final}}$	$\|\nabla F(\mathbf{X})\|_{\texttt{final}}$	Time (seconds) per iter.
L-BFGS(7)	304	320	6.14	1.0369816e-2	9.462e-8	0.020
L-BFGS(20)	451	490	9.75	1.0363602e-2	8.405e-8	0.022
Lloyd	1000	1000	17.75	1.0379564e-2	9.756e-6	0.018
Newton	144	263	15.44	1.0377779e-2	9.705e-8	0.107
P-L-BFGS(20,10)	182	239	6.17	1.0366966e-2	8.408e-8	0.034
P-L-BFGS(20,20)	203	264	6.14	1.0366771e-2	8.916e-8	0.030
P-L-BFGS(200,20)	228	280	8.43	1.0372586e-2	7.586e-8	0.037

when such a method converges, it may well converge to a solution of $G(\mathbf{X}) = 0$ that is an unstable critical point of the CVT energy function F.

The Lloyd-Newton method is a hybrid method consisting of Lloyd's method and Newton's root-finding method. In the Lloyd-Newton method, to increase the chance of converging to a stable CVT, a large number of Lloyd's iterations are first used to preinitialize the input before Newton's root finder is invoked, hence the name *Lloyd-Newton*.

To compare it with other methods, we implemented the Lloyd-Newton method according to the following flow provided in Du and Emelianenko [2006]:

(1) Perform Lloyd's algorithm. If the difference between the centroids and the sites is larger than ϵ, the sites are updated by centroids, goto 1; otherwise set stepsize to 1, goto 2.
(2) Perform Newton's method to solve the system of nonlinear equations in Equation (4).
(3) Let n be the number of computed sites that are outside of the domain Ω. If $n = 0$, update sites; if $n = 1$, halve the stepsize, goto 3; if $n > 1$, goto 1.

The performance of the Lloyd-Newton algorithm is sensitive to the parameter ϵ, which decides when one should switch from Lloyd's iteration to Newton's iteration. If ϵ is too small, the Lloyd-Newton method is equivalent to Lloyd's method. If ϵ is not very small (e.g., $\epsilon > 10^{-5}$), the Lloyd-Newton algorithm cannot benefit from the robustness of Lloyd's method and is therefore more likely to converge to an unstable CVT.

Incidentally, there is a minor robustness issue in the above procedure for the Lloyd-Newton method as provided in [Du and Emelianenko 2006]. In step (1), if the difference between the centroids and the sites is smaller than ϵ, then we go to (2). In step (2), if two or more updated sites computed by Newton's method are outside the domain Ω, then we go back to step (1). Therefore, when both conditions are satisfied simultaneously, the Lloyd-Newton method is stuck in an infinite loop. Such a case is referred to as a *failure* in the following. Although we do not undertake to fix this defect, we note that the issue is not essential to the idea of the Lloyd-Newton method and may be fixed by some modifications in the implementation flow without much difficulty.

We now provide experimental results about the Lloyd-Newton method. For each different value of the parameter ϵ, we collected statistics by performing 100 tests of computing CVT in the square $[-1, 1]^2$ with 100 sites. In each test, the 100 sites are initialized with a uniform random distribution. For each value of ϵ, we recorded the number of Lloyd's iterations needed, the number of Newton's iterations needed, and the numbers of stable CVTs and unstable CVTs computed by the Lloyd-Newton method. The termination condition is that $\|\nabla F(\mathbf{X})\| \leq 0.5 \times 10^{-15}$. The results are shown in Table IV.

We see that, when $\epsilon = 10^{-6}$, even with more than 300 Lloyd iterations, the Lloyd-Newton method still fails to compute a stable CVT most of the time—62 out of 100 times. In contrast, running L-BFGS 100 times with this example using the same initialization and termination conditions, we obtained stable CVTs in all 100 tests. The average number of L-BFGS iterations used in each test is 116.98, costing about one third of the time used by the Lloyd-Newton method. The P-L-BFGS has a similar level of robustness and efficiency as the L-BFGS method with this example.

To conclude, when the Jacobian matrix $J_{\mathbf{G}}$ is nonsingular, the Lloyd-Newton method is equivalent to minimizing the energy $\widehat{F} = \frac{1}{2}\sum \|\mathbf{x}_i - \mathbf{c}_i\|^2$ with the Gauss-Newton method. This approach is not optimally efficient because it requires first using a large number of Lloyd iterations to provide a very good initial value, making the computation inefficient. Without such an expensive initialization, it will suffer from a lack of robustness as it will often get stuck in an unstable CVT.

6. FAST CVT COMPUTATION ON MESH SURFACES

In this section we will test the efficiency improvement brought about by the quasi-Newton method for CVT computation on a surface. CVT can be extended to manifolds for several applications, including surface remeshing [Alliez et al. 2003]. It is natural to use the geodesic metric in the CVT formulation [Kunze et al. 1997; Leibon and Letscher 2000; Peyré and Cohen 2004], but the computational cost would then be quite high. Du et al. [2003] propose the constrained centroidal voronoi diagram (CCVT), in which the geodesic metric on a surface is approximated by the Euclidean metric in 3D. We adopt this formulation of CVT on a surface in the following experiments.

6.1 Constrained Centroidal Voronoi Tessellation

Denote a given smooth surface by $\Omega \subset \mathbb{R}^3$. Let $\mathbf{X}$ denote the distinct sites $(\mathbf{x}_i)_{i=1}^n$ in Ω, that is, $\forall i \neq j, \mathbf{x}_i \neq \mathbf{x}_j$. The Voronoi region of the site $\mathbf{x}_k$ is defined as $\Omega_k = \{\mathbf{x} \in \Omega \mid \|\mathbf{x} - \mathbf{x}_k\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \forall j \neq k\}$. The *constrained centroid* of Ω_k is defined as:

$$\mathbf{c}_k = \arg\min_{\mathbf{y}\in\Omega} \int_{\mathbf{x}\in\Omega_k} \rho(\mathbf{x})\|\mathbf{y} - \mathbf{x}\|^2 \, \mathrm{d}\sigma,$$

which exists but may not be unique [Du et al. 2003]. For an ordered set of sites $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ on Ω, its CCVT energy function is defined as:

$$F(\mathbf{X}) = \sum_{i=1}^{N} \int_{\mathbf{x}\in\Omega_i} \rho(\mathbf{x})\|\mathbf{x} - \mathbf{x}_i\|^2 \, \mathrm{d}\sigma.$$

Du et al. [2003] show that Lloyd's method works well for CCVT.

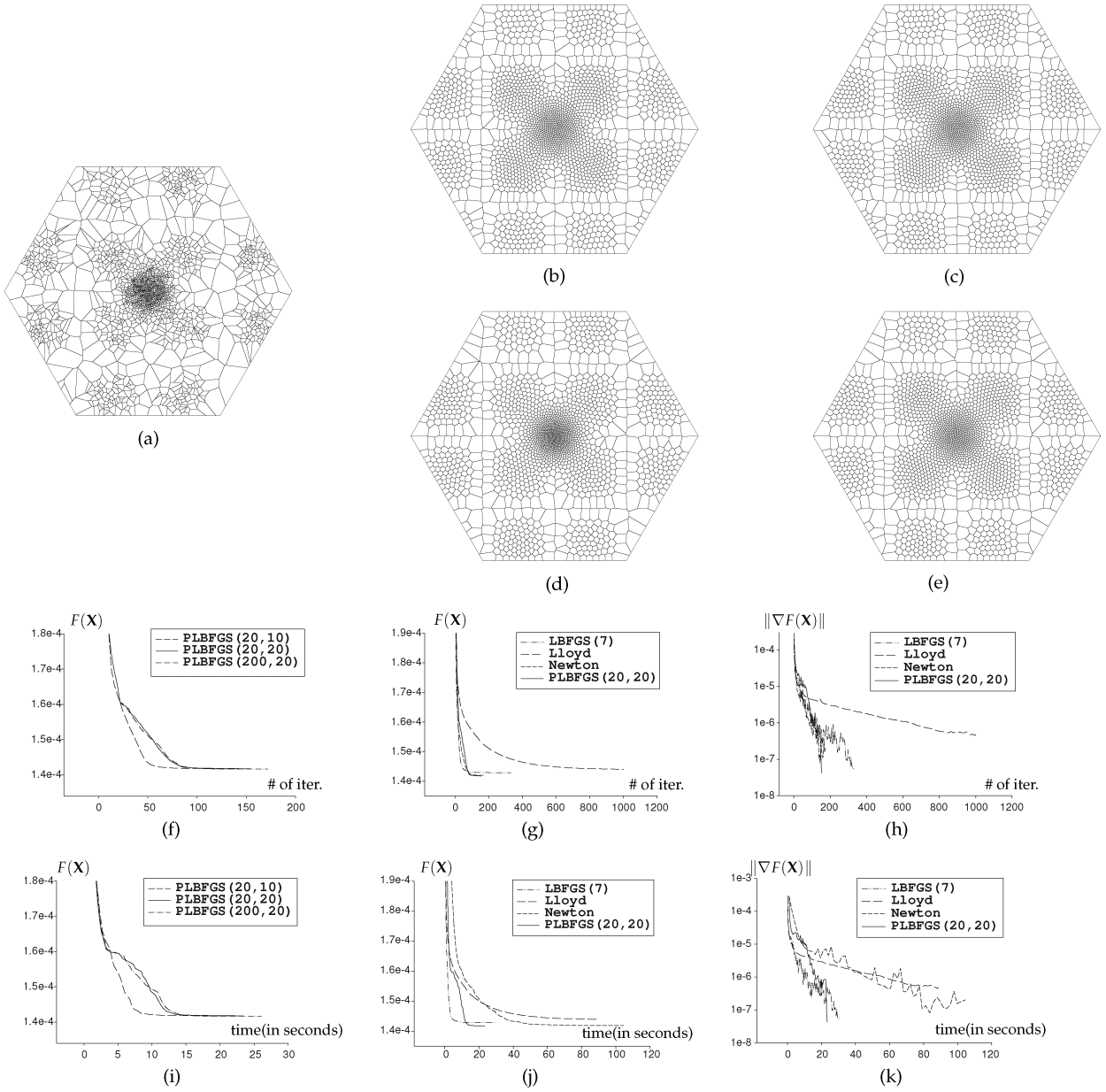


Fig. 9. *Example* 4. (a) The initial Voronoi tessellation with the sites distributed according to the density function; (b) the result after 100 iterations by P-L-BFGS(20,20); (c) the result after 154 iterations by P-L-BFGS(20,20); (d) the result after 178 Lloyd's iterations (which takes the same time as (b)); (e) the result after 1000 Lloyd's iterations; (f) $F(\mathbf{X})$ against the number of iterations of the P-L-BFGS methods; (g) $F(\mathbf{X})$ against the number of iterations of some selected methods; (h) $\|\nabla F(\mathbf{X})\|$ against the number of iterations of some selected methods; (i) $F(\mathbf{X})$ against the computation time of the P-L-BFGS methods; (j) $F(\mathbf{X})$ against the computation time of some selected methods; (k) $\|\nabla F(\mathbf{X})\|$ against the computation time of some selected methods.

On a surface, $F(\mathbf{X})$ is also defined in a piecewise manner—its expression takes different expressions for combinatorially different Voronoi tessellations, and the change in this expression occurs when there is a structural change of the Voronoi tessellation. From the discussion on the smoothness of CVT energy in Section 3, it can be argued that the constrained CVT energy function is also C^2 for most situations encountered in optimization.

For computing CCVT, we need the projection of the gradient of CVT onto the tangent plane of the surface Ω, which can be derived from the gradient component $\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i(\mathbf{x}_i - \mathbf{c}_i)$ of $F(\mathbf{X})$ as follows:

$$\left.\frac{\partial F}{\partial \mathbf{x}_i}\right|_{\Omega} = \frac{\partial F}{\partial \mathbf{x}_i} - \left(\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{N}(\mathbf{x}_i)\right) \cdot \mathbf{N}(\mathbf{x}_i), \tag{7}$$

where $\mathbf{N}(\mathbf{x}_i)$ is the unit normal vector of the surface Ω at $\mathbf{x}_i$.

This gradient function is used in the L-BFGS method for computing CCVT on a surface. Note that the updated sites $\mathbf{x}_i$ in every

Table II. Comparisons in *Example* 4

Method	# of iter.	# of CVT-func-calls	Time (seconds)	$F(\mathbf{X})_{\texttt{final}}$	$\|\nabla F(\mathbf{X})\|_{\texttt{final}}$	Time (seconds) per iter.
L-BFGS(7)	332	338	30.39	1.4287259e-4	4.406e-8	0.092
L-BFGS(20)	270	272	24.58	1.4314150e-4	4.406e-8	0.091
Lloyd	1000	1000	88.72	1.4401771e-4	4.778e-7	0.089
Newton	177	304	105.52	1.4192786e-4	3.491e-8	0.596
P-L-BFGS(20,10)	140	168	21.47	1.4173879e-4	3.489e-8	0.153
P-L-BFGS(20,20)	154	221	23.37	1.4175610e-4	4.254e-8	0.152
P-L-BFGS(200,20)	172	229	26.06	1.4162808e-4	4.223e-8	0.152

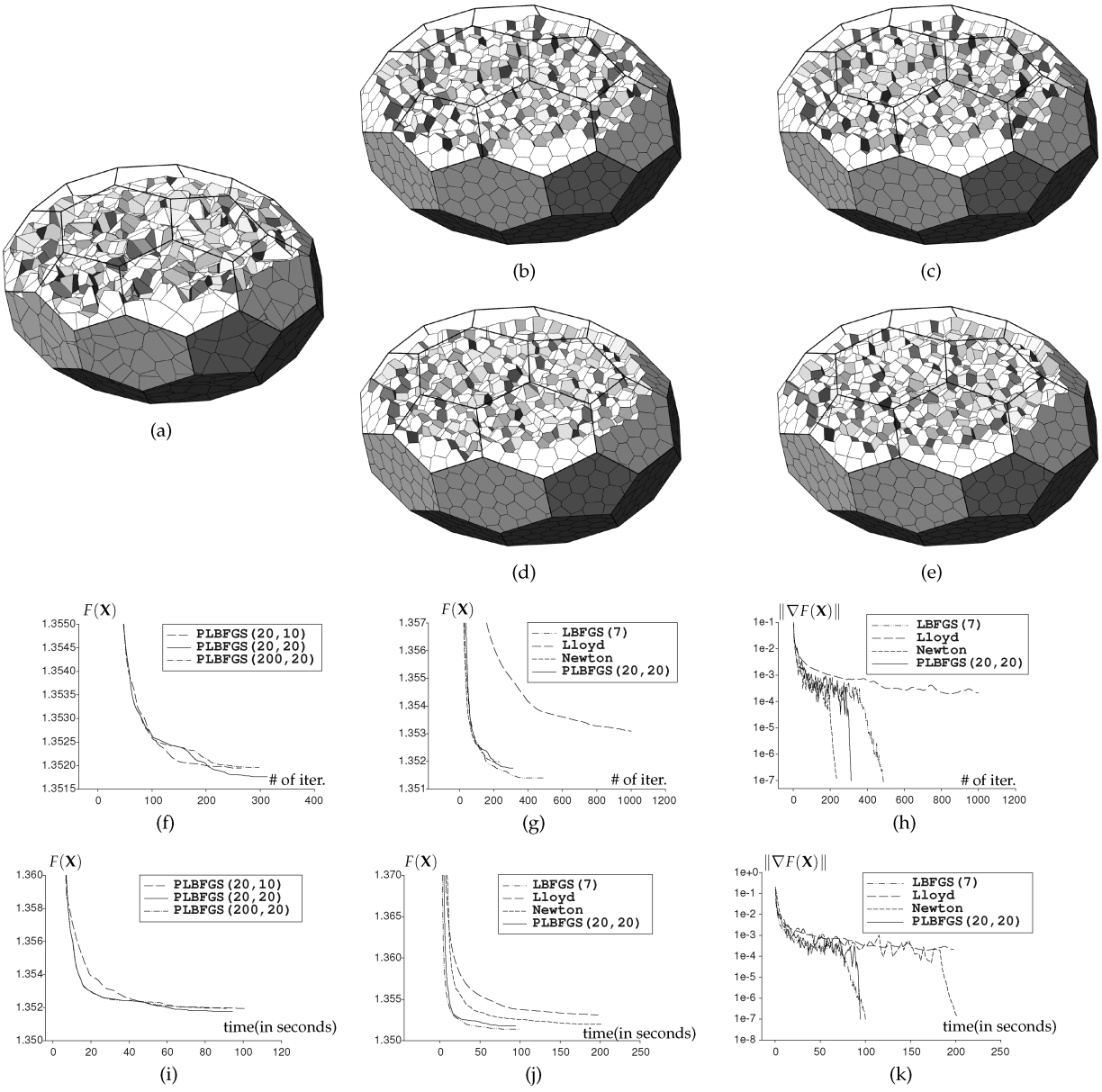


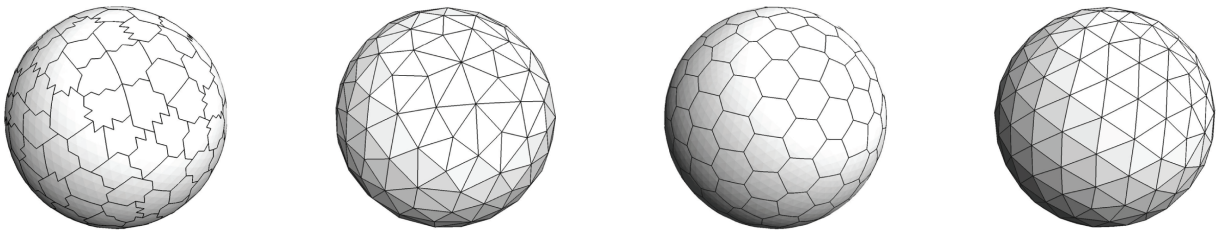
Fig. 10. *Example* 5. (a) The initial Voronoi tessellation; (b) the result after 100 iterations by P-L-BFGS(20,20); (c) the result after 313 iterations by P-L-BFGS(20,20); (d) the result after 129 Lloyd's iterations (which takes the same time as (b)); (e) the result after 1000 Lloyd's iterations; (f) $F(\mathbf{X})$ against the number of iterations of the P-L-BFGS methods; (g) $F(\mathbf{X})$ against the number of iterations of some selected methods; (h) $\|\nabla F(\mathbf{X})\|$ against the number of iterations of some selected methods; (i) $F(\mathbf{X})$ against the computation time of the P-L-BFGS methods; (j) $F(\mathbf{X})$ against the computation time of some selected methods; (k) $\|\nabla F(\mathbf{X})\|$ against the computation time of some selected methods.

Table III. Comparisons in *Example* 5

Method	# of iter.	# of CVT-func-calls	Time (seconds)	$F(\mathbf{X})_{\texttt{final}}$	$\|\nabla F(\mathbf{X})\|_{\texttt{final}}$	Time (seconds) per iter.
L-BFGS(7)	487	498	100.17	1.3513974	8.850e-8	0.206
L-BFGS(20)	645	665	136.25	1.3511813	9.971e-8	0.211
Lloyd	1000	1000	197.53	1.3530904	2.144e-4	0.198
Newton	236	493	202.39	1.3519816	9.864e-8	0.858
P-L-BFGS(20,10)	266	439	102.00	1.3519459	6.684e-8	0.383
P-L-BFGS(20,20)	313	428	94.33	1.3517604	9.865e-8	0.301
P-L-BFGS(200,20)	299	400	91.36	1.3519590	7.620e-8	0.306

Table IV.
of *failures* is the number of tests (out of a total of the 100 tests) in which the Lloyd-Newton Method fails to find a critical point of F, that is, it falls into an infinite loop; # of *unstable CVTs* is the number of times it returns an unstable CVT; # of *stable CVTs* is the number of times it returns a stable CVT; # of *Lloyd's iter*. is the average number of Lloyd iterations used in each test; # of *Newton's iter*. is the average number of Newton's iterations used in each test

ϵ	# of Failures	# of Unstable CVTs	# of Stable CVTs	# of Lloyd's iter.	# of Newton's iter.
10^{-5}	96	4	0	124.25	28.25
10^{-6}	31	31	38	317.72	40.86
10^{-7}	3	17	80	388.59	25.41
10^{-8}	3	6	91	481.98	18.22
10^{-9}	2	4	94	622.11	11.72
10^{-10}	0	1	99	776.98	5.13



(a) (b) (c) (d)

Fig. 11. Voronoi diagrams and their dual. (a) Voronoi diagram by face-clustering CVT; (b) Dual triangulation by face-clustering CVT; (c) Voronoi diagram by face-splitting CVT; (d) Dual triangulation by face-splitting CVT. Clearly, the result via face-splitting is more regular.

iteration need to be projected to their nearest points on the surface, as required by the formulation of CCVT.

6.2 Computing a Voronoi Diagram on a Mesh Surface

The most time-consuming part of CVT computation is to determine the Voronoi region for each site. Face clustering by the flood-fill algorithm [Cohen-Steiner et al. 2004] and boundary edge updating method [Valette and Chassery 2004; Valette et al. 2008] have proven extremely fast in approximating Voronoi diagrams. Valette and Chassery [2004] employed face clustering with approximated CCVT energy using Lloyd's algorithm. These methods are discrete in the sense that each triangle face of the input mesh is the smallest primitive that is not further subdivided. Therefore, these methods terminate quickly but often yield a suboptimal solution since the result is highly sensitive to the quality of the input mesh.

In contrast, we treat a mesh surface as a piecewise continuous surface and split the triangle faces to accurately evaluate the CVT energy when computing Voronoi cells. This leads to better remeshing quality as shown by the following example. Consider a sphere-shaped mesh surface with 2562 vertices and 5120 faces. The discrete CVT of 200 sites generated by the method of Valette and Chassery [2004] using Lloyd's iteration and its dual triangle mesh are shown in Figures 11(a) and (b). The CCVT of 200 sites generated by Lloyd's method with face splitting and its dual triangle mesh are shown in Figures 11(c) and (d). Clearly, splitting the triangles results in a mesh with more regular triangles and more regular distribution of vertex degrees. The method by Valette and Chassery [2004] stops after 33 iterations, while Lloyd's method with face splitting stops after 70 iterations ($\|\delta_{\mathbf{X}}\| < 0.5 \times 10^{-4}$). Indeed, in this example the former is much faster—it takes only 0.95 seconds, while Lloyd's method based on face splitting takes 4.33 seconds.

Careful implementation is needed to split triangle faces of a mesh surface by the boundaries of Voronoi cells. We skip the details of this geometric computation due to space limitations, since they are not so directly related to our main contribution in the present article. The geometric details are given in Yan et al. [2009].

6.3 Comparison of Lloyd's Method and L-BFGS for CCVT

We present two test examples to demonstrate the speedup by the L-BFGS method over Lloyd's method. The two input surfaces are the mesh surfaces of two 3D models, Rocker and Lion. The

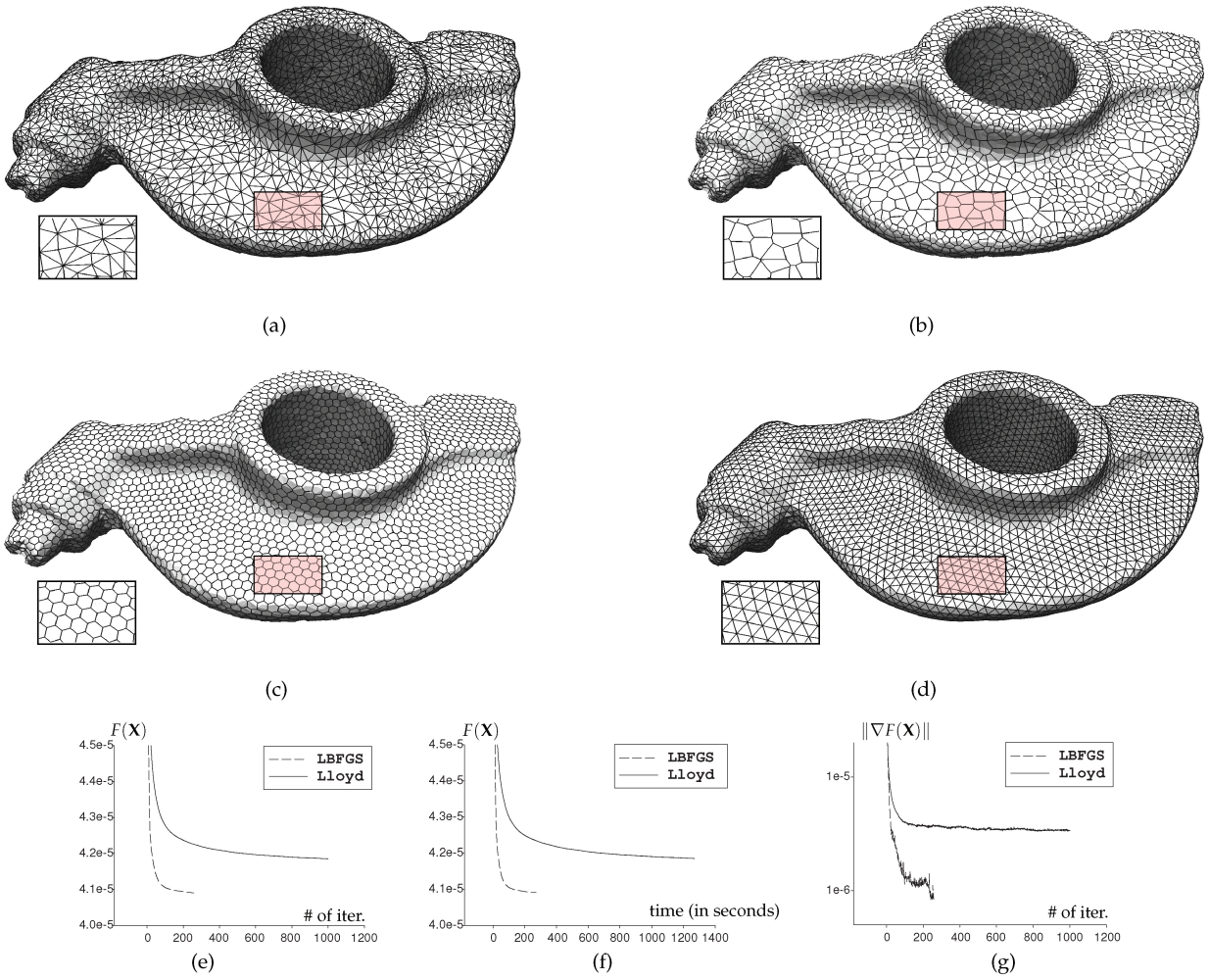


Fig. 12. Rocker model (the number of faces: 11,316; the number of vertices: 5658; the number of sites: 5658; size of bounding box: $0.49 \times 0.29 \times 0.96$). (a) The input mesh; (b) The initial Voronoi diagram; (c) CCVT computed by the L-BFGS method (the number of iterations: 284; $F(\mathbf{X})_{final} = 4.0909\text{e-}5$, $\|\nabla F(\mathbf{X})\|_{final} = 8.3197\text{e-}7$; computation time: 272.021 seconds); (d) The triangle mesh dual to CCVT; (e) $F(\mathbf{X})$ against the number of iterations; (f) $F(\mathbf{X})$ against the computation time (in seconds); (g) $\|\nabla F(\mathbf{X})\|$ against the number of iterations.

vertices of the input meshes are used as the initial sites. The input mesh, the initial Voronoi tessellation, the Voronoi diagram computed by L-BFGS, and the dual triangle mesh are shown as in Figures 12 and 13, respectively. We use $M = 7$ for L-BFGS (see the discussion in Section 4.1). Figures 12(e) and (f) (Figures 13(e) and (f), respectively) show the CVT energy computed by Lloyd's method and the L-BFGS method, against the number of iterations and computation time. The gradient curve is shown in Figure 12(g) (Figure 13(g), respectively). From these data we see that the L-BFGS method is significantly faster than Lloyd's method. For the Rocker model, for instance, Lloyd's method takes 1000 iterations to reduce the CVT energy to 4.18544×10^{-5} (with gradient norm $\|\nabla F(\mathbf{X})\| = 3.37589 \times 10^{-6}$), taking 1268.25 seconds. In comparison, the L-BFGS method reaches about the same CVT energy value ($F(\mathbf{X}) = 4.18519 \times 10^{-5}$ with gradient norm $\|\nabla F(\mathbf{X})\| = 2.82868 \times 10^{-6}$) using 36 iterations, taking 34.51 seconds.

For the Lion model, Lloyd's method takes 1000 iterations to reduce the CVT energy to 2.29266×10^{-5} (with gradient norm $\|\nabla F(\mathbf{X})\| = 2.4994 \times 10^{-6}$), taking 6112.38 seconds; in comparison, the L-BFGS method reaches about the same CVT energy value ($F(\mathbf{X}) = 2.29232 \times 10^{-5}$ with gradient norm $\|\nabla F(\mathbf{X})\| = 1.58981 \times 10^{-6}$, using 44 iterations, taking 216.49 seconds.

7. CONCLUSION AND DISCUSSION

We have introduced a new numerical framework for CVT computation in 2D, 3D, and on mesh surfaces. We have shown the C^2 smoothness of the CVT energy function in a convex domain, as well as the practical importance of this result by demonstrating the high efficiency of quasi-Newton methods for computing CVT in 2D, 3D, and on mesh surfaces. These quasi-Newton methods are significantly faster than Lloyd's method and more robust than the Lloyd-Newton method. All the applications of CVT, ranging from meshing, sampling theory, compression, to optimization, are expected to benefit from this acceleration by simply replacing Lloyd's method with our method.

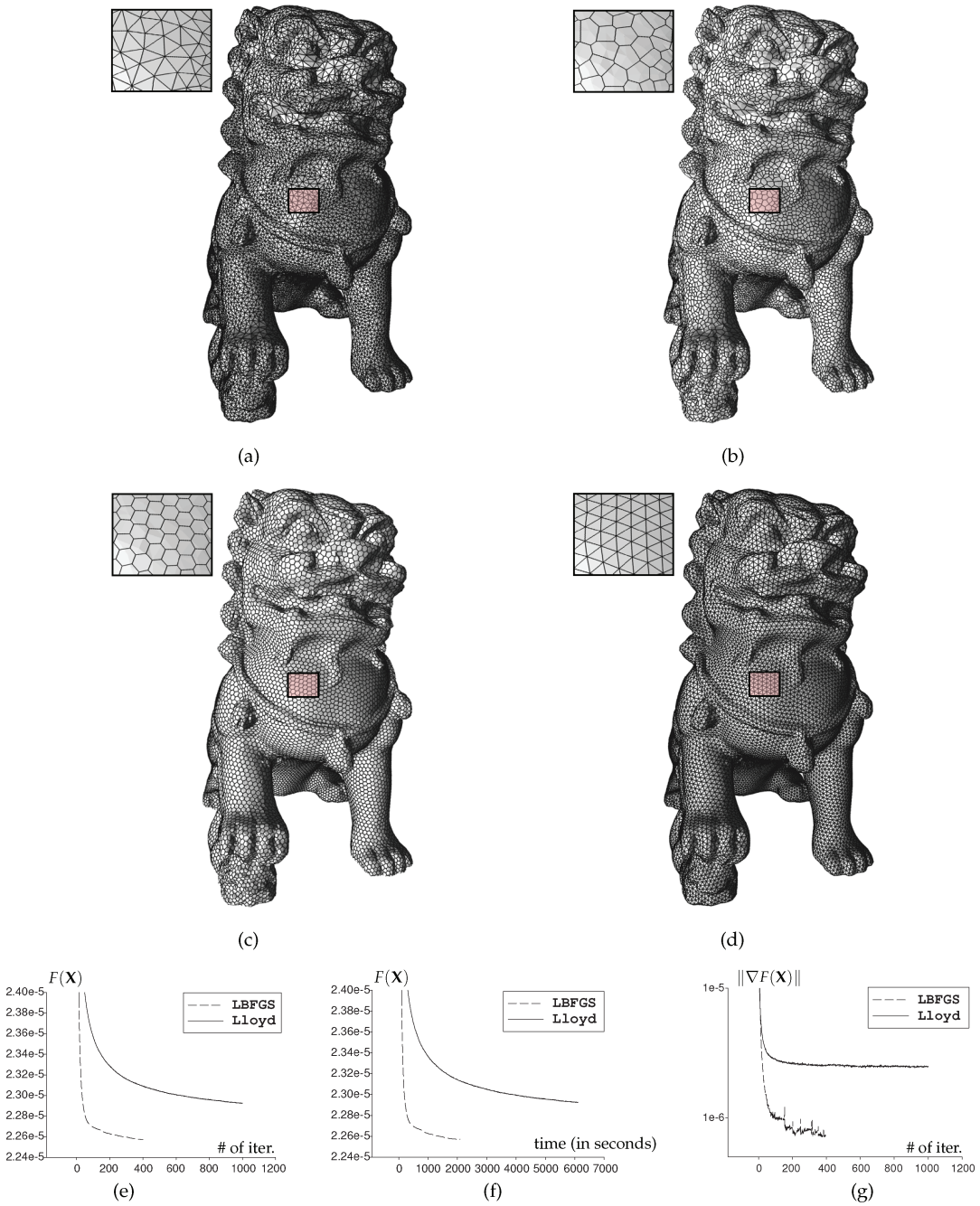


Fig. 13. Lion model (the number of faces: 50,000; the number of vertices: 25,002; the number of sites: 25,002; size of bounding box: $0.47 \times 0.85 \times 0.83$). (a) The input mesh; (b) The initial Voronoi diagram; (c) CCVT computed by the L-BFGS method (the number of iterations: 447; $F(\mathbf{X})_{final} = 2.2572\text{e-}5$, $\|\nabla F(\mathbf{X})\|_{final} = 7.3149\text{e-}7$; computation time: 2,090.95 seconds); (d) the triangle mesh dual to CCVT; (e) $F(\mathbf{X})$ against the number of iterations; (f) $F(\mathbf{X})$ against the computation time in seconds; (g) $\|\nabla F(\mathbf{X})\|$ against the number of iterations.

Computing a Voronoi tessellation is the most time consuming task in every iteration of all the methods discussed here. In practical applications, one may exploit the coherence of the Voronoi tessellations between successive iterations to speed up the computation. After a certain number of iterations, the structure of the Voronoi tessellation remains stable, or only undergoes small local modifications. Therefore, it may be possible to quickly check the validity of the triangulation inherited from the previous iteration and/or to perform local operations to update the triangulation. However, this issue does not much affect the relative comparison of the different methods we have considered, since all these methods would greatly benefit from this potential speedup.

All existing techniques, including the quasi-Newton approaches presented in this article, are only capable of computing a local minimum. We have tested the Region Teleportation approach as proposed in Cohen-Steiner et al. [2004]. Although it does sometimes help find a better local minimum, there is no theoretical justification to ensure, or strong experimental evidence to show, that it works most of the time. Because the CVT objective function has a large number of local minima, it is a challenge to devise an effective method to find the global minimizer or even a local minimizer with guarantees of optimality, for example, bounded distance to the global optimizer, or bounded difference of CVT energy from the global minimum. Such a method would be very useful for generating high-quality meshes. It would be also of great utility to empirically study the possible configuration of a globally optimal CVT in 3D, in relation to Gersho's conjecture, along the line of investigation of Du and Wang [2005b].

There are several problems to be considered in further research. One problem is to study the impact of the initialization on the speed of convergence of the optimization method. For instance, sophisticated initialization methods can significantly improve the speed of k-means [Arthur and Vassilvitskii 2007]. The same idea is probably applicable in our context. Another important problem is the generalization to the CVT framework with an arbitrary metric, with the aim of designing efficient anisotropic mesh generation algorithms. In particular, this generalized setting raises interesting questions about the smoothness of F and effective optimization techniques.

When applying CVT to meshing on a surface or a domain with boundary, one also needs to consider special sites that are constrained to lie on sharp feature curves or boundaries in order to ensure that the resulting mesh preserves the features and boundaries of the input surface. This poses the problem of defining a CVT framework that accommodates both free sites and constrained sites, and issues of efficient computation with these constraints.

We have shown that the CVT energy function is C^2 for a convex domain in 2D and 3D when the density function $\rho(\mathbf{x})$ is C^2. We conjecture that this is still true if $\rho(\mathbf{x})$ is C^0, as supported by our empirical study. It would be important to settle this conjecture, as this relaxed condition on the smoothness of the density function is assumed in many applications where both the domain and background function ρ have a piecewise-linear representation.

REFERENCES

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. *ACM Trans. Graph. 24*, 3, 617–625.

ALLIEZ, P., COLIN DE VERDIÈRE, É., DEVILLERS, O., AND ISENBURG, M. 2003. Centroidal Voronoi diagrams for isotropic surface remeshing. *Graph. Models 67*, 3, 204–231.

ARTHUR, D. AND VASSILVITSKII, S. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 1027–1030.

ASAMI, Y. 1991. A note on the derivation of the first and second derivatives of objective functions in geographical optimization problems. *J. Faculty Eng.* The University of Tokyo(B) *XLI*, 1, 1–13.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4, 469–483.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Trans. Graph. 23*, 3, 905–914.

CORTÉS, J., MARTÍNEZ, S., AND BULLO, F. 2005. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM Control, Optimisation Calculus Variations 11*, 4, 691–719.

DU, Q. AND EMELIANENKO, M. 2006. Acceleration schemes for computing centroidal Voronoi tessellations. *Numer. Linear Alg. Appl. 13*, 2-3, 173–192.

DU, Q., EMELIANENKO, M., AND JU, L. 2006. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM J. Number. Anal. 44*, 1, 102–119.

DU, Q., FABER, V., AND GUNZBURGER, M. 1999. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Rev. 41*, 4, 637–676.

DU, Q. AND GUNZBURGER, M. 2002. Grid generation and optimization based on centroidal Voronoi tessellations. *Appl. Math. Computat. 133*, 2-3, 591–607.

DU, Q., GUNZBURGER, M. D., AND JU, L. 2003. Constrained centroidal Voronoi tesselations for surfaces. *SIAM J. Sci. Comput. 24*, 5, 1488–1506.

DU, Q. AND WANG, D. 2003. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *Int. J. Num. Meth. Eng. 56*, 9, 1355–1373.

DU, Q. AND WANG, D. 2005a. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM J. Sci. Comput. 26*, 3, 737–761.

DU, Q. AND WANG, D. 2005b. The optimal centroidal Voronoi tessellations and the Gersho's conjecture in the three-dimensional space. *Comput. Math. Appl. 49*, 9-10, 1355–1373.

DU, Q. AND WANG, X. 2004. Centroidal Voronoi tessellation based algorithms for vector fields visualization and segmentation. In *Proceedings of the IEEE Conference on Visualization*. IEEE Computer Society, Los Alamitos, CA, 43–50.

FABRI, A. 2001. CGAL-the computational geometry algorithm library. In *Proceedings of the 10th International Meshing Roundtable*. 137–142.

GENZ, A. AND COOLS, R. 2003. An adaptive numerical cubature algorithm for simplices. *ACM Trans. Math. Softw. 29*, 3, 297–308.

GERSHO, A. 1979. Asymptotically optimal block quantization. *IEEE Trans. Inform. Theory 25*, 4, 373–380.

GRUBER, P. M. 2004. Optimum quantization and its applications. *Advances Math. 186*, 2, 456–497.

IRI, M., MUROTA, K., AND OHYA, T. 1984. A fast Voronoi-diagram algorithm with applications to geographical optimization problems. In *Proceedings of the 11th IFIP Conference*. 273–288.

JIANG, L., BYRD, R. H., ESKOW, E., AND SCHNABEL, R. B. 2004. A preconditioned L-BFGS algorithm with application to molecular energy minimization. Tech. rep., University of Colorado.

JU, L., DU, Q., AND GUNZBURGER, M. 2002. Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations. *Parall. Comput. 28*, 10, 1477–1500.

KUNZE, R., WOLTER, F.-E., AND RAUSCH, T. 1997. Geodesic Voronoi diagrams on parametric surfaces. In *Proceedings of the Computer Graphics International*. 230–237.

LEIBON, G. AND LETSCHER, D. 2000. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *Proceedings of the 16th Annual Symposium on Computational Geometry (SCG)*. 341–349.

LIN, C.-J. AND MORÉ, J. J. 1999. Incomplete Cholesky factorizations with limited memory. *SIAM J. Sci. Comput. 21*, 1, 24–45.

LIU, D. C. AND NOCEDAL, J. 1989. On the limited memory BFGS method for large scale optimization. *Math. Prog. Series A and B 45*, 3, 503–528.

LLOYD, S. P. 1982. Least squares quantization in PCM. *IEEE Trans. Inform. Theory 28*, 2, 129–137.

MACQUEEN, J. 1966. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 15th Berkeley Symposium on Mathematical Statistics and Problems, Vol. 1*. University of California Press, 281–297.

MASSEY, W. S. 1991. *A Basic Course in Algebraic Topology*. Springer.

MCKENZIE, A., LOMBEYDA, S. V., AND DESBRUN, M. 2005. Vector field analysis and visualization through variational clustering. In *Proceedings of EuroVis*. 29–35.

MORE, J. J. AND THUENTE, D. J. 1994. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw. 20*, 3, 286–307.

NOCEDAL, J. 1980. Updating quasi-Newton matrices with limited storage. *Math. Computat. 35*, 773–782.

NOCEDAL, J. AND WRIGHT, S. J. 2006. *Numerical Optimization*, 2nd Ed. Springer.

OKABE, A., BOOTS, B., SUGIHARA, K., AND CHIU, S. N. 2000. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams* 2nd Ed. John Wiley & Sons, Inc.

OSTROVSKY, R., RABANI, Y., SCHULMAN, L. J., AND SWAMY, C. 2006. The effectiveness of Lloyd-type methods for the k-means problem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 165–176.

PEYRÉ, G. AND COHEN, L. 2004. Surface segmentation using geodesic centroidal tesselation. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*. 995–1002.

SCHLICK, T. 1992. Optimization methods in computational chemistry. In *Reviews in Computational Chemistry*, K. B. Lipkowitz and D. B. Boyd, Eds. John Wiley & Sons, Inc.

SCHNABEL, R. B. AND ESKOW, E. 1999. A revised modified Cholesky factorization algorithm. *SIAM J. Optim. 9*, 4, 1135–1148.

TÒTH, G. F. 2001. A stability criterion to the moment theorem. *Studia Sci. Math. Hungar. 34*, 209–224.

VALETTE, S. AND CHASSERY, J.-M. 2004. Approximated centroidal Voronoi diagrams for uniform polygonal mesh coarsening. *Comput. Graph. Forum 23*, 3, 381–389.

VALETTE, S., CHASSERY, J.-M., AND PROST, R. 2008. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Trans. Vis. Comput. Graph. 14*, 2, 369–381.

YAN, D.-M., LÉVY, B., LIU, Y., SUN, F., AND WANG, W. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Comput. Graph. Forum* (Symposium on Geometry Processing 2009), *28* 5, 1445–1455.

Received December 2008; acepted May 2009