HKU CS Tech Report TR-2008-18

On Centroidal Voronoi Tessellation — Energy Smoothness and Fast Computation

Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, Chenglei Yang

Centroidal Voronoi tessellation (CVT) is a fundamental geometric structure that finds many applications in computational science and engineering, including computer graphics. The prevailing method for computing CVT is Lloyd's method, which has linear convergence and is inefficient in practice. Our goal is to develop efficient methods for CVT computation, justify the fast convergence of these methods theoretically and demonstrate their superiority with experimental examples in various cases. Specifically, it is shown that the CVT energy function has C^2 smoothness in convex domains and in most other commonly encountered domains with smooth density, correcting the view in the literature that this function is non-smooth (that is, merely C^0 but not C^1). Due to its C^2 smoothness, it is therefore possible to minimize the CVT energy functions using Newton-like optimization methods and expect fast convergence. We apply quasi-Newton methods to computing CVT and demonstrate their faster convergence than Lloyd's method and their better robustness than the Lloyd-Newton method, a previous attempt at CVT computation acceleration. The application of these results to surface remeshing in computer graphics is also studied.

Key Words: centroidal Voronoi tessellation, constrained CVT, Lloyd's method, triangular meshes, remeshing, numerical optimization, quasi-Newton methods

1. INTRODUCTION

A *centroidal Voronoi tessellation (CVT)* is a special Voronoi tessellation of a compact domain in Euclidean space by a set of seed points such that each seed point coincides with the centroid of its Voronoi cell. For example, Figure 1(a) shows an ordinary Voronoi tessellation of a circular domain where the seeds do not coincide with the centroids of the Voronoi cells, and Figure 1(b) shows a CVT of the same domain. CVT provides optimal positions of seed points in the domain with respect to a given density function and is therefore very useful in many fields, including optimal quantization, clustering, data compression, optimal mesh generation, cellular biology, optimal quadrature, coverage control and geographical optimization, etc. An excellent introduction to CVT in theory and applications can be found in [Du et al. 1999; Okabe et al. 2000].

CVT has recently been applied to mesh generation and geometry processing [Du and Gunzburger 2002; Du and Wang 2003; Alliez et al. 2003; Alliez et al. 2005] 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; 2005a] and Valette *et al*. [2004; 2008].

Authors' address: Yang Liu and Bruno Lévy, Project ALICE, INRIA, Villers les Nancy, France; email:{liuyang, levy}@loria.fr. Wenping Wang, Feng Sun, Dong-Ming Yan and Lin Lu, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong, China; email: {wenping, fsun, dmyan, llu}@cs.hku.hk. Chenglei Yang, School of Computer Science and Technology, Shangdong University, China; email: chl_yang@sdu.edu.cn.

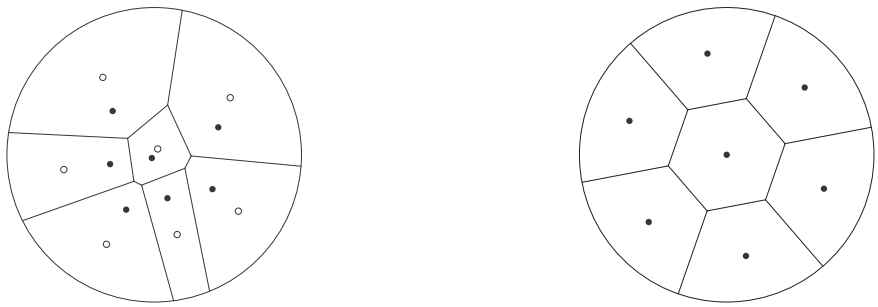


Fig. 1. (a): An ordinary Voronoi tessellation of a circular domain with 7 seeds marked with black dots and the the centroids of the Voronoi cells marked by small circles; (b): a CVT of the same domain with 7 seeds.

Equivalently, CVT can be defined by the critical points (that is, gradient vanishing points) of a certain CVT energy function, 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's conjecture [Gersho 1979]. The globally minimal CVT is difficult to obtain because the CVT energy function is nonlinear and non-convex. Gresho's conjecture in 2D has been proved by Tòth [2001], asserting that in a globally optimal CVT the Voronoi cells away from domain boundary approach to regular hexagons as the number of seeds 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 paper we are interested in efficient CVT computation. The most popular method of CVT computation is Lloyd's method [Lloyd 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 inefficient – it has only linear convergence and is extremely slow for practical applications.

The probabilistic method by MacQueen [1966] is another method for CVT computation, but is not widely used in practice for its lack of computational advantage. For acceleration of CVT computation, Lloyd's method has been implemented in a multi-grid 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 to 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 and the (*wrong*) belief that this function is nonsmooth [Iri et al. 1984]. One notable previous attempt at accelerating CVT computation is the Lloyd-Newton method [Du and Emelianenko 2006]. Since this method minimizes another function different from the CVT energy function, it often produces undesirable CVTs corresponding unstable critical points of the CVT energy function. We will analyze this in detail later in the paper.

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

—it is shown that the piecewise CVT function is C^2 in a *convex compact domain* in 2D and 3D as well as other commonly encountered situations 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 Lloyd's method and the Lloyd-Newton method [Du and Emelianenko 2006], as ex-

pected 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 plan for the remainder of the paper is as follows: in Section 2 we present the formulation of CVT problem and review the existing work on CVT computation. We show in Section 3 that the CVT energy function with sufficiently smoothness density is C^2 in a convex domain in 2D/3D space and propose in Section 4 to use quasi-Newton methods to accelerate CVT computation and demonstrate its superior efficiency in Section 5. In Section 6, we extend our smoothness analysis and fast method to CVT computation on mesh surfaces in 3D, and show how it can be used for quality surface remeshing. We conclude the paper and discuss the future research in Section 7.

2. BACKGROUND AND PREVIOUS WORK

2.1 CVT formulation

Let $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ be an *ordered set* of n seeds in a connected compact region $\Omega \subset \mathbb{R}^N$. The Voronoi region Ω_i of $\mathbf{x}_i$ is defined as

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

The Voronoi regions Ω_is of all the seeds form the Voronoi diagram (VD) of $\mathbf{X}$. A natural assumption is that any two seed points are distinct, i.e., $\mathbf{x}_i \neq \mathbf{x}_j$, $\forall i \neq j$, for the Voronoi boundary consists of bisecting lines of pairs of seeds and a bisecting line is not well defined for two identical seeds. Hence, we shall consider the behavior of $F(\mathbf{X})$ in $\mathbf{\Gamma}_c := \{\mathbf{X} \in \mathbf{\Gamma} | \mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j\}$.

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 constant $\gamma > 0$. Then the centroid of Ω_i is given by

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

where $\mathrm{d}\sigma$ is the area differential.

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

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 seed $\mathbf{x}_i$. Then the *CVT energy function* $F : \Omega^{nN} \to \mathbb{R}$ 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 associated with the seed $\mathbf{x}_i$. 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., a color space of an image) and that the seeds $\mathbf{x}_i$ correspond to samples (e.g., the elements of a colormap). Minimizing the energy F ensures that each sample $\mathbf{x}_i$ is a representative of about the same amount of information in Ω.

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 $m_i = \int_{\mathbf{x}\in\Omega_i} \rho(\mathbf{x})\,\mathrm{d}\sigma$ 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 $F(\mathbf{X})$, i.e., 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. In practice, we prefer a CVT corresponding 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* centroidal Voronoi tessellation *of a compact domain Ω with n seed points $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ is the Voronoi tessellation given by the seeds $\mathbf{X}_0$ which 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 has some negative eigenvalues. A stable CVT, whose Hessian is positive-definite, is shown in Figure 2(b).

(a) (b)

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

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

$$\forall i, \quad \mathbf{x}_i = \mathbf{c}_i = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}\,\mathrm{d}\sigma}{\int_{\Omega_i} \rho(\mathbf{x})\,\mathrm{d}\sigma} \tag{3}$$

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

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 is a fixed-point iteration that simply consists of iteratively moving all the seeds to the centroids of their Voronoi cells, respectively. 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 (3), Lloyd's method can be understood as a gradient descent method that always decreases the energy F without step-size control [Du et al. 1999]. The same type of 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 paper.

—**Geometric characterization :** CVT is a solution of a system of non-linear equations expressing that each seed $\mathbf{x}_i$ coincides with the centroid of its cell $\mathbf{c}_i$ (Equation (3)). 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 analyze this method and compare it with our new methods in detail later in Section 5.2.

The distinction between these two points of view is at the heart 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 fully studying CVT computation from this variational point of view will lead us to faster methods than Lloyd's method.

2.2 Variational point of view

To go beyond Lloyd's method, we consider computing CVT by minimizing the CVT 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 method and its variants). However, due to the believed lack of smoothness of F and the apparent complexity in computing Hessian for a large-scale CVT problem, there have been few successful attempts in the literature in this direction [Iri et al. 1984; Du and Emelianenko 2006].

Minimizing the piecewise-defined CVT energy function F is a numerical optimization problem with several unconventional aspects about the structure and continuity of F, as we will explain. Newton iteration for non-linear optimization uses a 2^{nd}-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 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 second-order derivatives, Newton method has a chance to work only if F is at least C^2 [Nocedal and Wright 2006]. Newton method is not suitable for large-scale problems due to its costly computation of the full Hessian. Therefore, in practice, one often uses quasi-Newton methods to deal with large-scale problems.

Despite the recent result [Cortés et al. 2005] that the CVT function F is C^1, a major impediment to the development of fast Newton-like methods for CVT computation is the lack of understanding about the smoothness of F; for instance, it has long been believed [Iri et al. 1984] that F is nonsmooth, that is, it is C^0 but not C^1. In contrast, we will show that F is almost always C^2. More specifically, F is C^2 in 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 cases except for some cases rarely encountered in practice where it becomes C^1. Furthermore, these results carry over to the constrained CVT problem on mesh surfaces. This newly established smoothness of F provides the necessary justification in our investigation on developing efficient quasi-Newton methods for accelerating CVT computation.

Before entering the heart of the matter, 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 construct the Delaunay triangulation of the vertices defined by $\mathbf{X}$ (i.e. all the seeds), then evaluate the integrals over each cell of its dual Voronoi diagram (see Equation (1)) and differentiate them. The expression of these integrals is rather complicated, since the vertices of the integration domains 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 F changes as well, since it will be based on a different triangulation. In other words, F is piecewise defined in the space of $\mathbf{X}$ and the pieces correspond to the subsets of the space of $\mathbf{X}$ where the combinatorial structure of the Delaunay triangulation remains the same. A combinatorial change occurs in 2D when four or even more vertices become co-circular (or 5 or even more vertices 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 the space of $\mathbf{X}$. Crossing such a gateway results in changes in the expression of F. As we will see in Section 3, F is C^2 at such transitions.

3. ENERGY SMOOTHNESS

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

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

The proof is given in Appendix A.

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 separately elsewhere, due to space limitation.

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

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

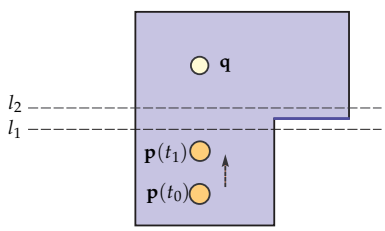


Fig. 3. A configuration where the CVT function is C^1

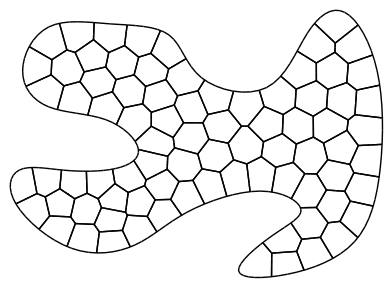


Fig. 4. A 2D non-convex CVT after several Lloyd's iterations. The CVT energy is C^2 around this configuration.

In general, Theorem 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 $\partial\Omega$ is contained in a face of a CVT cell or a smooth part of the boundary curve of Ω is tangent to some face of a Voronoi cell – the CVT function is C^1 but not C^2 in this

case. This is illustrated in 2D in Figure 3. Consider the two points $\mathbf{p}$ and $\mathbf{q}$ in the shown 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 line of $\mathbf{p}$ and $\mathbf{q}$ contains the thick blue horizontal edge, F is C^1 but not C^2.

The 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 seeds with a reasonable distribution (see Figure 4). That is to say, even if the function $F(\mathbf{X})$ is not globally C^2 in this case, the regions where an optimization procedure operates on is almost always C^2.

REMARK 2. *It however seems possible to relax the requirement on the C^2 smoothness of the density function $\rho(\mathbf{X})$. We conjecture that Theorems 1 and 2 still hold when $\rho(\mathbf{X})$ is C^0.*

We now use two simple examples to illustrate the smoothness of 2D and 3D CVT functions. We will see that, as asserted by Theorems 1 and 2, C^2 continuity of the CVT function is attained even when the topological structure of the Voronoi diagram changes.

EXAMPLE 1. *2D case: Here the domain Ω is the square $[-1,1]^2$ with eight seeds, as shown in Figure 5(a). One seed $\mathbf{p}$ moves along straight line $\mathbf{p}(t)$ linearly parameterized by t, and the other seven seeds are fixed and located on a circle. The Voronoi diagram changes during 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 $F''(t)$ is C^0, therefore $F(t)$ is C^2. The kinks in the graph of $F''(t)$ correspond to the structural transitions of VD of the domain when the seed $\mathbf{p}(t)$ crosses the circle.*

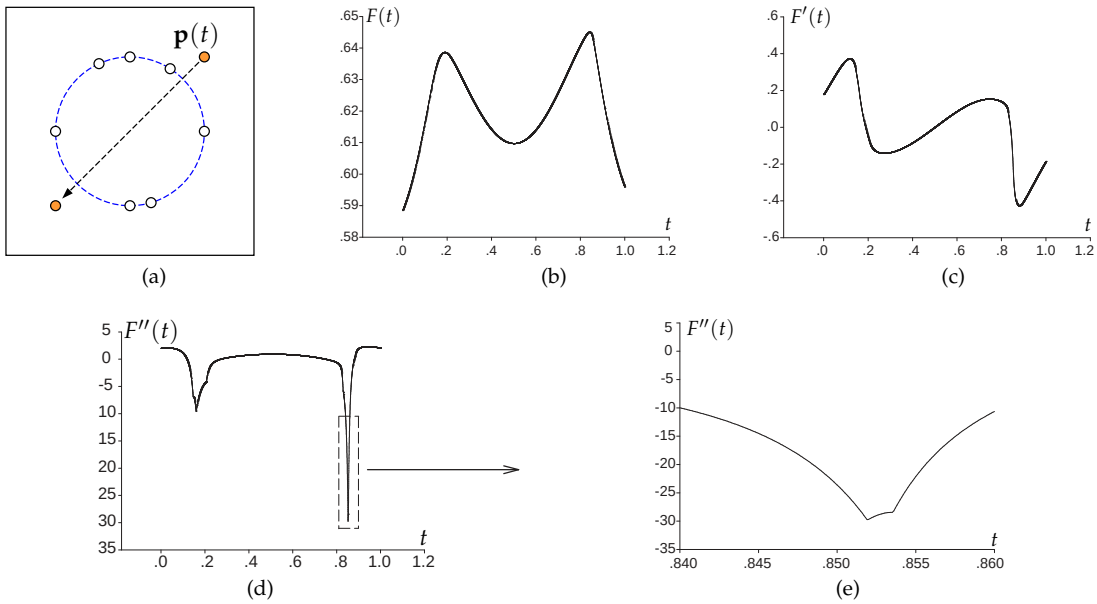


Fig. 5. Illustrations of C^2 smoothness of the 2D CVT function.

EXAMPLE 2. *3D case: Here the domain Ω is the cube $[-1,1]^3$ with eight seeds, as shown in the Figure 6(a). One of the seeds moves along a linear trajectory $\mathbf{p}(t)$ and it passes through a sphere formed by four of the other seeds. The*

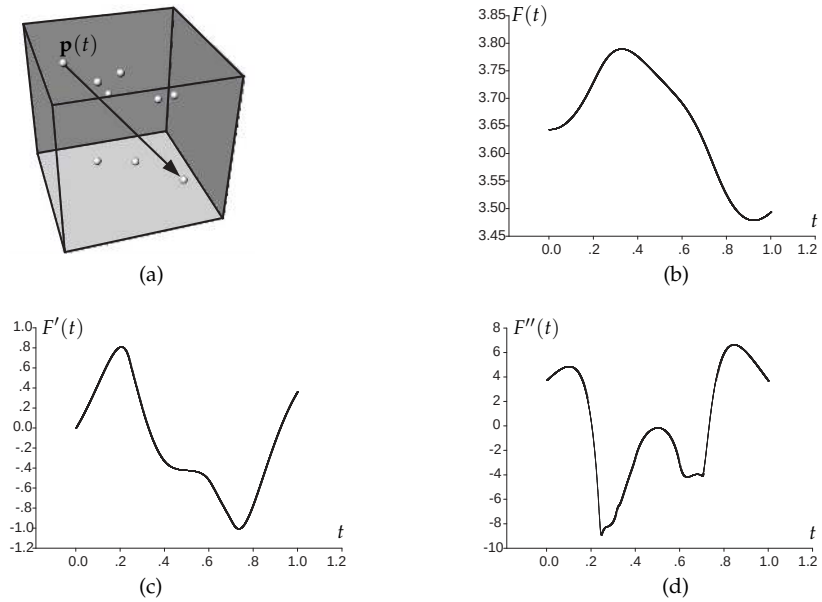


Fig. 6. Illustrations of C^2 smoothness of the 3D CVT function.

graphs of $F(t)$, $F'(t)$ and $F''(t)$ are shown in Figure 6(b)-(c), verifying that the CVT function F is C^2 for this domain Ω in 3D.

4. NUMERICAL OPTIMIZATION

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

4.1 Newton method and quasi-Newton method

Since the 2D and 3D CVT functions are almost always C^2, we may consider using Newton method for nonlinear optimization [Nocedal and Wright 2006] to compute a local minimizer of the CVT function $F(\mathbf{X})$. Newton 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)), and the explicit formulae for the second order derivatives, which are the elements of the Hessian, are available in [Iri et al. 1984; Asami 1991].

Although the Hessian matrix $\mathbf{H}$ is sparse, it is in general not positive-definite, therefore must be modified to be so to define a meaningful search direction. Iri *et al.* [1984] modified the Hessian into a diagonal matrix $\mathbf{D}$ by replacing the diagonal $\frac{\partial^2 F}{\partial x_{ik}^2}$ by $2m_i$. This simplification in fact leads to Lloyd's method, thus is not a good approximation and slows down convergence.

With proper modification such as modified Cholesky factorization [Schnabel and Eskow 1999; Nocedal

and Wright 2006], Newton method is applicable to small and median size optimization problems. Figure 7 shows a simple example in which Newton method is applied to computing a CVT of a square with 20 seeds. As expected due to the C^2 smoothness of the CVT function, the log-scale plot of the gradient $\|\nabla F(\mathbf{X})\|$ shows clearly the quadratic convergence of Newton method.

The main problem of Newton 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 are a large number of seeds in CVT computation. For large-scale CVT computation, the *BFGS* (Broyden-Fletcher-Goldfarb-Shanno) method [Nocedal and Wright 2006], a classical quasi-Newton method, would be more suitable, since it needs only the function value and the gradient, and involves only matrix-vector multiplications. However, the major drawback of the BFGS methods 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* [Nocedal 1980], or the *L-BFGS method* for short, to be discussed below.

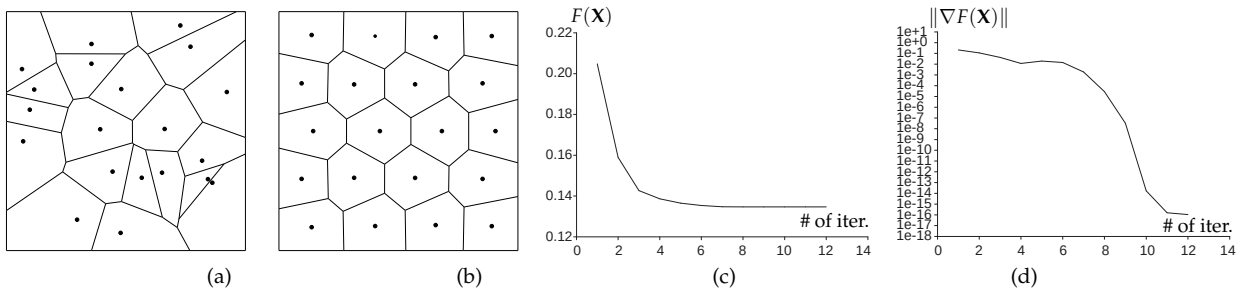


Fig. 7. (a): An initial Voronoi tessellation of 20 points; (b): The final Voronoi tessellation after optimization; (c): $F(\mathbf{X})$ versus the number of iterations; (d): $\|\nabla F(\mathbf{X})\|$ versus the number of iterations.

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 BGFS formula. Here we quote the concise description about the main idea of L-BFGS 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

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

where $\hat{M} = \min\{k, M-1\}$. 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 seeds. 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 uses $O(n^2)$ operations. This saving is key to the efficiency improvement of the L-BFGS over the BFGS method.

Although only the gradient is required in their computation, the proper convergence of the BFGS and L-BFGS algorithm requires that the objective function be C^2 [Nocedal and Wright 2006; Liu and Nocedal 1989]. That is why we have to be concerned with the C^2 continuity of the CVT energy function.

The following 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 see in experiments shortly, L-BFGS is much faster than Lloyd's method. When the Hessian is available, the convergence of L-BFGS can be accelerated further by frequently using the Hessian as the initial of $\widetilde{H}_0$ — this is the idea of the preconditioned L-BFGS method, or P-L-BFGS method for short, by Schlick [1992] and Jiang *et al*. [2004]. Of course, the exact Hessian should not be used in every iteration, otherwise the method will be essentially the full Newton method and become too costly to be useful.

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 at 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 to be the current exact Hessian

after every T iterations. Appropriately chosen values of M and T can yield an acceptable balance between the accuracy of the approximate inverse Hessian and the average time cost per iteration.

In the case of CVT in 2D and 3D, the exact Hessian can be constructed as follows. Let J_i denote the indices of those seed points 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 derivatives of the CVT function are given by the following explicit formulae [Iri et al. 1984; Asami 1991]:

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

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` which indicates whether the Hessian is required;
(2) Evaluate CVT function and compute its gradient g; Construct the Hessian matrix H and apply modified Cholesky algorithm to obtain positive-definite $\widehat{H}$ if `flag` $=$ `.true.` and ($T = 0$ or k `mod` $T = 0$).
(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 satisfied, stop;
(9) Set $k := k + 1$. Go to step 2.

if `flag` $:= \mathit{false}$ in the above algorithm, the P-L-BFGS method is the L-BFGS method actually.

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 discuss the robustness issue of the Lloyd-Newton method, in comparison with the quasi-Newton method.

5.1 Comparison with Lloyd's method

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

Besides Lloyd's method, we have implemented Newton method, the L-BFGS method, and the P-L-BFGS method. In each of the test examples below, we will compare Lloyd's method with all these methods. 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 integrates 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 $\mathbf{x} \in \Omega$ in our P-L-BFGS framework explicitly, especially when the boundary of the domain is more complex than a square/cube, we reduce the increment of the seed points 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 there is no sufficient decrease in line-search. The numerical integration method over simplices from [Genz and Cools 2003] is used when $\rho(\mathbf{x}) \not\equiv \texttt{constant}$. For comparing the performance of different methods, we are concerned with not only the convergence speed but also the total computational time.

In all the methods to be compared, a CVT function call is invoked in each iteration to construct the Voronoi diagram of the current seeds for computing the CVT energy value and gradient. It might be invoked multiple times in each iteration due to line-search, like in the P-L-BFGS method. Therefore, in practice the total number of CVT function calls used 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 $2,000$ points randomly in Ω as the initial seeds (see Figure 8(a)). Table I provides detailed information of all the tested methods.*

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

Table I. Comparisons of Example 3. # of iter. is the number of iterations and # of CVT-func-calls is the number of CVT function calls.

Before reaching the stopping criterion of gradient $\|\nabla F(\mathbf{X})\| \leq 10^{-7}$, Lloyd's method has used $1,000$ iterations and therefore was terminated at the CVT energy (9.75×10^{-6}) higher than the values produced by the other methods. All the other methods have reached stopping criterion of $\|\nabla F(\mathbf{X})\| \leq 10^{-7}$.

Table I shows that P-L-BFGS(20,20) is more efficient than P-L-BFGS(20,10), P-L-BFGS(200, 20), as also attested in Figure 8(f) and (i). Although the number of iterations used by P-L-BFGS(20,10) and Newton method is less than that of P-L-BFGS(20,20), their more frequently use of costly Hessian computation and modification slow 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 Figure 8(g), (h), (j) and (k) in terms of the CVT energy and gradient with respect of the number of iterations and the computational time. Similarly, since L-BFGS(7) is more efficient than L-BFGS(20), we show only L-BFGS(7) in the above figures.

Figure 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 observing comparing the number of hexagonal Voronoi cells (in blue). This visual evaluation makes sense because of the observation by Gersho [Gersho 1979] (see the discussion in Section 1).

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

All these data demonstrate 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 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 $[-2,2]\times[-2,2]$. We sample $4,000$ points randomly in Ω as the initial seeds (see Figure 9). The density function is $\rho(\mathbf{x}) = e^{-20(x^2+y^2)} + 0.05\sin^2(\pi x)\sin^2(\pi y)$. Figure 9 shows the comparisons of $F(\mathbf{X})$ and $\|\nabla F(\mathbf{X})\|$ with respect to the number of iterations and computational time. Table II shows other information of all the tested methods, in the same format as in the previous example.*

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)	385	394	75.30	6.5491805e-5	9.086e-8	0.196
L-BFGS(20)	402	408	78.09	6.5420137e-5	9.997e-8	0.194
Lloyd	1000	1000	196.52	6.5833232e-5	3.277e-7	0.197
Newton	192	323	218.00	6.5781663e-5	8.137e-8	1.135
P-L-BFGS(20,10)	166	211	55.08	6.5771781e-5	8.737e-8	0.332
P-L-BFGS(20,20)	191	287	64.88	6.5564676e-5	6.411e-8	0.340
P-L-BFGS(200,20)	227	334	75.87	6.5723330e-5	8.677e-8	0.334

Table II. Comparisons of Example 4.

In this example we also find that the L-BFGS and the P-L-BFGS give the best performance as compared to Lloyd's method and Newton method. We note the P-L-BFGS is not very effective in its initial stage, but starts to converge very fast after 15 iterations.

EXAMPLE 5. *3D CVT with density $\rho(\mathbf{x}) \equiv 1$: $2,000$ seed points are sampled from an ellipsoid-shaped polyhedron with the bounding box $[-3.236, 3.236]\times[-2.427, 2.427]\times[-1.942, 1.942]$. Figure 10 shows the comparisons of $F(\mathbf{X})$ and $\|\nabla F(\mathbf{X})\|$ with respect to the number of iterations and computational time. 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 Lloyd's method is the most inefficient one.*

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	1.975
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 III. Comparisons of Example 5.

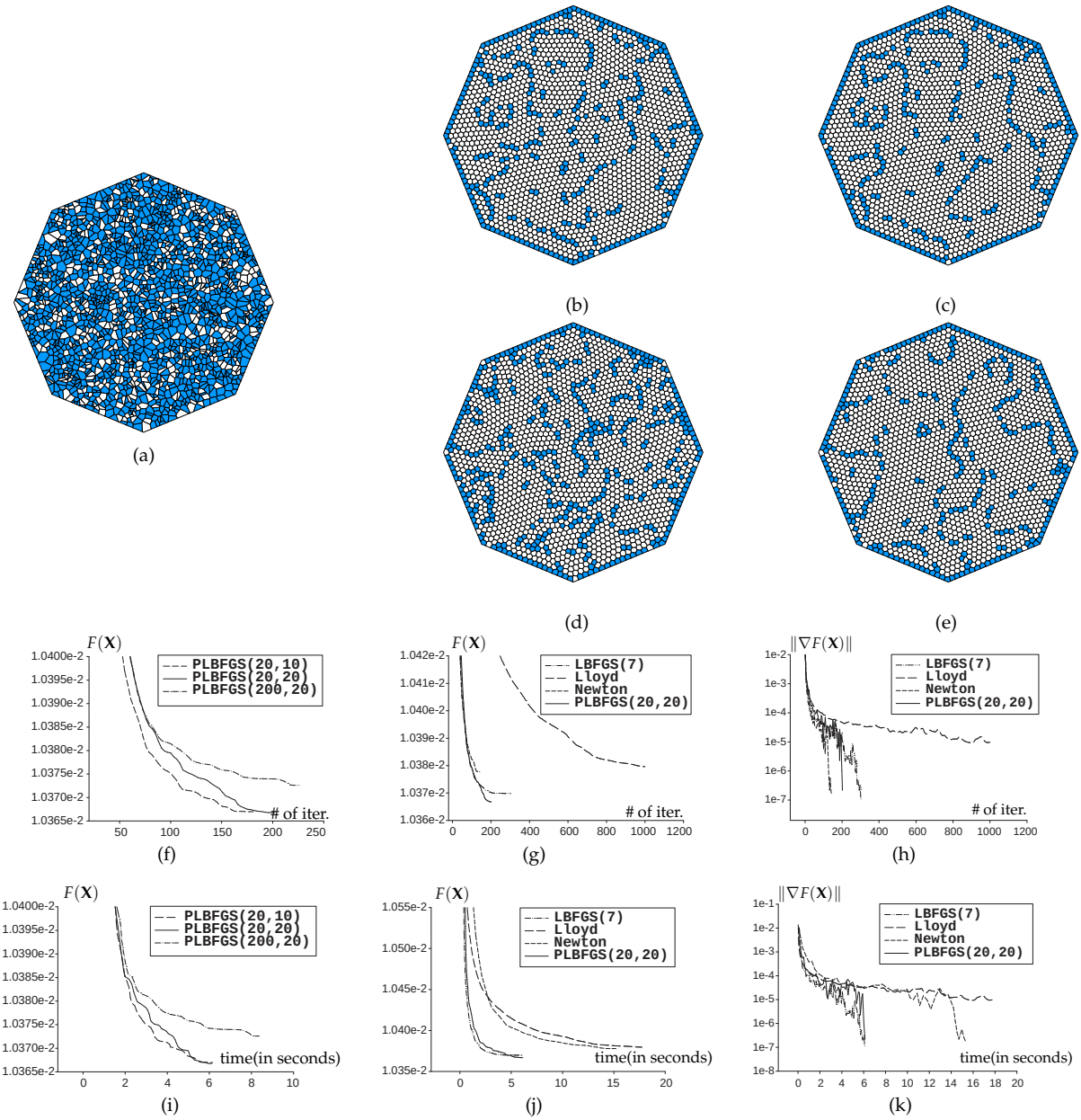


Fig. 8. Example 3. (a): The initial Voronoi tessellation; (b): the result after 100 iterations by P-L-BFGS(20, 20); (c): the result after 203 iterations by P-L-BFGS(20, 20); (d): the result after 153 Lloyd's method (the computational time is same as (b)); (e): the result after 1000 Lloyd's method; (f): $F(\mathbf{X})$ versus # of iter. of the P-L-BFGS methods; (g): $F(\mathbf{X})$ versus # of iter. of selected methods; (h): $\|\nabla F(\mathbf{X})\|$ versus # of iter. of selected methods; (i): $F(\mathbf{X})$ versus the computational time of P-L-BFGS methods; (j): $F(\mathbf{X})$ versus the computational time of selected methods; (k): $\|\nabla F(\mathbf{X})\|$ versus the computational time of selected methods.

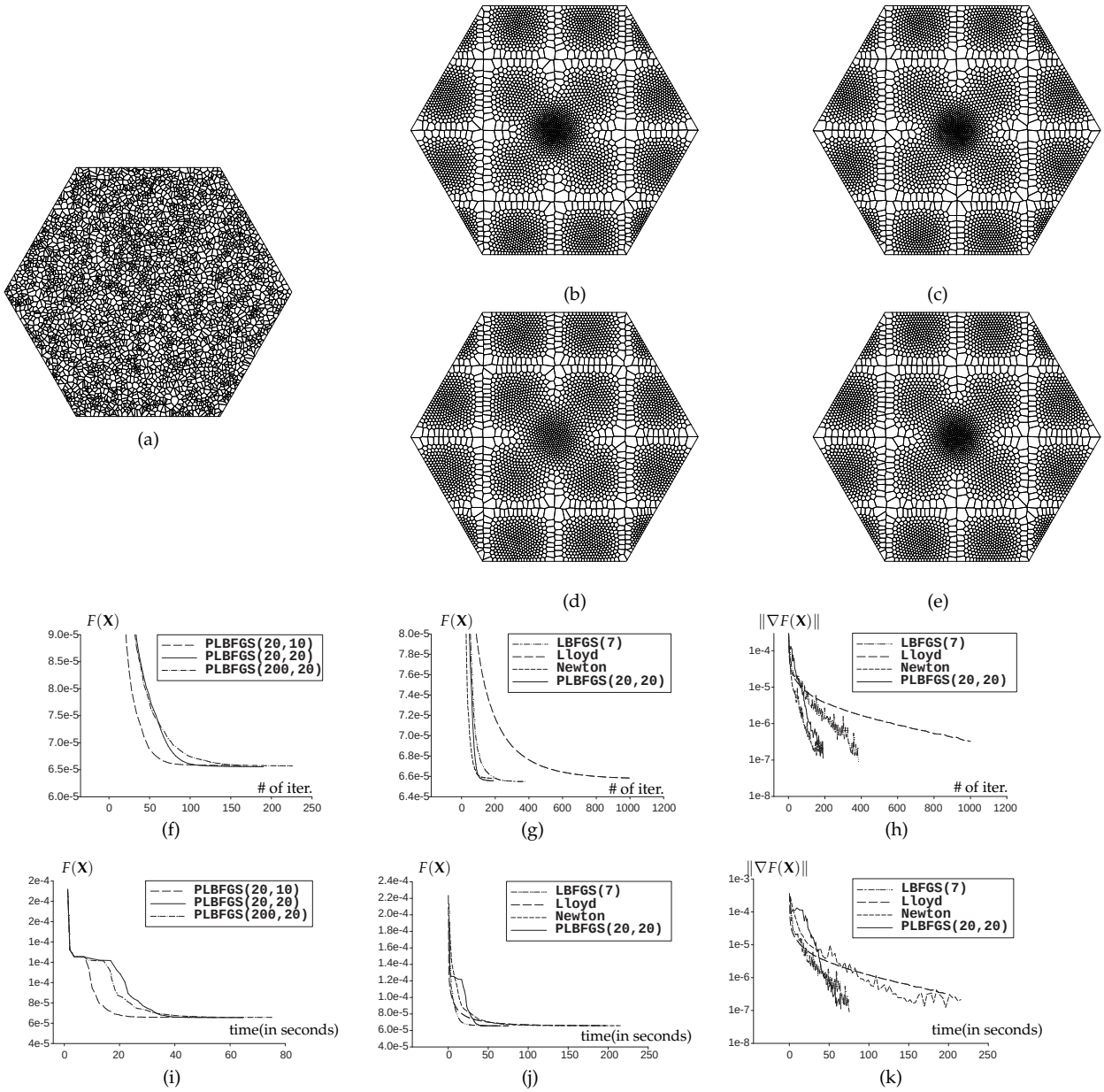


Fig. 9. Example 4. (a): The initial Voronoi tessellation; (b): the result after 100 iterations by P-L-BFGS(20, 20); (c): the result after 191 iterations by P-L-BFGS(20, 20); (d): the result after 203 Lloyd's method (the computational time is the same as (b)); (e): the result after 1000 Lloyd's method; (f): $F(\mathbf{X})$ versus # of iter. of P-L-BFGS methods; (g): $F(\mathbf{X})$ versus # of iter. of selected methods; (h): $\|\nabla F(\mathbf{X})\|$ versus # of iter. of selected methods; (i): $F(\mathbf{X})$ versus the computational time of P-L-BFGS methods; (j): $F(\mathbf{X})$ versus the computational time of selected methods; (k): $\|\nabla F(\mathbf{X})\|$ versus the computational time of selected methods.

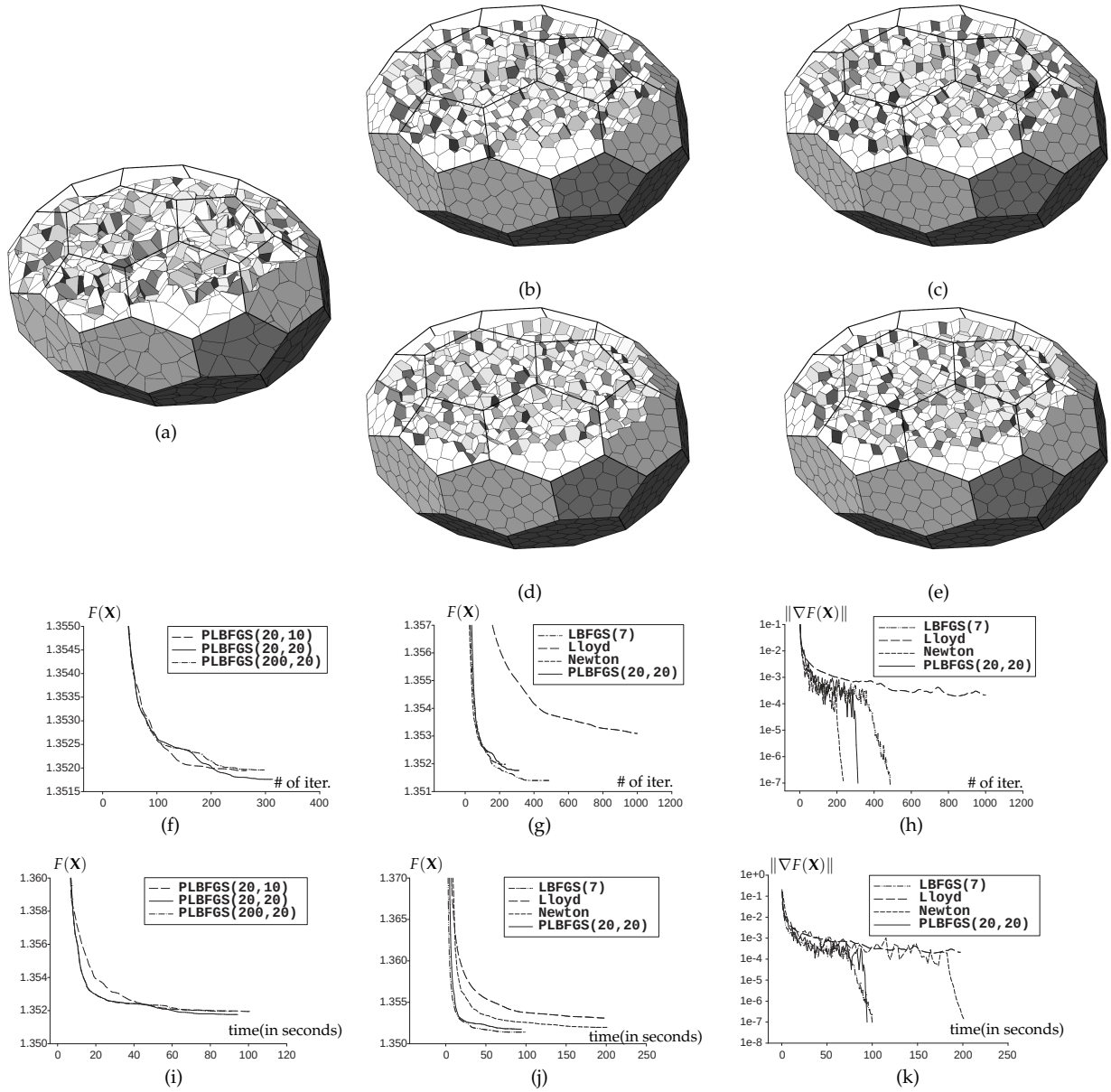


Fig. 10. Example 5. (a): The initial Voronoi tessellation; (b): the result after 100 iterations by P-L-BFGS(20, 20); (c): the result after 313 iterations by P-L-BFGS(20, 20); (d): the result after 129 Lloyd's method (the computational time is the same as (b)); (e): the result after 1000 Lloyd's method;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

5.2 Lloyd-Newton method

The Lloyd-Newton method [Du and Emelianenko 2006] is a previous attempt at CVT computation acceleration. We will show that it essentially uses the Gauss-Newton method to minimize an objective function different from the CVT energy function, and as a consequence, it often fails to find a stable CVT, unless many Lloyd iterations are first used to pre-condition the initialization. But that will then make the overall method very inefficient.

The Lloyd-Newton method uses Newton root-finding method to solve the system of non-linear equations given by Equation (3). 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. Equation (3) 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 the *Lloyd map*. Newton method for solving a system of non-linear equations 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 :

$$\begin{aligned} \text{solve} \quad J_{\mathbf{G}}\, \delta_{\mathbf{X}} &= -G(\mathbf{X}) \\ \mathbf{X} &\leftarrow \mathbf{X} + \delta_{\mathbf{X}} \end{aligned} \tag{4}$$

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

We now show that the Lloyd-Newton method is essentially the same as applying the Gauss-Newton method to the least squares problem that minimizes the function

$$\hat{F} = \frac{1}{2} \sum \|\mathbf{x}_i - \mathbf{c}_i\|^2 = \frac{1}{2} \|G(\mathbf{X})\|^2 \tag{5}$$

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

$$\nabla \hat{F} = J_{\mathbf{G}}^T\, \mathbf{G}$$

$$\mathbf{H}' = J_{\mathbf{G}}^T\, J_{\mathbf{G}} + \mathbf{Q} \simeq J_{\mathbf{G}}^T\, J_{\mathbf{G}}$$

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 number of equations and unknowns are same, $J_{\mathbf{G}}$ is a square matrix. If $J_{\mathbf{G}}$ is non-singular, $J_{\mathbf{G}}^T$ can be removed from both sides, and this produces exactly the same update scheme as in the Lloyd-Newton method (cf. Equation 4).

There are two aspects that make it difficult to compute CVT by minimizing $\hat{F} = \frac{1}{2}\|\mathbf{X} - \mathbf{T}(\mathbf{X})\|^2 = \frac{1}{2}\|G(\mathbf{X})\|^2$ using a Newton-like method. First, the Lloyd map $\mathbf{T}(\mathbf{X})$ is only a C^1 function. So the quadratic convergence cannot be expected in general by either applying Newton method to solve $G(\mathbf{X}) = 0$ or applying the Gauss-Newton method to minimize $\hat{F} = \frac{1}{2}\|G(\mathbf{X})\|^2$. Second, even more important, 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 function F or to a local nonzero minimizer of $\hat{F}$, which is not even a critical point of F.

In the originally proposed Lloyd-Newton method [Du and Emelianenko 2006], to increase the chance of converging to a stable CVT, a large number of Lloyd's iterations are first performed to condition the input before the Newton root finder is invoked, hence the name *Lloyd-Newton*. Therefore, the Lloyd-Newton method can be regarded as a hybrid method consisting of Lloyd's method and the Newton root-finding method.

To compare it with other methods experimentally, we implemented the Lloyd-Newton method following the flow provided in [Du and Emelianenko 2006]. It is listed below.

(1) Perform Lloyd's algorithm. If the difference of centroids and seeds is larger than ϵ, the seeds are updated by centroids, goto 1; otherwise set stepsize to 1, goto 2;

(2) Perform Newton method to solve the system of nonlinear equations in Equation (3);

(3) n is the number of computed seeds which are outside of the domain Ω. If $n = 0$, update seeds; if $n = 1$, half the stepsize, goto 3; if $n > 1$, goto 1.

The performance of the Lloyd-Newton algorithm is sensitive to the parameter ϵ, which dictates when one should switch from Lloyd's iteration to Newton iteration. If ϵ is too small, the Lloyd-Newton method is just 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 will likely move to an unstable CVT.

Incidentally, there is a flaw in the above procedure for the Lloyd-Newton method as provided [Du and Emelianenko 2006]. In step (1), if the difference bewteen the centroids and the seeds is smaller than ϵ, then we go to (2). In step (2), if there are two or more seeds computed by Newton method are outside of the domain Ω, then we go to step (1). Therefore, when both conditions are satisfied simultaneously, the Lloyd-Newton method is stuck in an infinite loop.

We now provide experimental results for examining 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 seeds. In each test, the 100 seeds are initialized with a uniformly random distribution. For each value of ϵ, we record the number of Lloyd's iterations needed, the number of Newton iterations needed, and the numbers of stable CVTs and unstable CVTs computed by the Lloyd Newton method. The results are shown in the following table.

ϵ	# of failures	# of unstable CVTs	# of stable CVTs	# of Lloyd iter.	# of Newton 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

In this table, # of *failure* is the number of tests (out of a total of the 100 tests) in which the Lloyd-Newton method fails to find a critical points of F – either it returns a non-zero minimizer or falls in an infinite loop; # of *unstable CVTs* is the number of times it returns an unstable CVT; # of *minimum* is the number of times it returns a stable CVT; # of *Lloyd* is the averaged number of Lloyd iterations used in each test; # of *Newton* is the averaged number of Newton iterations used in each test.

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, we obtained stable CVTs in all the 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 the similar level of robustness and efficiency as the L-BFGS method.

To conclude, the Lloyd-Newton method, based on finding the fixed points of the Lloyd map $\mathbf{T}$, is equivalent to minimizing the energy $\hat{F} = \frac{1}{2}\sum \|\mathbf{x}_i - \mathbf{c}_i\|^2$ with the Gauss-Newton method. This approach is not efficient because it requires to first use a large number of Lloyd iterations to provide a very good initial value, making the computation inefficient. However, without such an expensive initialization, it will suffer from lack of robustness as it will often get stuck in an unstable CVT or fail to converge at all.

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 with several possible applications, including *surface remeshing* [Alliez et al. 2003]. It is natural to use geodesic metric in 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*. propose *Constrained Centroidal Voronoi Diagram* (CCVT) [2003], in which geodesic metric on a surface is approximated by Euclidean metric in 3D. We adopt this formulation of CVT on a surface in the following experiments in mesh surfaces.

6.1 Constrained centroidal Voronoi tessellation

Denote a given smooth surface by $\Omega \subset \mathbb{R}^3$. Let $\mathbf{X}$ denote the distinct seed points $(\mathbf{x}_i)_{i=1}^n$ in Ω, that is, $\forall i \neq j, \mathbf{x}_i \neq \mathbf{x}_j$. The Voronoi region of the seed $\mathbf{x}_k$ 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]. The CCVT function is defined by

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

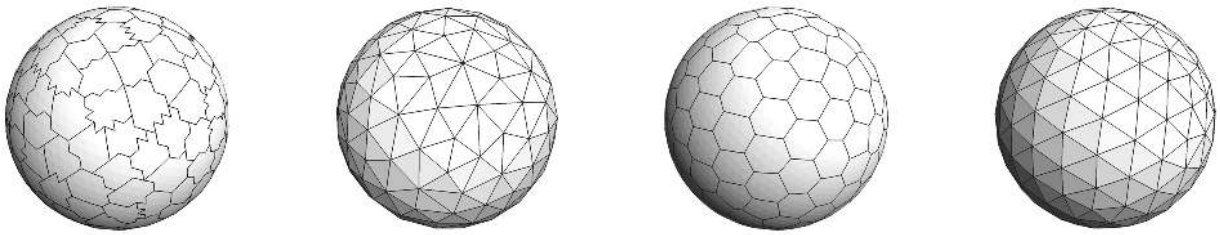
Du *et al*. proved that Lloyd's method works well for CCVT [2003].

On a surface $F(\mathbf{X})$ is also defined in a piecewise manner – its expression takes different forms for combinatorially different Voronoi tessellations, and the change in this expression occurs when there is a structural change of the Voronoi tessellation. It can also be shown that at such a change the CVT function is in general C^2.

For computation of CCVT, we need the projection of the gradient of CVT on 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 CVT as follows:

$$\left.\frac{\partial F}{\partial \mathbf{x}_i}\right|_{\Omega} = \frac{\partial F}{\partial \mathbf{x}_i} - \left(\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{N}(\mathbf{x}_i)\right) \cdot \mathbf{N}(\mathbf{x}_i) \tag{6}$$

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



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

Fig. 11. Voronoi diagrams and their dual. (a): Voronoi diagram by face-clustering CVT; (b) the dual triangulation by face-clustering CVT; (c) Voronoi diagram by face-splitting CVT; (d) the dual triangulation by face-splitting CVT. We can find that the result via face-splitting is more regular.

With this gradient function, we are able to implement L-BFGS method for computing CCVT on a surface. Note that, the updated $\mathbf{x}_i$ in every iteration needs to be projected to the nearest point on the surface, as required by the formulation of CCVT.

6.2 Computing Voronoi diagram on mesh surface

The most time consuming part of CVT computation is to determine the Voronoi region for each seed. 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 approximate Vornoi diagrams. Valette *et al*. employed face clustering with approximated CCVT energy using Lloyd algorithm [2004]. These methods are discrete in the sense that each triangle face of the input mesh is the smallest primitive that is not subdivided anymore. Therefore, these methods terminate quickly, but at a sub-optimal solution.

In contrast, we treat a mesh surface as a piecewise continuous surface, and we 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 2,562 vertices and 5,120 faces. The discrete CVT of 200 seeds generated using Lloyd's iteration in the method by Valette *et al*. [2004] and its dual triangle mesh are shown shown in Figure 11(a) and (b). The CCVT of 200 seeds generated by Lloyd's method with face splitting and its dual triangle mesh are shown in Figure 11(c) and (d). There is an obvious difference between the two resulting meshes in terms of the regularity of vertex degrees and shape of the triangle faces. The method by Valette *et al*. [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 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 computation due to space limitation, since they are not so directly related to our main contribution of the present paper.

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

We will 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 vertices of the input meshes are used as the initial seeds. The input mesh, the initial Voronoi tessellation, the computed Voronoi diagram by L-BFGS, and the dual triangle mesh are shown as in Figures 12 and 13, respectively. The parameter M is set to be 7 in L-BFGS (see the discussion in Section 4.1). Figure 12(e) and (f) (Figure 13 (e) and

(f), resp.) show the CVT energy computed by Lloyd's method and the L-BFGS method, with respect to the number of iterations and computation time. The gradient curve is shown in Figure 12(g) (and Figure 13(g), resp.). 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 1,000 iterations to reduce the CVT energy to 4.18544×10^{-5} (with gradient $\|\nabla F(\mathbf{X})\| = 3.37589 \times 10^{-6}$), taking 1,268.25 seconds. In comparison, the L-BFGS method reaches about the same CVT energy value ($F(\mathbf{X}) = 4.18519 \times 10^{-5}$ with gradient $\|\nabla F(\mathbf{X})\| = 2.82868 \times 10^{-6}$) using 36 iterations, taking 34.51 seconds.

For the Lion model, Lloyd's method takes 1,000 iterations to reduce the CVT energy to 2.29266×10^{-5} (with gradient $\|\nabla F(\mathbf{X})\| = 2.4994 \times 10^{-6}$), 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 $\|\nabla F(\mathbf{X})\| = 1.58981 \times 10^{-6}$, 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. Our main theoretical result is the C^2 smoothness of the CVT energy function in a convex domain. We have also shown the practical importance of this result by demonstrating superior efficiency of quasi-Newton methods for computing CVT in 2D, 3D and on mesh surfaces. Our algorithm is much faster than the classical 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.

Computing Voronoi tessellation is the most-time consuming task in every iteration of all the methods discussed. In practical applications, one may exploit the coherence of the Voronoi tessellations between successive iterations to speed up the computation. Normally, after a sufficiently many iterations, the structure of the Voronoi tessellation will change only a little or not at all from its previous iteration. Therefore, it is possible to perform a quick check to confirm the validity of all edges inherited from the previous iteration or perform edge flips to fix the edges if there are a small number of structural changes. However, this issue does not affect much the comparison results of different methods, since according to our test the resulting speedup is not very significant compared with our current implementation of quick VD computation and in any case the speedup will benefit all the methods compared.

All other existing techniques, including the quasi-Newton approach presented in this paper, are only capable of computing a local minimum. We have tested the "Region Teleportation" approach as proposed in [Cohen-Steiner et al. 2004] and have found that, although it does help find better a local minimum sometimes, there is no theoretic result and strong statistical evidence to guarantee that it always works. Because the CVT objective function is known to have a large number of local minimizers, it is a challenge to devise an effective method to find the global minimizer or even a local minimizer of as low CVT energy as practically possible. Such a method would be very useful in generating high quality mesh and of great utility in studying empirically 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 interesting directions for further research. One 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 for meshing on a surface or a domain with boundary, one also needs to consider special

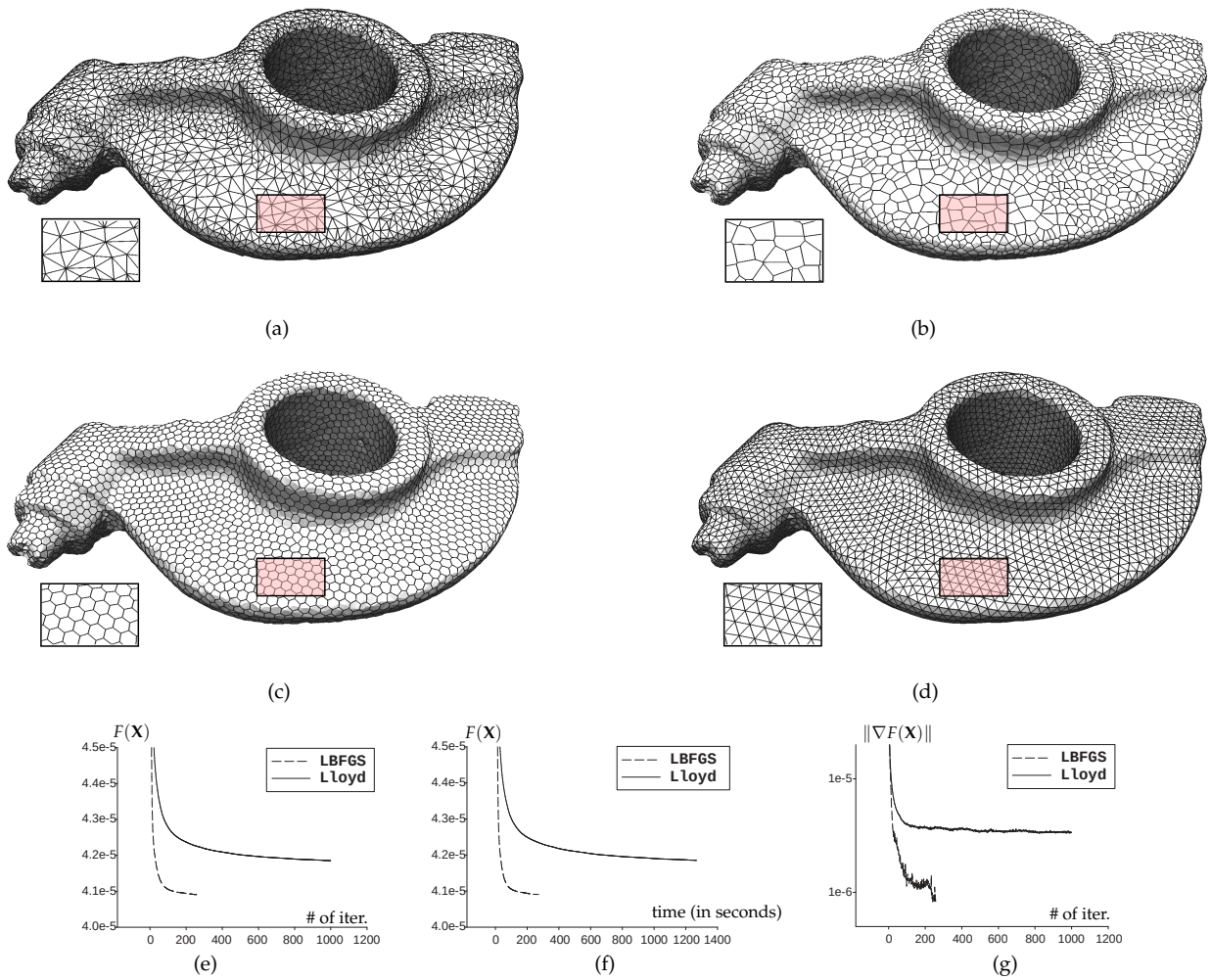


Fig. 12. Rocker model (# of faces: 11,316, # of vertices: 5,658, # of seeds: 5,658, size of bounding box: $0.49 \times 0.29 \times 0.96$). (a) The input mesh; (b): the initial Voronoi diagram; (c): CCVT computed by L-BFGS (# of iter: 284, $F(\mathbf{X})_{final}$=4.0909e-5, $\|\nabla F(\mathbf{X})\|_{final}$=8.3197e-7, computational time: 272.021 seconds); (d): the triangle mesh dual to CCVT; (e): $F(\mathbf{X})$ versus the # of iterations; (f): $F(\mathbf{X})$ versus the computation time (in seconds); (g): $\|\nabla F(\mathbf{X})\|$ versus the # of iterations.

seeds that are constrained to lie on sharp feature curves or boundary in order to preserve the features or boundary representation with the resulting mesh. This poses the problem of defining a CVT framework that accommodates both free seeds and constrained seeds, and issues of efficient computation within this framework.

In this paper we have shown that the CVT function is C^2 in a convex domain in 2D and 3D if the density function $\rho(\mathbf{X})$ is C^2. We conjecture that this is still true if ρ is C^0, as supported by our empirical study. It would be important to settle this conjecture, as the relaxed condition on the smoothness of the density function is useful in many applications that may benefit from our techniques for efficinet CVT computation.

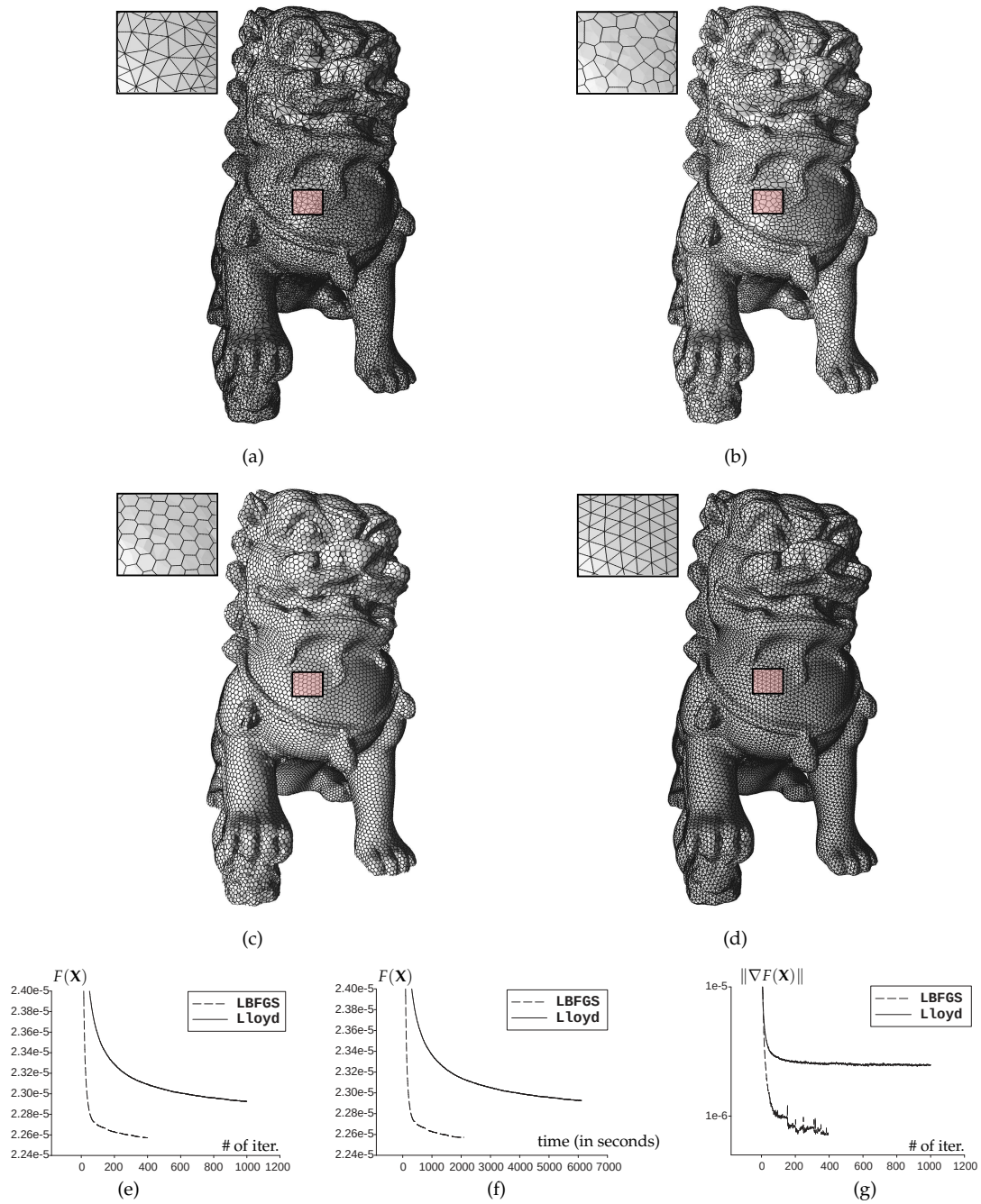


Fig. 13. Lion model (# of faces: 50,000, # of vertices: 25,002, # of seeds: 25,002, size of bounding box: $0.47 \times 0.85 \times 0.83$). (a) The input mesh; (b): the initial Voronoi diagram; (c): CCVT computed by L-BFGS (# of iter: 447, $F(\mathbf{X})_{final}$=2.2572e-5, $\|\nabla F(\mathbf{X})\|_{final}$=7.3149e-7,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

REFERENCES

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. *ACM Transactions on Graphics (Proceeding of SIGGRAPH 2005) 24,* 3, 617–625.

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

ASAMI, Y. 1991. A note on the derivation of the first and second derivatives of objective functions in geographical optimization problems. *Journal of Faculty of Engineering, 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 Transactions on Graphics (Proceeding of SIGGRAPH 2004) 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 and Calculus of Variations 11,* 4, 691–719.

DU, Q. AND EMELIANENKO, M. 2006. Acceleration schemes for computing centroidal Voronoi tessellations. *Numerical Linear Algebra with Applications 13*, 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*, 102–119.

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

DU, Q. AND GUNZBURGER, M. 2002. Grid generation and optimization based on centroidal Voronoi tessellations. *Applied Mathematics and Computation 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. *International Journal for Numerical Methods in Engineering 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. *Computers and Mathematics with Applications 49,* 9-10, 1355 – 1373.

DU, Q. AND WANG, X. 2004. Centroidal Voronoi tessellation based algorithms for vector fields visualization and segmentation. In *IEEE Visualization Proceedings of the conference on Visualization '04*. IEEE Computer Society, Washington, DC, USA, 43–50.

FABRI, A. 2001. CGAL-the computational geometry algorithm library. In *Proceedings of 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. *Information Theory, IEEE Transactions on 25,* 4, 373–380.

GRUBER, P. M. 2004. Optimum quantization and its applications. *Advances in Mathematics 186*, 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. *Parallel Comput. 28,* 10, 1477–1500.

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

LEIBON, G. AND LETSCHER, D. 2000. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *SCG '00: Proceedings of the sixteenth annual Symposium on Computational Geometry*. 341–349.

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

LLOYD, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28,* 2, 129–137.

MACQUEEN, J. 1966. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*. Univ. of Calif. 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. *Mathematics of Computation 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. Wiley.

OSTROVSKY, R., RABANI, Y., SCHULMAN, L. J., AND SWAMY, C. 2006. The effectiveness of Lloyd-type methods for the k-means problem. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 165–176.

PEYRÉ, G. AND COHEN, L. 2004. Surface segmentation using geodesic centroidal tesselation. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*. 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. on Optimization 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. *Computer Graphics Forum (Eurographics 2004 proceedings) 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 Transactions on Visualization and Computer Graphics 14,* 2, 369–381.

A. PROOF OF THEOREM 1

To prove Theorem 1, we first need a thorough understanding of the structure changes of a Voronoi tessellation and need to classify degenerate configurations where the C^2 continuity of the CVT function is to be established. We also introduce necessary notations along the way.

Let VD($\mathbf{X}$) denote the Voronoi tessellation of an ordered set of seeds $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n \in \mathbb{R}^{2n}$. Suppose that the boundary of the domain Ω is a C^0 piecewise curve consisting of a finite number of smooth curve segments. For a given $\mathbf{X}$, a vertex $\mathbf{u}_0$ of VD($\mathbf{X}$) is *singular* if it is of one of the following three types:

Type I: $\mathbf{u}_0$ is at the center of a circle containing more than three seeds in $\mathbf{X}_0$ (see Figure 14(a));

Type II: $\mathbf{u}_0$ is at the joining point $\mathbf{u}_0$ of two consecutive boundary curve segments and on a Voronoi edge passes through $\mathbf{u}_0$ (see Figure 15(a));

Type III: $\mathbf{u}_0$ is the center of a circle containing at least three seeds and located on a boundary curve segment of the domain (see Figure 16(a)).

The Voronoi tessellation VD($\mathbf{X}$) is *degenerate* if at least one of its vertices is singular; otherwise, VD($\mathbf{X}$) is *proper*. If VD($\mathbf{X}$) is degenerate, we call $\mathbf{X}$ a *degenerate set of seeds*, or a *degenerate point* in $\mathbb{R}^{2n}$; if VD($\mathbf{X}$) is proper, then we call $\mathbf{X}$ a *proper set of seeds*, or a *proper point* in $\mathbb{R}^{2n}$.

Oriented abstract cell complex – Given a specific proper point $\mathbf{X}_p$, we encode the structure of VD($\mathbf{X}_p$) by an *oriented abstract cell complex* (or a *complex*, for short), denoted as $C(\mathbf{X}_p)$. The complex $C(\mathbf{X}_p)$ is composed of oriented faces, edges, and vertices, which are isomorphic to the faces, edges and vertices of VD($\mathbf{X}_p$). Specifically, an oriented face f_C of $C(\mathbf{X}_p)$ corresponds to a face f_V of VD($\mathbf{X}_p$) formed by the corresponding vertices, and the orientation of f_C corresponds to the counterclockwise orientation of the boundary of $f_{\mathtt{VD}}$. And the oriented edges in $C(\mathbf{X}_p)$ are defined likewise. A vertex u_C of $C(\mathbf{X}_p)$ is encoded as the triple of seeds such that the center of the circum-circle of the three seeds is u_C's corresponding vertex u_V in VD($\mathbf{X}_p$).

We stress that $C(\mathbf{X}_p)$ encodes only the structure of VD($\mathbf{X}_p$), not its geometry. Therefore, VD($\mathbf{X}_p$) is a particular geometric realization of the complex $C(\mathbf{X}_p)$. Two different proper sets of seeds, $\mathbf{X}_1$ and $\mathbf{X}_2$, may have the same complex, that is, $C(\mathbf{X}_1) = C(\mathbf{X}_2)$. The set of all complexes with n seeds, which are finite in number,

induces equivalence classes in the set of all proper sets of seeds. Treating $C(\mathbf{X})$ as an abstract complex has two advantages: 1) the abstract complex is detached from its geometric realization VD($\mathbf{X}$); and 2) a proper $C(\mathbf{X}_p)$ serves as a representative of its equivalence class under isomorphism of cell complexes. These two properties will be exploited in the following discussion.

Given a complex C_0 and any set of seeds $\mathbf{X}$, we may use the connectivity information in C_0 and the geometry of $\mathbf{X}$ to form a *configuration* on the plane, which we will denote as $(\mathbf{X}; C_0)$. The configuration $(\mathbf{X}; C_0)$ can also be interpreted as geometric embedding of the complex C_0 – the cells of C_0 and the cells $(\mathbf{X}; C_0)$ are in one-to-one correspondence and the vertices of $(\mathbf{X}; C_0)$ are the circumcenters of the triples of seeds in $\mathbf{X}$ as specified by C_0, assuming that the triples of seeds are not collinear and thus their circumcenters are finite. For an oriented face f_C in C_0, its corresponding face f'_C in $(\mathbf{X}; C_0)$ may well be a self-intersecting polygon; the configuration $(\mathbf{X}; C)$ coincides with VD($\mathbf{X}$) only when $C(\mathbf{X}) = C_0$.

CVT energy expression $\Phi(\mathbf{X}; C)$ – Each proper cell complex C is associated with an expression for the CVT energy function $F(\mathbf{X})$. For a particular proper point $\mathbf{X}_p$, each term $F_i(\mathbf{X}_p)$ of $F(\mathbf{X}_p)$ is an integral of a smooth function over a Voronoi cell of VD($\mathbf{X}_p$). This expression of CVT function, to be denoted as $\Phi(\mathbf{X}; C_p)$, is associated with the complex $C_p \equiv C(\mathbf{X}_p)$. Denote $\Pi_p = \{\mathbf{X} \in \mathbb{R}^{2n} | C(\mathbf{X}) = C_p\}$, that is, the *open* set of the proper sets of seeds whose VD have the same structural as that of VD($\mathbf{X}_p$). Let $\bar{\Pi}_{\mathbf{X}_p}$ denote the closure of Π_p. Then $F(\mathbf{X}) = \Phi(\mathbf{X}; C_p)$ if $\mathbf{X} \in \bar{\Pi}_{\mathbf{X}_p}$.

The expression $\Phi(\mathbf{X}; C_p)$ is naturally extended to a neighborhood of $\bar{\Pi}_p$, that is, it is well define even if $C(\mathbf{X}) \neq C_p$. Denote $N(S; r) = \bigcup_{\mathbf{x} \in S}\{B(\mathbf{x}; r)\}$, where $B(\mathbf{x}; r)$ is the closed ball centered at $\mathbf{x}$ and has radius r. Then $N(\bar{\Pi}_p; \delta)$ is δ-neighborhood of $\bar{\Pi}_p$ for some $\delta \geq 0$. We skip the detailed argument here but state that there exists $\delta > 0$ such that the expression $\Phi(\mathbf{X}; C_p)$ is well-defined and C^2 on $N(\bar{\Pi}_p; \delta)$. For any $\mathbf{X}_q \in N(\bar{\Pi}_p; r)$ which is not in $\bar{\Pi}_p$, the integration domains for evaluating the integrations in $\Phi(\mathbf{X}; C_p)$ may be self-intersecting polygons in the configuration $(\mathbf{X}_q; C_p)$. By Green's theorem, these integrals can be converted to line integrals and evaluated along the sequence of oriented edges in $(\mathbf{X}; C_p)$ as indicated by the edges of the oriented faces in C_p. We note that $\Phi(\mathbf{X}; C)$ is C^2 for any complex C, since the density function ρ has been assumed to C^2.

Neighborhood of a degenerate point – Let $\mathbf{X}_0$ be a degenerate set of seeds. Applying an arbitrarily small perturbation $\delta_{\mathbf{X}}$ to $\mathbf{X}_0$, we may obtain a proper point $\mathbf{X}_0 + \delta_{\mathbf{X}}$ that defines the complex $C(\mathbf{X}_0 + \delta_{\mathbf{X}})$. All complexes that can be obtained this way are denoted by the set $\mathcal{H}(\mathbf{X}_0)$; that is, $\mathcal{H}(\mathbf{X}_0)$ contains all the "neighboring" complexes given by the proper points in the neighborhood of the degenerate point $\mathbf{X}_0$, and VD($\mathbf{X}_0$) is a degenerate geometric realization of all the complexes in $\mathcal{H}(\mathbf{X}_0)$. This reflects the fact that a degenerate set of seeds $\mathbf{X}_0$ allows multiple Delaunay triangulations. Obviously, $|\mathcal{H}(\mathbf{X}_0)| \geq 2$.

As an illustration, a local view involving four co-circular seeds $\mathbf{x}_i$, $i = 1, 2, 3, 4$, of a degenerate set of seeds $\mathbf{X}_0$ along with its Voronoi tessellation is shown in Figure 14(a). The two complexes C_1 and C_2 (or their corresponding configurations) incident to this degenerate point are shown in Figure 14(b) and (c)– they are obtained by two perturbations $\delta_{\mathbf{X}}$ and $\delta'_{\mathbf{X}}$ to $\mathbf{X}_0$, respectively. For C_1, we have the following oriented faces associated the seeds $\mathbf{x}_i$ in terms of counterclockwise oriented sequences of their boundary vertices : $R_{1,1} : \mathbf{v}_1\mathbf{w}_1\mathbf{w}'_1\mathbf{v}_2\mathbf{u}_4\mathbf{u}_2\mathbf{v}_1$, $R_{1,2} : \mathbf{v}_2\mathbf{w}_2\mathbf{w}'_2\mathbf{v}_3\mathbf{u}_4\mathbf{v}_2$, $R_{1,3} : \mathbf{v}_3\mathbf{w}_3\mathbf{w}'_3\mathbf{v}_4\mathbf{u}_2\mathbf{u}_4\mathbf{v}_3$, and $R_{1,4} : \mathbf{v}_4\mathbf{w}_4\mathbf{w}'_4\mathbf{v}_1\mathbf{u}_2\mathbf{v}_4$. For C_2, we have $R_{2,1} : \mathbf{v}_1\mathbf{w}_1\mathbf{w}'_1\mathbf{v}_2\mathbf{u}_3\mathbf{v}_1$, $R_{2,2} : \mathbf{v}_2\mathbf{w}_2\mathbf{w}'_2\mathbf{v}_3\mathbf{u}_1\mathbf{u}_3\mathbf{v}_2$, $R_{2,3} : \mathbf{v}_3\mathbf{w}_3\mathbf{w}'_3\mathbf{v}_4\mathbf{u}_1\mathbf{v}_3$, and $R_{2,4} : \mathbf{v}_4\mathbf{w}_4\mathbf{w}'_4\mathbf{v}_1\mathbf{u}_3\mathbf{u}_1\mathbf{v}_4$.

Smoothness of CVT function $F(\mathbf{X})$ – With the above preparation, we now turn to the analysis of smoothness of CVT energy $F(\mathbf{X})$. For any proper point $\mathbf{X} \in \mathbf{\Gamma}_c$, we have the CVT function $F(\mathbf{X}) = \Phi(\mathbf{X}; C_0)$, where $C_0 = C(\mathbf{X})$. Thus, $F(\mathbf{X})$ is a piecewise defined function, since it takes different expressions for configurations $\mathbf{X}$ defining different complexes.

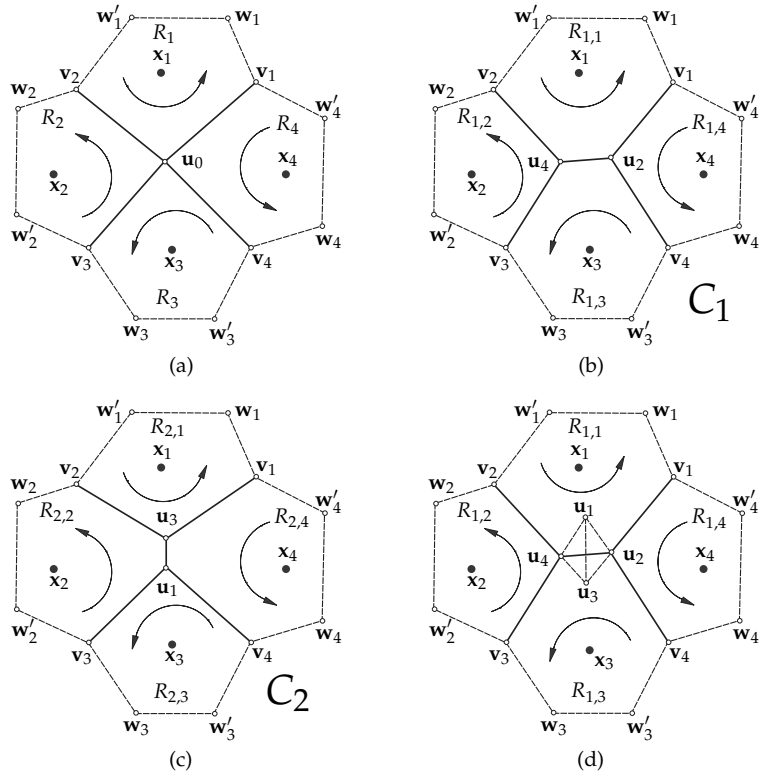


Fig. 14. An illustration for a Type I singular vertex.

First we consider the smoothness of $F(\mathbf{X})$ at a proper point $\mathbf{X}_p \in \mathbf{\Gamma}_c$. For any sufficiently small change $\delta_\mathbf{X}$, $\mathbf{X}_0 + \delta_\mathbf{X}$ is also proper and defines the same complex as given by $\mathbf{X}_0$, that is, $C_p \equiv C(\mathbf{X}_0 + \delta_\mathbf{X}) = C(\mathbf{X}_0)$. It follows that the CVT functions $F(\mathbf{X}_0)$ and $F(\mathbf{X}_0 + \delta_\mathbf{X})$ are evaluated using the same expression $\Phi(\mathbf{X}; C_p)$. Hence, $F(\mathbf{X})$ is C^2 at a proper point, since it inherits the C^2 smoothness of $\Phi(\mathbf{X}; C_p)$.

To complete the proof Theorem 1, it remains to establish the C^2 smoothness of $F(\mathbf{X})$ at a degenerate point X_0. For this we have the following lemma.

LEMMA 1. *Let $\mathbf{X}_0$ be a degenerate set of seeds. Let C_1 and C_2 be any two distinct complexes in $\mathcal{H}(\mathbf{X}_0)$. Then their CVT expressions $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ have C^2 contact at $\mathbf{X}_0$.*

PROOF: Suppose that an arbitrary perturbation $\delta_\mathbf{X}$ of order $O(h)$, where $h > 0$ is arbitrarily small, is applied to $\mathbf{X}_0$ to yield $\mathbf{X} = \mathbf{X}_0 + \delta_\mathbf{X}$. Since $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ are C^2, we can write down their second order Taylor expansions as $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ at $\mathbf{X}_0$: $\Phi(\mathbf{X}; C_1) = Q_1(\mathbf{X}_0; \delta_\mathbf{X}) + O(h^3)$ and $\Phi(\mathbf{X}; C_2) = Q_2(\mathbf{X}_0; \delta_\mathbf{X}) + O(h^3)$. Here, $Q_1(\mathbf{X}_0; \delta_\mathbf{X})$ and $Q_2(\mathbf{X}_0; \delta_\mathbf{X})$ are, respectively, quadratic polynomials in $\delta_\mathbf{X}$ with their coefficients being various derivatives of $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ at $\mathbf{X}_0$, up to the second order. In order to show that $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ have C^2 contact at $\mathbf{X}_0$, we need to show that $Q_1(\mathbf{X}_0; \delta_\mathbf{X}) = Q_2(\mathbf{X}_0; \delta_\mathbf{X})$, $\forall \delta_\mathbf{X}$, that is, all their corresponding coefficients agree with each other.

We claim that $\Phi(\mathbf{X}; C_2) - \Phi(\mathbf{X}; C_1) = O(h^3)$ implies $Q_1(\mathbf{X}_0; \delta_\mathbf{X}) = Q_2(\mathbf{X}_0; \delta_\mathbf{X})$. This can be seen as follows. Suppose that $\Phi(\mathbf{X}; C_2) - \Phi(\mathbf{X}; C_1) = O(h^3)$. Assume that $Q_1(\mathbf{X}_0; \delta_\mathbf{X}) \neq Q_2(\mathbf{X}_0; \delta_\mathbf{X})$. Then $Q_1(\mathbf{X}_0; \delta_\mathbf{X}) -$

$Q_2(\mathbf{X}_0;\delta_\mathbf{X})$ can only be of the order $O(h^q)$, $q = 0, 1$ or 2. However, from the above Taylor expansions of $\Phi(\mathbf{X};C_1)$ and $\Phi(\mathbf{X};C_2)$, we derive

$$Q_2(\mathbf{X}_0;\delta_\mathbf{X}) - Q_1(\mathbf{X}_0;\delta_\mathbf{X}) = \Phi(\mathbf{X};C_2) - \Phi(\mathbf{X};C_1) + O(h^3) = O(h^3).$$

This is a contradiction. Hence, $\Phi(\mathbf{X};C_2) - \Phi(\mathbf{X};C_1) = O(h^3)$ implies $Q_1(\mathbf{X}_0;\delta_\mathbf{X}) = Q_2(\mathbf{X}_0;\delta_\mathbf{X})$.

In the following we will only prove $\Phi(\mathbf{X};C_2) - \Phi(\mathbf{X};C_1) = O(h^3)$ in detail when $\mathrm{VD}(\mathbf{X}_0)$ contains exactly one singular vertex, for when there are multiple singular vertices in $\mathrm{VD}(\mathbf{X}_0)$, we still have $\Phi(\mathbf{X};C_2) - \Phi(\mathbf{X};C_1) = O(h^3)$ by adding up the individual terms of order $O(h^3)$ contributed by all the singular vertices.

Let $\mathbf{u}_0$ be the only singular vertex in $\mathrm{VD}(\mathbf{X}_0)$. Suppose that $\mathbf{u}_0$ is incident to k Voronoi regions and we may suppose that these regions, denoted R_i, are associated with the seeds $\mathbf{x}_i$, $i = 1, 2, \ldots, k$. Apply an arbitrary but fixed perturbation $\delta_\mathbf{X}$ of order $O(h)$ to the degenerate point $\mathbf{X}_0$, and denote $\mathbf{X} = \mathbf{X}_0 + \delta_\mathbf{X}$. Let $\Phi(\mathbf{X};C_1) = \sum_{i=1}^n \phi_{1,i}$ and $\Phi(\mathbf{X};C_2) = \sum_{i=1}^n \phi_{2,i}$. The k terms $\phi_{1,i}$ and $\phi_{2,i}$ associated with the seeds $\mathbf{x}_i$, $i = 1, 2, \ldots, k$, are integrations over oriented regions $R_{1,i}$ and $R_{2,i}$ in the configurations $(\mathbf{X}, C_1)$ and $(\mathbf{X}, C_2)$, respectively. Then we have $\Phi(\mathbf{X};C_2) - \Phi(\mathbf{X};C_1) = \sum_{i=1}^k (\phi_{2,i} - \phi_{1,i})$. Here, all other terms $\phi_{2,i}$ and $\phi_{1,i}$, $i = k+1, \ldots, n$, are canceled out, because they have the identical integration domains $R_{1,i}$ and $R_{2,i}$, $i = k+1, \ldots, n$. This means that the geometric change alone (i.e., the change of positions of the seeds) without structural change of $\mathrm{VD}(\mathbf{X}_0)$ does not contribute to the difference between $\Phi(\mathbf{X};C_2)$ and $\Phi(\mathbf{X};C_1)$.

Denote $T_i = R_{2,i} - R_{1,i}$, where the operation "$-$" refers to the connected sum between two chains [Massey 1991] with the boundary orientation of $R_{1,i}$ reversed. Let $\sum$ denote the connected sum of multiple chains. Clearly, $\sum_{i=1}^n R_{1,i} = \sum_{i=1}^n R_{2,i} = \Omega$, where Ω is regarded as an oriented region. Also, we have $T_i = R_{2,i} - R_{1,i} = \bar{0}$ for $i = k+1, \ldots, n$, where $\bar{0}$ stands for the empty path or the empty oriented region. It follows that

$$\sum_{i=1}^k T_i = \sum_{i=1}^n T_i = \sum_{i=1}^n (R_{2,i} - R_{1,i}) = \sum_{i=1}^n R_{2,i} - \sum_{i=1}^n R_{1,i} = \Omega - \Omega = \bar{0}$$

The fact that $\sum_{i=1}^k T_i = \bar{0}$ is most critical to this proof. To help better appreciate this fact, we now illustrate it using the simple examples of the singular vertex $\mathbf{u}_0$ being of each of the three types.

Case 1 –*Type I singular vertex*: Consider the degenerate set of seeds $\mathbf{X}_0$ with a singular vertex $\mathbf{u}_0$ shown in Figure 14(a). Here $\mathbf{u}_0$ is the circumcenter of 4 seeds; this is the case of $k = 4$. The two complexes C_1 and C_2 in $\mathcal{H}(\mathbf{X}_0)$ correspond to the proper configurations shown in Figure 14(b) and (c). With the perturbation $\delta_\mathbf{X}$, we have $\mathbf{X} = \mathbf{X}_0 + \delta_\mathbf{X}$. Without loss of generality, suppose that $C_1 = C(\mathbf{X} + \delta_\mathbf{X})$. Superimposing the configurations $(\mathbf{X} + \delta_\mathbf{X}; C_1)$ and $(\mathbf{X} + \delta_\mathbf{X}; C_2)$, we obtain the situation in Figure 14(d). The oriented regions $T_i = R_{2,i} - R_{1,i}$, $i = 1, 2, 3, 4$, as differences of oriented regions, are represented by the follow sets of oriented edges, denoted as ordered pairs $(\mathbf{u}, \mathbf{u}')$, on the boundaries of the T_i: $T_1 = \{(\mathbf{u}_2, \mathbf{u}_4), (\mathbf{u}_4, \mathbf{u}_3), (\mathbf{u}_3, \mathbf{u}_2)\}$, $T_2 = \{(\mathbf{u}_1, \mathbf{u}_3), (\mathbf{u}_3, \mathbf{u}_4), (\mathbf{u}_4, \mathbf{u}_1)\}$, $T_3 = \{(\mathbf{u}_1, \mathbf{u}_4), (\mathbf{u}_4, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{u}_1)\}$, $T_4 = \{(\mathbf{u}_1, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{u}_3), (\mathbf{u}_3, \mathbf{u}_1)\}$. Recall that two oriented edges with the same endpoints but opposite orientations cancel each other. It follows that $\sum_{i=1}^4 T_i = \bar{0}$.

Case 2 – *Type II singular vertex*: Consider the degenerate set of seeds $\mathbf{X}_0$ with a singular vertex $\mathbf{u}_0$ shown in Figure 15(a). Here $\mathbf{u}_0$ is the joint of two boundary curve segment and thus incident to two Voronoi cells; this is the case of $k = 2$. The two complexes C_1 and C_2 in $\mathcal{H}(\mathbf{X}_0)$ correspond to the proper configurations shown in Figure 15(b) and (c). With the perturbation $\delta_\mathbf{X}$, we have $\mathbf{X} = \mathbf{X}_0 + \delta_\mathbf{X}$. Without loss of generality, suppose that $C_1 = C(\mathbf{X} + \delta_\mathbf{X})$. Superimposing the configurations $(\mathbf{X} + \delta_\mathbf{X}; C_1)$ and $(\mathbf{X} + \delta_\mathbf{X}; C_2)$, we obtain the situation in Figure 15(d). Note that the vertex $\mathbf{u}_2$, as required by the complex C_2, which is originally the intersection of a bisecting line and the curve segment $\mathcal{B}_2$ in Figure 15(c), is now the intersection point

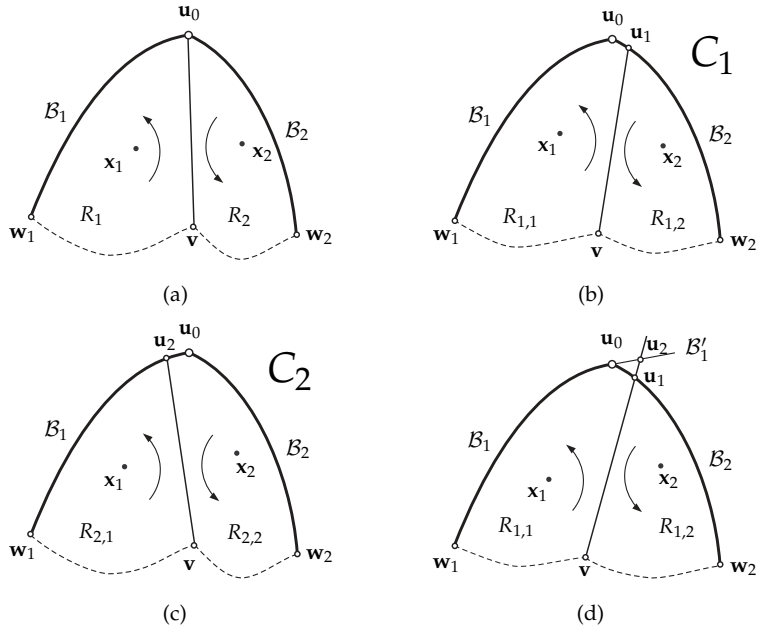


Fig. 15. An illustration for a Type II singular vertex.

between the bisecting line $\mathbf{vu}_1$ and the extended tangent line of the boundary curve $\mathcal{B}_2$ at $\mathbf{u}_0$. The two oriented faces of $\mathbf{x}_1$ and $\mathbf{x}_2$ in the complexes C_1 and C_2 are, respectively, $R_{1,1} : \mathbf{vu}_1\mathbf{u}_0\mathbf{w}_1\mathbf{v}, R_{1,2} : \mathbf{u}_1\mathbf{vw}_2\mathbf{u}_1$ and $R_{2,1} : \mathbf{u}_2\mathbf{w}_1\mathbf{vu}_2, R_{2,2} : \mathbf{u}_2\mathbf{vw}_2\mathbf{u}_0\mathbf{u}_2$. Then the oriented regions $T_i = R_{2,i} - R_{1,i}$, $i = 1, 2$, are represented by the follow sets of oriented edges: $T_1 = \{(\mathbf{u}_0, \mathbf{u}_1), (\mathbf{u}_1, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{u}_0)\}$, $T_2 = \{(\mathbf{u}_1, \mathbf{u}_0), (\mathbf{u}_0, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{u}_1)\}$. It follows that $\sum_{i=1}^{k} T_i = T_1 + T_2 = \bar{0}$.

Case 3 – *Type III singular vertex*: Consider the degenerate set of seeds $\mathbf{X}_0$ with a singular vertex $\mathbf{u}_0$ shown in Figure 16(a). Here $\mathbf{u}_0$ is incident to three Voronoi cells; this is the case of $k = 3$. The two complexes C_1 and C_2 in $\mathcal{H}(\mathbf{X}_0)$ correspond to the proper configurations shown in Figure 16(b) and (c). With the perturbation $\delta_{\mathbf{X}}$, we have $\mathbf{X} = \mathbf{X}_0 + \delta_{\mathbf{X}}$. Without loss of generality, suppose that $C_1 = C(\mathbf{X} + \delta_{\mathbf{X}})$. Superimposing the configurations $(\mathbf{X} + \delta_{\mathbf{X}}; C_1)$ and $(\mathbf{X} + \delta_{\mathbf{X}}; C_2)$, we obtain the situation in Figure 16(d). The three oriented faces of $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ in the complexes C_1 and C_2 are, respectively, $R_{1,1} : \mathbf{w}_3\mathbf{u}_0\mathbf{o}\mathbf{w}_1\mathbf{w}_3$, $R_{1,2} : \mathbf{w}_2\mathbf{o}\mathbf{u}_0\mathbf{w}_4\mathbf{w}_2$, $R_{1,3} : \mathbf{w}_4\mathbf{u}_0\mathbf{w}_3\mathbf{w}_4$ and $R_{2,1} : \mathbf{w}_1\mathbf{w}_3\mathbf{u}_1\mathbf{w}_1$, $R_{2,2} : \mathbf{w}_2\mathbf{u}_2\mathbf{w}_4\mathbf{w}_2$, $R_{2,3} : \mathbf{w}_3\mathbf{w}_4\mathbf{u}_2\mathbf{u}_1\mathbf{w}_3$. Then the oriented regions $T_i = R_{2,i} - R_{1,i}$, $i = 1, 2, 3$, are: $T_1 = \{(\mathbf{u}_0, \mathbf{u}_1), (\mathbf{u}_1, \mathbf{o}), (\mathbf{o}, \mathbf{u}_0)$, $T_2 = \{(\mathbf{u}_0, \mathbf{o}), (\mathbf{o}, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{u}_0)\}$, and $T_3 = \{(\mathbf{u}_0, \mathbf{u}_2), (\mathbf{u}_2, \mathbf{o}), (\mathbf{o}, \mathbf{u}_1), (\mathbf{u}_1, \mathbf{u}_0)\}$. It follows that $\sum_{i=1}^{k} T_i = T_1 + T_2 + T_3 = \bar{0}$.

Now we come back to the discussion of the general case. Denote $g_i(\mathbf{x}) = \rho(\mathbf{x})\|\mathbf{x} - \mathbf{x}_i\|^2$. Then $\phi_{1,i} = \int_{\mathbf{x}\in R_{1,i}} g_i(\mathbf{x})\, \mathrm{d}\sigma$ and $\phi_{2,i} = \int_{\mathbf{x}\in R_{2,i}} g_i(\mathbf{x})\, \mathrm{d}\sigma$, $i = 1, 2, \ldots, k$. Since the seeds $\mathbf{x}_i$, $i = 1, 2, \ldots, k$, are co-circular in $\mathbf{X}_0$, we have $\|\mathbf{x}_i - \mathbf{o}\| = r$, where $\mathbf{o}$ is the center and r the radius of the circle S containing the seeds $\mathbf{x}_i$, $i = 1, 2, \ldots, k$. Clearly, after the perturbation, at $\mathbf{X}_1$ we have $\|\mathbf{x} - \mathbf{o}\| = O(h)$, $\forall \mathbf{x} \in T_i$. Consequently, $g_i(\mathbf{x}) = \rho(\mathbf{x})\|\mathbf{x} - \mathbf{x}_i\|^2 = \rho(\mathbf{x})[r^2 + O(h)]$, $\forall \mathbf{x} \in T_i$, $i = 1, 2, \ldots, k$.

Due to the convexity of the domain Ω, the vertices of T_i, $i = 1, 2, \ldots, k$ are all in the $O(h)$ neighborhood of $\mathbf{u}_0$. Therefore, the length of any edge of T_i is $O(h)$. It follows that the area of T_i is $\int_{\mathbf{x}\in T_i} \mathrm{d}\sigma = O(h^2)$.

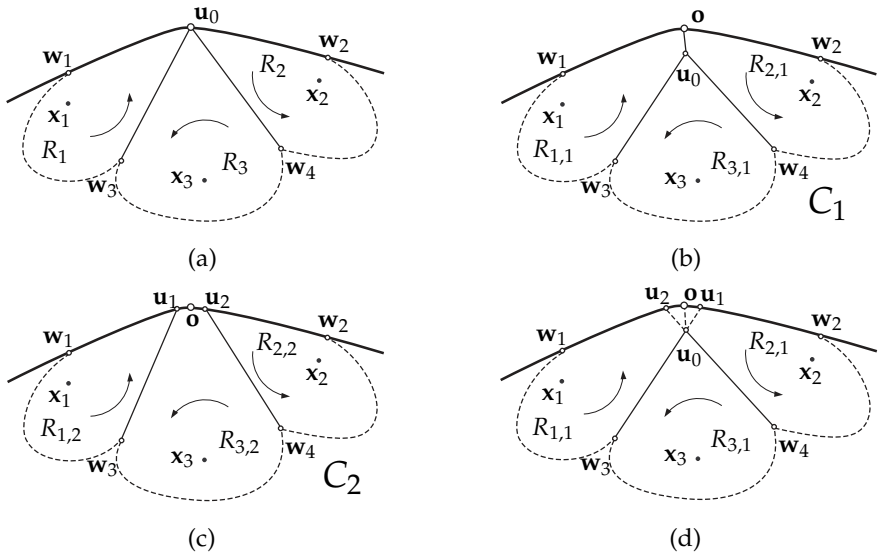


Fig. 16. An illustration for a Type III singular vertex.

From these observations, we have

$$
\begin{aligned}
&\Phi(\mathbf{X}; C_2) - \Phi(\mathbf{X}; C_1) = \sum_{i=1}^{k} [\phi_{2,i} - \phi_{1,i}] \\
=\ & \sum_{i=1}^{k} [\int_{\mathbf{x}\in R_{2,i}} g_i(\mathbf{x})\, \mathrm{d}\sigma - \int_{\mathbf{x}\in R_{1,i}} g_i(\mathbf{x})\, \mathrm{d}\sigma] \\
=\ & \sum_{i=1}^{k} \int_{\mathbf{x}\in T_i} g_i(\mathbf{x})\, \mathrm{d}\sigma = \sum_{i=1}^{k} \int_{\mathbf{x}\in T_i} \rho(\mathbf{x})[r^2 + O(h)]\, \mathrm{d}\sigma \\
\leq\ & r^2 \sum_{i=1}^{k} \int_{\mathbf{x}\in T_i} \rho(\mathbf{x})\, \mathrm{d}\sigma + \gamma \sum_{i=1}^{k} \int_{\mathbf{x}\in T_i} O(h)\, \mathrm{d}\sigma \\
=\ & r^2 \int_{\mathbf{x}\in \sum_{i=1}^{k} T_i} \rho(\mathbf{x})\, \mathrm{d}\sigma + O(h^3) \\
=\ & r^2 \int_{\mathbf{x}\in \emptyset} \rho(\mathbf{x})\, \mathrm{d}\sigma + O(h^3) = O(h^3)
\end{aligned}
\tag{7}
$$

Now we have shown $\Phi(\mathbf{X}; C_1) - \Phi(\mathbf{X}; C_2) = O(h^3)$ at a degenerate point $\mathbf{X}_0$. Hence, the lemma is proved. □

This completes the proof of Theorem 1. ■