Point Cloud Resampling Using Centroidal Voronoi Tessellation Methods

Zhonggui Chen[a,c], Tieyi Zhang[a], Juan Cao[b,c,*], Yongjie Jessica Zhang[c], Cheng Wang[a]

[a]*Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen 361000, China*
[b]*School of Mathematical Sciences, Xiamen University, Xiamen 361000, China*
[c]*Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Abstract

This paper presents a novel technique for resampling point clouds of a smooth surface. The key contribution of this paper is the generalization of centroidal Voronoi tessellation (CVT) to point cloud datasets to make point resampling practical and efficient. In particular, the CVT on a point cloud is efficiently computed by restricting the Voronoi cells to the underlying surface, which is locally approximated by a set of best-fitting planes. We also develop an efficient method to progressively improve the resampling quality by interleaving optimization of resampling points and update of the fitting planes. Our versatile framework is capable of generating high-quality resampling results with isotropic or anisotropic distributions from a given point cloud. We conduct extensive experiments to demonstrate the efficacy and robustness of our resampling method.

Keywords: point cloud, resampling, centroidal Voronoi tessellation, restricted Voronoi cells

1. Introduction

With the rapid development of modern scanning and data acquisition technologies, a huge amount of point clouds on the shape of complicated geometric objects are routinely collected by 3D scanners at an explosive speed. 3D point data is widely used for a variety of applications, such as reverse engineering, prototyping, and entertainment industry. Raw point clouds usually involve noise, redundancy, incompleteness, and uneven distributions, negatively affecting the performance of downstream operations. Hence, resampling the acquired raw data into a noise-free dataset, which is evenly distributed and well represents the underlying shape, has become an important preprocessing stage for point cloud based geometric processing applications.

Most recent attempts have been focused on providing high-quality isotropic point cloud resampling techniques. In many existing methods, the quality of created sampling data points heavily relies on the input data. Uneven distributions are usually introduced in the resampling results when the input point sets are locally sparse or distribute unevenly. Anisotropic resampling, which aims to generate point distributions following given tensor fields such as curvature, provides a more compact representation of the underlying surface. Hence, anisotropic point cloud resampling is more desired in many applications, such as surface approximation and free-from surface modeling.

As a counterpart of point cloud resampling, surface remeshing has received considerable attention over the past few years, and there has been a flourishing of research in computer graphics and numerical computation community on high quality isotropic/anisotropic remeshing. Among them, centroidal Voronoi tessellation (CVT) is a popular technique that has been successfully applied to isotropic/anisotropic remeshing. By minimizing a tailored energy function, CVT based remeshing methods are capable of producing well-shaped elements while preserving the given shape faithfully. Despite the success of CVT methods in remeshing, there remain difficulties in extending them to point cloud resampling. In CVT based surface remeshing techniques, the restricted Voronoi diagram (RVD), i.e., the intersection between a 3D Voronoi diagram and an input surface mesh, is required. Nevertheless, the underlying surface of a point cloud is generally unknown in advance. We notice that the Voronoi diagram can be computed locally as long as the underlying local surface is available. Based on this observation, we can approximate the underlying surface by using tangent planes and compute the Voronoi diagram restricted on them. This approximation gives continuous geometric regions rather than a discrete set of points for the computation of CVT, and enables us to apply the CVT methods to point cloud resampling. Based on the above idea, we propose a versatile CVT-based framework for uniform/weighted or isotropic/anisotropic point resampling in this paper; see Figure 1. To the best of our knowledge, this is *the first study to address the problem of anisotropic point cloud resampling*. The specific contributions of this paper are as follows:

- We provide an extension of the CVT energy function defined on point clouds. The surface represented by a point cloud is approximated by a set of tangent planes, and the RVD is computed as the intersection of the Voronoi diagram of resampling points and tangent planes. Specifically, a regular octagon centered at a resampling point is used to represent the tangent plane. The size of the octagon is

* Corresponding author. Email address: Juancao@xmu.edu.cn (Juan Cao)
Part of this work was done while Zhonggui Chen and Juan Cao were visiting the Department of Mechanical Engineering, Carnegie Mellon University.

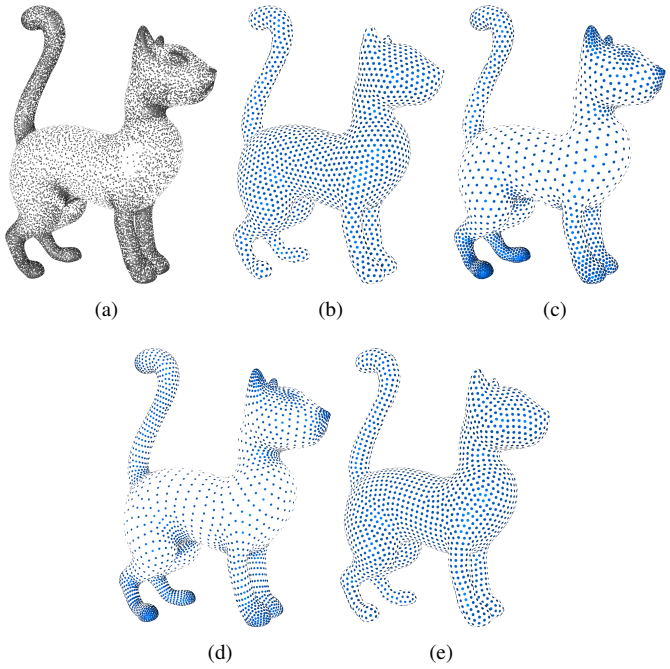


Figure 1: CVT-based isotropic and anisotropic resampling results of an unstructured point could. (a) Input point cloud; (b) uniform resampling; (c) weighted resampling; (d) anisotropic resampling; and (e) L_p resampling result.

adaptively determined to enable fast computation of RVD and robust evaluation of our energy function.

- Effective optimization methods coupled with an improved initialization algorithm are presented to minimize the energy function efficiently, leading to high-quality resampling results with isotropic or anisotropic distribution. By converting the discrete point cloud into a set of continuous planes, the Lloyd's method is used to generate evenly spaced points according to a given density function, despite the sparse and uneven distributions in the input point cloud. The BFGS method, which cannot be directly applied to minimization of the energy function, is tactfully incorporated into our optimization framework to generate anisotropic resampling results.

- Our resampling method can effectively remove noise and fill holes or preserve boundaries of the point cloud. The computation time mainly depends on the number of resampling points, thus it scales up very well to input datasets with millions of points. Furthermore, owing to the dual relationship between the restricted Voronoi diagram and the restricted Delaunay triangulation, our method naturally yields a high-quality surface mesh from the resampling points, providing a direct method for remeshing the underlying surface without a costly surface reconstruction.

The remainder of this paper is organized as follows. After reviewing related previous work in Section 2, we give our objective function in Section 3. We then describe details of our algorithm in Section 4. Implementation details and experimental results are shown in Section 5. Finally, we close with discussions and limitations in Section 6.

2. Related Work

Here, we only review some closely related resampling methods for point clouds and the CVT methods. For a more comprehensive review on methods of filtering 3D point clouds, we refer the readers to [1].

Simplification methods. Simplification is a commonly used tool for efficiently reducing abundances of geometric data. There have been a lot of studies focusing on mesh simplification [2, 3], and some analogous methods can be applied to point cloud simplification [4]. The goal of point cloud simplification is to find a point cloud X with a target sampling rate that minimizes the distance between the surfaces represented by X and the input point dataset P. However, due to the discrete nature of the data, it is not easy to measure the distance between two point clouds. Song and Feng [5] approximated the local shapes of a point cloud by tangent planes, and defined the objective function as the sum of squared distances between the resampled points and the tangent planes, which is then minimized by iteratively clustering the data and selecting a representative point for each cluster. Based on a mean-shift clustering scheme, a curvature-aware resampling method was proposed [6]. To improve the efficiency and quality of simplification, a hierarchical cluster tree was adopted in [7]. Shi *et al*. [8] employed the k-means clustering algorithm to group points together and choose representative points. Another error metric based on a Hausdorff distance of principal curvature vectors was proposed in [9]. Point cloud simplification mainly focuses on how to preserve the surface shape represented by the point cloud, rather than the distribution of the output points, leading to a low level of local uniformity in the resampling results.

Consolidation methods. The raw data points acquired by a scanner usually contain noises and outliers, making the downstream reconstruction difficult. Consolidation is a process to massage the point set into the surface where it was sampled from, by removing noises and outliers [10, 11, 12]. These consolidation methods usually resample point sets to avoid point redundancy or under-sampling problem. Alexa *et al.* [10] reconstructed a surface from a point set using the moving least squares method, where the point with the smallest contribution to the reconstructed shape is repeatedly removed. Lipman *et al*. [13] proposed a locally optimal projection (LOP) method, which approximates the shape of the raw dataset by a number of uniformly distributed points. To deal with non-uniform distributions in raw data, Huang *et al*. [14] developed a weighted locally optimal projection operator (WLOP). Liao *et al*. [15] further considered both spatial and geometric feature information of the point clouds and proposed a feature-preserving locally optimal projection operator. Another edge-aware point set resampling method was given in [16]. It first resamples away from the edges using an anisotropic LOP operator, and then progressively resamples the points near the edges. Preiner *et al*. [17] presented a continuous formulation of the WLOP operator and achieved a significant acceleration. LOP operator and its variants work on adequately dense raw datasets, but they suffer more or less from non-uniform distributions. In addition, they generally target for an even point distribution and do

not have any control over density distribution of the resampling points.

Sampling methods. Point sampling is one of the core algorithms in computer graphics. A lot of methods have been proposed to generate distributions with blue noise properties on a meshed surface [18, 19, 20, 21]. Some of them can also be directly applied to resampling point clouds lying near a surface [18, 20]. While blue noise properties are desirable in many applications such as stippling and rendering, in the context of surface remeshing, uniform distributions are generally preferred for generating high-quality isotropic remeshing results. A recent point cloud resampling method based on the Gaussian-weighted graph Laplacian [22] is capable of making the resultant point distribution conformal to a target density distribution while achieving good local uniformity between points. However, this algorithm runs in $O(N \log N)$, where N is the number of the resampled points. The time increases significantly when N gets large. In this paper, we propose an efficient method that constructs high-quality isotropic/anisotropic resampling results of a point cloud in a time complexity depending only linearly on the target number of points.

CVT methods. A CVT is a particular type of Voronoi tessellation whose generating points coincide with the centroids of the corresponding Voronoi regions. It has been used in a wide scope of applications, ranging from computational sciences to engineering [23]. In computer graphics, CVT based methods have been a promising tool for generating high-quality surface and volume meshes. The Lloyd's method [24] is the most widely used to generate CVTs, due to its simplicity. But it converges slowly for large-scale problems. A quasi-Newton method was adopted in [25] to speed up the rate of convergence. CVT methods can also be applied to surface meshing by resorting to parameterization [26] or efficient algorithms of computing Voronoi diagrams restricted to surfaces embedded in 3D [27] or high dimensional spaces [28]. By incorporating a tensor field and an L_p norm into the CVT objective function, the optimized Voronoi cells align their axes with the given tensor field, resulting in an anisotropic surface/volume mesh [29]. CVT methods relying on the exact computation of restricted Voronoi diagrams on continuous domains are capable of achieving well-separateness between points in the resultant point distribution. On the other hand, discrete CVTs, such as k-means clusters, are greatly affected by non-uniform distributions in raw data.

In this paper, the restricted Voronoi diagram on a point cloud is computed as the intersection between the Voronoi cells of the resampling points and a set of disks centered at the resampling points. Thus existing CVT objective functions can be defined and efficiently minimized over point clouds. It removes the effect of non-uniform distributions in point clouds and leads to high-quality isotropic/anisotropic resampling results.

3. Formulation of Objective Function

Let $X = \{\mathbf{x}_i\}_{i=1}^n$ be a set of seed points on a given domain $\Omega \subset R^d$, then a centroidal Voronoi tessellation corresponds to a minimizer of the objective function, which has a general formulation as follows:

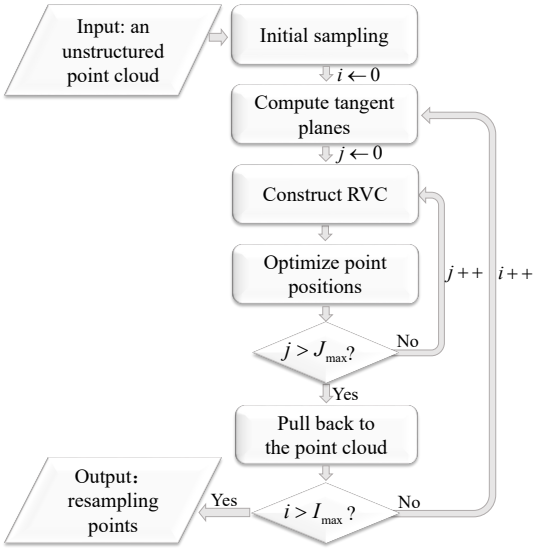


Figure 2: Algorithm overview of CVT-based point resampling.

$$E(X) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(\mathbf{x})\psi(\mathbf{x}, \mathbf{x}_i)\mathrm{d}\sigma, \tag{1}$$

where $\Omega_i = V(\mathbf{x}_i) \cap \Omega$ is the Voronoi cell $V(\mathbf{x}_i)$ of $\mathbf{x}_i$ restricted to the domain Ω, $\rho(\mathbf{x})$ is a density function, and $\psi(\mathbf{x}, \mathbf{y})$ is a metric defining distance between points $\mathbf{x}$ and $\mathbf{y}$. The minimizers of the CVT objective function correspond to different point distributions, depending on the choice of the distance metric $\psi(\cdot, \cdot)$. A typical case is $\psi(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|^2$, which results in isotropic point distributions on Ω. For more details on the theory of CVT, we refer readers to a comprehensive survey paper [23].

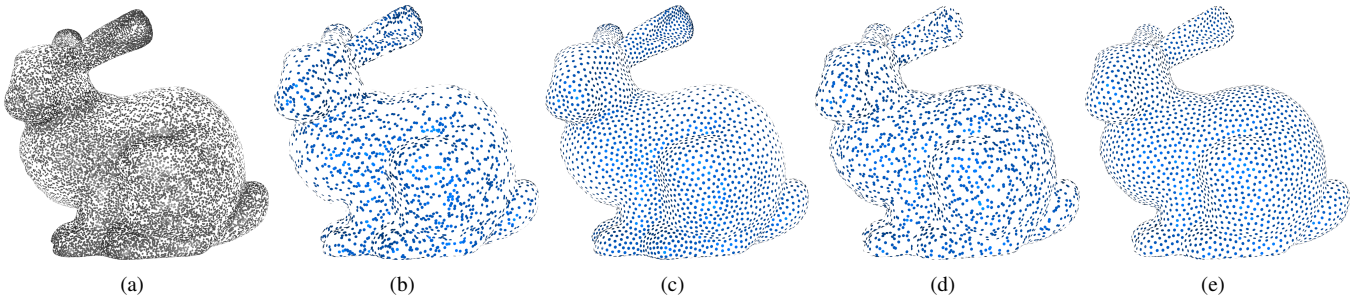
Extending the computation of CVTs in the context of point cloud resampling is not straightforward, as the underlying surface is unknown in general. In this paper, we settle for computing the intersection of Voronoi cells and an approximation of the underlying surface. Specifically, we locally approximate the underlying surface at the point $\mathbf{x}_i$ by its tangent plane τ_i, which is computed as the least squares fitting plane of the k-nearest neighbors of $\mathbf{x}_i$ in the input point cloud. Then, we locally compute the *restrict Voronoi cell (RVC)* for each point $\mathbf{x}_i$ as the intersection of the 3D Voronoi cell $V(\mathbf{x}_i)$ and its tangent plane τ_i. The CVT objective function is therefore modified as follows:

$$E(X) = \sum_{i=1}^{n} \int_{V(\mathbf{x}_i) \cap \tau_i} \rho(\mathbf{x})\psi(\mathbf{x}, \mathbf{x}_i)\mathrm{d}\sigma. \tag{2}$$

In the next sections, we will employ different distance metrics to generate desired point distribution of resampling results and describe our algorithms of optimizing the corresponding objective functions.

4. Algorithm of CVT-Based Point Resampling

Our goal is to provide a versatile framework for generating a point cloud with an isotropic/anisotropic distribution while



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

Figure 3: Initial sampling. (a) Input point cloud with 35K points; (b) initial sampling result by randomly choosing 5K points from (a); (c) an optimized result after applying 10 Lloyd's iterations to (b); (d) initial sampling result according to the areas of RVCs; and (e) a result after applying 10 Lloyd's iterations to (d).

well approximating the underlying shape of a given raw point dataset. Our CVT-based algorithm takes an unstructured point cloud $P = \{\mathbf{p}_i\}_{i=1}^m$ as input, with a density function $\rho(\mathbf{x})$, and a desired point number n. Starting from the raw point cloud, our algorithm proceeds as shown in Figure 2.

(1) In the initialization stage, n points $\{\mathbf{x}_i\}_{i=1}^n$ are randomly sampled from the raw point cloud according to a density function.

(2) For each sampling point $\mathbf{x}_i$, we estimate its tangent plane by computing the least squares fitting plane of its k-nearest data points.

(3) We locally compute Voronoi cells restricted to the tangent planes.

(4) Sampling points are relocated on each local tangent plane suggested by the Lloyd's method or a gradient-based optimization method. We repeat Steps (3)-(4) J_{max} times without changing local tangent planes.

(5) As the updated sampling points may lie a litter far away from the underlying surface, these points are pulled back onto the underlying surface.

The distribution of the resampling points is gradually improved by iteratively running Steps (2)-(5). In other words, point positions and their tangent planes are alternatively updated. The algorithm terminates upon either of the following two criteria: convergence occurs, i.e., the maximum distance moved by any point in an iteration falls below a preset threshold, or a predetermined maximum number of iterations is reached. Details of the algorithm is given as follows.

4.1. Initialization

A straightforward initialization of our algorithm is to randomly sample n points from the point cloud P, which is greatly affected by the distribution of the input points and results in an initial point set that is inconsistent with the target density function $\rho(\mathbf{x})$; see Figure 3(b) for an example. To speed up the convergence of our iterative algorithm, we here propose a more sophisticated algorithm for initialization as follows.

1. A point set $\{\mathbf{x}_i\}_{i=1}^n$ of size n is randomly chosen from the input data P and its 3D Voronoi diagram is computed.

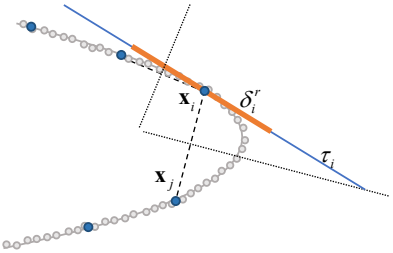


Figure 4: RVC computation: data points and resampling points are marked in gray and blue, respectively.

2. For each point $\mathbf{x}_i$, we compute its tangent plane τ_i. In particular, for a point $\mathbf{x}_i$, we find its k nearest neighbors $\{\mathbf{p}_{i_j}\}_{j=1}^k$ from the point cloud P and compute the plane τ_i that best fits the samples $\{\mathbf{p}_{i_j}\}_{j=1}^k$ in the least-squares sense.

3. We compute the restriction of Voronoi cell $V(\mathbf{x}_i)$ onto the corresponding tangent plane τ_i. Note that, the computation of RVC is non-trivial, whose details will be given in Section 4.2. If no confusion arises, we will use the same notation τ_i for both the tangent plane and the RVC (i.e., the cropped tangent plane) of the point $\mathbf{x}_i$.

4. The weighted area for each RVC τ_i, denoted by $A_w(\tau_i)$, is computed as $A_w(\tau_i) = |\tau_i| \sum_{j=1}^k \rho(\mathbf{p}_{i_j})/k$, where $|\tau_i|$ is the area of RVC τ_i. Then, n points are randomly sampled from the RVC set $\{\tau_i\}_{i=1}^n$, with the probability of selecting a RVC τ_i proportional to $A_w(\tau_i)/\sum_{i=1}^n A_w(\tau_i)$.

5. The newly sampled points may deviate from the underlying surface, and will be pulled back to the surface according to the method described in Section 4.3. The pulled back point set, also denoted by $\{\mathbf{x}_i\}_{i=1}^n$, will be the initial sampling points for our method.

Note that, our initialization method solely relies on the density functions. Hence, the distribution of the resampling points is independent of the distribution of the input data points. Figure 3 shows resampling results with a constant density function. We can see that the points obtained by random initialization are clustered heavily on the ear region of the model, even after applying Lloyd relaxation. By contrast, our method generates a better initialization, leading to a more uniform distribution after applying only a few Lloyd iterations.

4.2. RVC computation

To obtain the Voronoi cells, one needs to first build the 3D Delaunay triangulations of the point set $\{\mathbf{x}_i\}_{i=1}^n$. In this paper, we adopt the clipping method [28] to compute the RVD for every point, while avoiding the high computational cost associated with computing the 3D Voronoi cells of the resampling points. The algorithm is based on the observation that Voronoi cell $V(\mathbf{x}_i)$ is the intersection of half-spaces, bounded by the bisectors of point pairs $(\mathbf{x}_i, \mathbf{x}_j)$. In particular, RVD of $\mathbf{x}_i$ is obtained by clipping the tangent plane τ_i by the bisectors of all point pairs $(\mathbf{x}_i, \mathbf{x}_j)$, $j \in 1, ..., n, j \neq i$.

In our implementation, the tangent plane τ_i of $\mathbf{x}_i$ is replaced with an octagon of radius r and centered at $\mathbf{x}_i$, denoted by δ_i^r, where r is set to be the average distance between $\mathbf{x}_i$ and its six nearest neighbors in resampling set X; see Figure 4. The benefits of this substitution are twofold.

(1) It makes the computation of the energy function stable. Since some Voronoi cells are unbounded, it may lead to a RVC with infinite area if we use the unbounded tangent plane for the RVC computation. As shown in Figure 4, the tangent plane τ_i could be nearly parallel to the bisector of $(\mathbf{x}_i, \mathbf{x}_j)$ on highly curved regions, which will result in a RVC with a very large area and an improper approximation of the underlying surface. Thus it will be difficult or even impossible to compute and minimize the energy function in Equation (2).

(2) It accelerates the aforementioned clipping algorithm. When implementing the clipping algorithm, we only need to clip the octagon δ_i^r by bisectors of $\mathbf{x}_i$ and the points $\mathbf{x}_j$ with distances less than $2r$, as the rest bisectors have no contribution to the clipping results. We use the NanoFLANN library [30, 31] for the efficient neighbor searching within a radius bound.

Note that the computation of a RVC is independent on the others, thus it can be easily parallelized. RVCs shown in Figure 5 are obtained by computing the intersections between the disks δ_i^r and their corresponding Voronoi cells. The RVCs may not cover the underlying surface of the input point cloud when the points are unevenly distributed, but they will gradually reduce the gaps while the point positions are being optimized.

4.3. Optimization

This section describes the minimization of energy functions associated with different distance metrics $\psi(\cdot, \cdot)$ defined in Section 3. In particular, the Lloyd's method and the BFGS method are adopted to generate isotropic and anisotropic resampling results, respectively.

4.3.1. Lloyd's method

One prevalent method used for computing isotropic CVTs is the Lloyd's method [24], which interleaves moving each point to the centroid of the corresponding Voronoi cells and recomputing the Voronoi tessellation, until certain stopping criterion

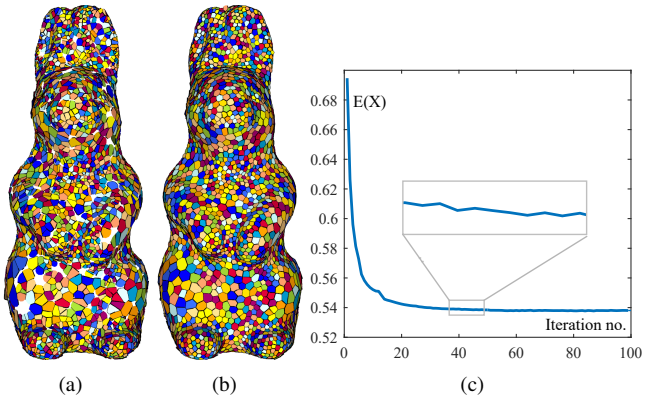


Figure 5: Lloyd's method. (a, b) The RVCs before and after Lloyd's relaxation; and (c) the plot of CVT energy function versus the iteration number of Lloyd's relaxation.

is met. We follow a similar approach by alternatively optimizing the point positions and updating RVCs. However, we update the points in a slightly different way. In traditional restricted CVT computation, the centroids of restricted Voronoi cells are restricted to a given mesh/surface. However, the underlying surface is unknown in our application. A straightforward way of computing RVC on an input point cloud, i.e., the k-means clustering method [8], is to collect the points of the input data inside a Voronoi cell and compute their discrete centroid, leading to sampling results heavily relying on the distribution of the input data. To achieve better results, we compute the centroid of each Voronoi cell restricted to a tangent plane:

$$\mathbf{o}_i = \frac{\int_{V(\mathbf{x}_i)\cap\tau_i} \rho(\mathbf{x})\mathbf{x}\mathrm{d}\sigma}{\int_{V(\mathbf{x}_i)\cap\tau_i} \rho(\mathbf{x})\mathrm{d}\sigma}. \tag{3}$$

The point $\mathbf{x}_i$ is then moved to $\mathbf{o}_i$. Note that, the density function $\rho(\mathbf{x})$ is only defined over the input point cloud. To compute the integral terms in the above equation, we need first extend the definition of the density function to RVCs. To do this, we approximate the density at each vertex of RVC by the average of densities at its k-nearest points in the input point cloud. After triangulating each RVC, the density at any point on a RVC can then be computed by linear interpolation. With the interpolated density function in hand, the integral terms in Equation (3) can be computed numerically by using a quadrature rule on triangles, e.g., the Gaussian quadrature on triangles.

4.3.2. Pulling back

The centroid of each RVC obtained above is on the tangent plane, and may deviate from the underlying surface. We therefore pull the centroids back to the underlying surface. In particular, we find a new plane, still denoted as τ_i, that best fits $\mathbf{x}_i$'s k-nearest points from the input point cloud. Then, $\mathbf{x}_i$ is projected onto the plane τ_i. Once the positions of all $\mathbf{x}_i$ are updated, we will compute the Voronoi cells restricted on the new tangent planes and start the next iteration until convergence or the termination condition is met.

Examples of Lloyd's relaxation are shown in Figure 3 and Figure 5. Figure 5 shows a plot of the CVT objective function

versus the iteration number of Lloyd's relaxation. We can see that the CVT objective function generally decreases, with oscillations occurring after updating tangent planes and RVCs. This is because the estimated tangent planes do not change smoothly with respect to the positions of $\mathbf{x}_i$.

4.3.3. BFGS method

Lloyd's relaxation is a gradient decent algorithm with linear convergence, and there are many more advanced ways of numerical optimizations for CVT energy functions. Among them, the BFGS method [32], which achieves superlinear convergence, is the most commonly used one for accelerating the CVT computation. Traditional Newton's method, though converging quadratically, is not suitable for CVT computation, due to the prohibitive cost of computing and storing the inverse Hessian matrix. Whereas, BFGS belongs to the family of quasi-Newton methods, which uses accumulated gradient information to approximate the inverse Hessian. However, BFGS cannot be directly used for the acceleration of the minimization of our energy function. It can be applied successfully only if the energy function is at least C^2 continuous. As pointed out above, the CVT objective function in Equation (2) is not smooth, as the integral domains change discontinuously. This phenomenon is radically different from the case of the classical CVT energy optimization. As a C^2 continuous function, the CVT energy can be monotonically optimized by both the Llody's relaxation and the BFGS methods.

Although the BFGS method is not suitable for improving the convergence rate of the Lloyd's method in our case, we still resort to it for the minimization of the anisotropic CVT energy function when the Lloyd's method is no longer applicable. To facilitate the discussion, we let $\rho(\mathbf{x}) \equiv 1$ and assume that the distance metric in Equation (2) has the form $\psi(\mathbf{x}, \mathbf{y}) = \|M(\mathbf{x})(\mathbf{x} - \mathbf{y})\|_p^p$, where $\|.\|_p$ denotes the L_p norm and $M(\mathbf{x})$ is a given matrix defining a tensor field $G(\mathbf{x})$, i.e., $G(\mathbf{x}) = M(\mathbf{x})^T M(\mathbf{x})$. Then, the energy function (2) becomes:

$$E(X) = \sum_{i=1}^{n} \int_{V(\mathbf{x}_i) \cap \tau_i} \|M(\mathbf{x})(\mathbf{x} - \mathbf{x}_i)\|_p^p \mathrm{d}\sigma. \tag{4}$$

The minimization of the above energy function leads to an anisotropic sampling with respect to the tensor field $G(\mathbf{x})$. In order to run the BFGS method for the minimization, we make the above energy function a smooth function by fixing the tangent planes during the iterations. That is, we only use the BFGS method to compute the new point positions and their corresponding RVCs on the current tangent planes. After we update the tangent planes, we restart the BFGS method. Thus, it is not necessary to run the BFGS method until convergence as we only need to get an intermediate result. In our experiments, we set the maximum number of iterations for the BFGS method to be 5, i.e. $J_{max} = 5$ in our algorithm as shown in Figure 2. A backtracking line search is used to determine a step size that reduces the objective function.

Computing $E(\mathbf{x})$ and $\nabla E(\mathbf{x})$. The BFGS method requires the evaluations of the energy function $E(\mathbf{x})$ given in Equation (4) and its gradient $\nabla E(\mathbf{x})$. Here we give the closed-form expressions for computing $E(\mathbf{x})$ and $\nabla E(\mathbf{x})$, which are the counterparts of the computation of anisotropic CVTs on surface meshes [29] now applied to tangent planes. We refer the readers to [29, 33] for the detailed derivations.

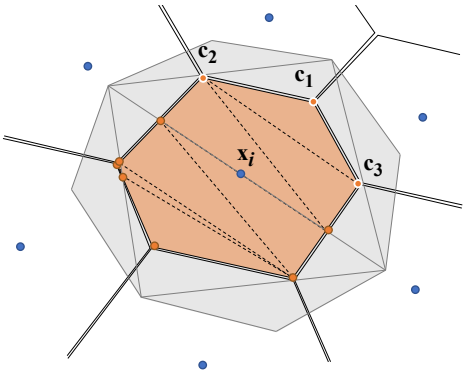


Figure 6: Computing $E(\mathbf{x})$ and $\nabla E(\mathbf{x})$ on a restricted Voronoi cell. The RVC of $\mathbf{x}_i$ (marked in orange) consists of a set of triangles obtained by clipping a octagon (marked in gray), with bisector planes between $\mathbf{x}_i$ and its neighbor points.

Recall that the RVCs are the intersections between the 3D Voronoi cells and the octagons lying on tangent planes and centered at the resampling points. Each RVC is a set of triangles obtained by using the clipping method [28]. Suppose T is one triangle of the RVC of $\mathbf{x}_i$ with vertices $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$ as shown in Figure 6. We assume that the matrix $M(\mathbf{x})$ is constant over T, denoted by M_T, and is computed by averaging the corresponding matrices at the vertices of T. Then the energy $E(\mathbf{x})$ over T is given by:

$$\begin{aligned} E_T(X) &= \int_T \|M_T(\mathbf{x} - \mathbf{x}_i)\|_p^p \, \mathrm{d}\sigma \\ &= \frac{|T|}{\binom{2+p}{2}} \sum_{\alpha+\beta+\gamma=p} \overline{\mathbf{u}_1^\alpha * \mathbf{u}_2^\beta * \mathbf{u}_3^\gamma}, \end{aligned}$$

$$\text{where} \begin{cases} \mathbf{u}_j &= M_T\left(\mathbf{c}_j - \mathbf{x}_i\right), \\ \mathbf{u}_1 * \mathbf{u}_2 &= \left[x_1 x_2, y_1 y_2, z_1 z_2\right]^t, \\ \mathbf{u}^\alpha &= \mathbf{u} * \mathbf{u} * \cdots * \mathbf{u} \, (\alpha \text{ times}), \\ \overline{\mathbf{u}} &= x + y + z. \end{cases}$$

The gradient of $E_T(X)$ relative to X is obtained by the chain rule:

$$\frac{dE_T(\mathbf{x}_i, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)}{dX} = \frac{dE_T}{d\mathbf{x}_i} + \frac{dE_T}{d\mathbf{c}_1}\frac{d\mathbf{c}_1}{dX} + \frac{dE_T}{d\mathbf{c}_2}\frac{d\mathbf{c}_2}{dX} + \frac{dE_T}{d\mathbf{c}_3}\frac{d\mathbf{c}_3}{dX}.$$

Finally, we get the energy $E(\mathbf{x})$ and its gradient $\nabla E(\mathbf{x})$ by summing the contributions $E_T(X)$ and $\nabla E_T(\mathbf{x})$ of each triangle T.

5. Implementation and Results

In this section, we present the results of our CVT-based method applied to isotropic/anisotropics resampling of point clouds and show its application to surface reconstruction. We experimented with both synthetic and real world datasets. In order to evaluate our resampling results, we also compare our method with the state-of-the-art approaches, including the WLOP method [14], and the graph Laplacian based

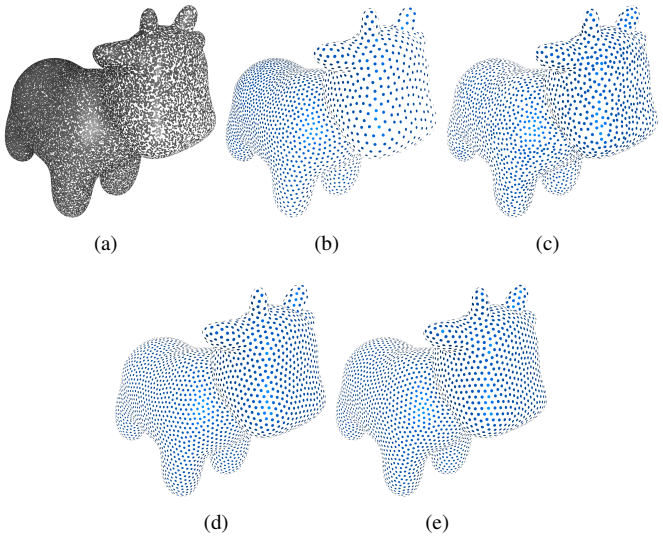


Figure 7: Comparisons with the state-of-the-art methods. (a) Input point cloud with 110K points; (b) result from the WLOP method [14], $\sigma = 0.182$; (c) result from the k-means method, $\sigma = 0.124$; (d) result from the graph Laplacian based method [22], $\sigma = 0.037$; and (e) our CVT-based result, $\sigma = 0.052$.

method [22]. All tests are executed on an Intel Core i7-6700k with 16GB RAM, running the multi-threading implementation. The Core i7-6700k is a quad core processor with 2 simultaneous multi-threading contexts per core.

5.1. Uniform resampling

Let the metric $\psi(\mathbf{x}, \mathbf{y})$ be the squared Euclidean distance between two points $\mathbf{x}$ and $\mathbf{y}$, i.e., $\psi(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$, and density function $\rho(x) \equiv 1$, then the minimization of the energy function in Equation (2) leads to a uniform point cloud. We compare our CVT-based method to the aforementioned state-of-the-art approaches, using the cow model shown in Figure 7. The number of the input data points is 110K, while the output point number is set to 5K. To achieve fair results, we use implementations provided in [14, 22] and conduct all the experiments on the same machine. To give a rough quantitative measure of the uniformity of points, we adopt the standard deviation of distances to the nearest neighbors at resampling points, which is denoted by σ. The average value of the distances is normalized. We also implement a discrete version of the Lloyd's method, which moves each resampling point to the center of its associated cluster of the data points as the k-means algorithm [8] does. From Figure 7 it can be seen that the WLOP method [14] and k-means method fail to generate uniform point distributions, as they suffer from the non-uniform distribution of the input point cloud. Both the graph Laplacian based method [22] and our CVT-based method is capable of generating uniform resampling results. Our method is much more efficient than the graph Laplacian based method [22], as we will show in the next paragraph.

Running time comparison. Our iterative resampling algorithm involves repeatedly querying the k-nearest neighbors of a resampling point among the input point cloud. To optimize the queries, the input point cloud is organized in a kd-tree by using the NanoFLANN library [30, 31] in the initialization stage,

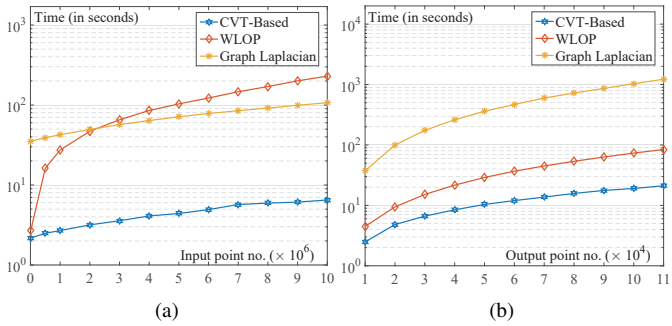


Figure 8: (a) Running time against the number of input points, ranging from 10K to 10M, with a fixed output point number ($m = 10K$); and (b) running time against the number of output points, ranging from 10K to 110K, with a fixed input point number ($n = 110K$).

of which time complexity is $O(n \log(n))$ with respect to the number of input data n. Then, in each iteration, the k-nearest neighbors of each resampling point can be found in $O(\log(n))$ time. In total, the running time of our CVT-based algorithm is $O(n \log(n) + I \cdot m \cdot \log(n))$, where n is the number of the input points, I is the iteration number, and m is the number of resampling points. Note that, the time complexity is $O(n \log(n))$ with respect to the number of input points n and linear with respect to the number of output points m. We might think that the computation time mainly depends on the number of input points at the first glance, as n is usually much larger than m in practical applications. However, through experiments, we can show that the total computation time of our algorithm is mainly determined by the number of output points m when the iteration number is fixed. The kd-tree construction is very efficient in practice by using the NanoFLANN library, whose contribution to the total running time is small. We test our algorithm on a sequence of point cloud datasets with different resolutions while fixing the output point number ($m = 10K$) and iteration number ($I = 35$). The number of points in the input point cloud varies from $10K$ to $10M$. The plot of running time against the input point number is shown in Figure 8(a). When the number of the input points increases from $10K$ to $10M$, the running time for our CVT-based method, the WLOP method [14], and the graph Laplacian method [22] is increased by 4.3, 226.6, and 71.7 seconds, respectively. We also generate resampling results with different sizes from the same point cloud. Figure 8(b) reports the running time of our CVT-based method, the WLOP method [14], and the graph Laplacian based method [22]. We can see that our algorithm is much more efficient than the other two methods when the resampling point number m increases.

5.2. Weighted resampling

Given an input point cloud and a user-specified density function $\rho(\mathbf{x})$ defined over it, we can also generate a resampling result adapted to the given density function. In our experiments, the density function is set to $\rho(\mathbf{x}) = |\bar{\kappa}_1(\mathbf{x})| + |\bar{\kappa}_2(\mathbf{x})|$, where $\bar{\kappa}_i(\mathbf{x}) = \max(|\kappa_i(\mathbf{x})|, 10^{-4})$ and $|\kappa_i(\mathbf{x})|$, $i = 1, 2$, are the two principal curvatures at $\mathbf{x}$, respectively. Here, the principal curvatures are truncated by 10^{-4} to avoid the instability in numerical calculation. We estimate the curvatures at each data

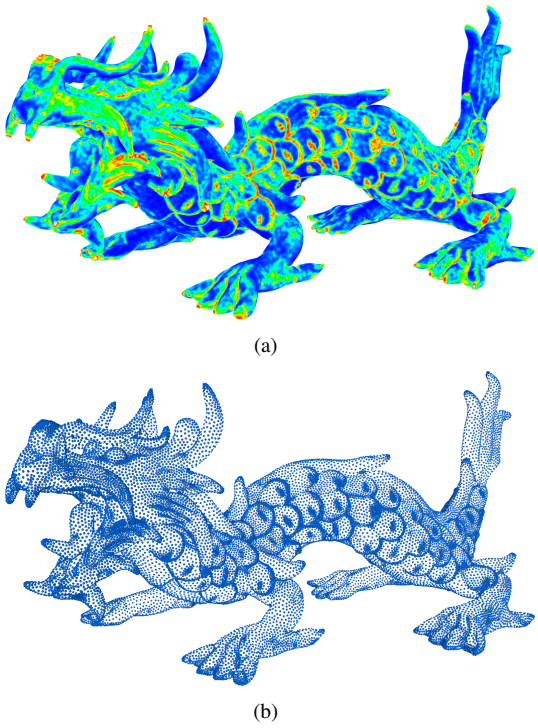


Figure 9: Weighted resampling. (a) Input point cloud (3M points) with color-coded density function derived from the discrete curvatures of the point cloud (red indicates the maximum value and blue indicates the minimum value); and (b) weighted resampling result (50K points) adapted to the density function, obtained after 35 Lloyd' iterations in 16.4 seconds.

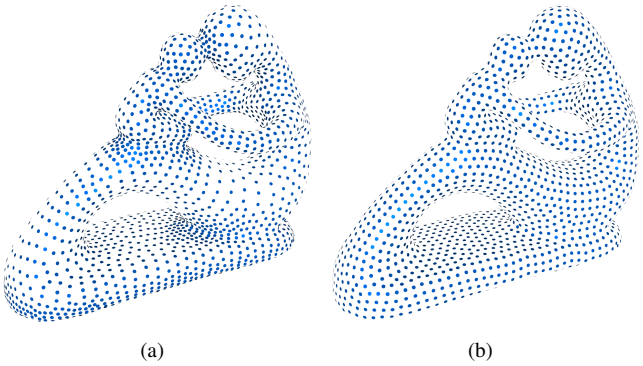


Figure 10: Anisotropic resampling. (a) Resampling result adapted to curvature tensor field; and (b) L_8 resampling result.

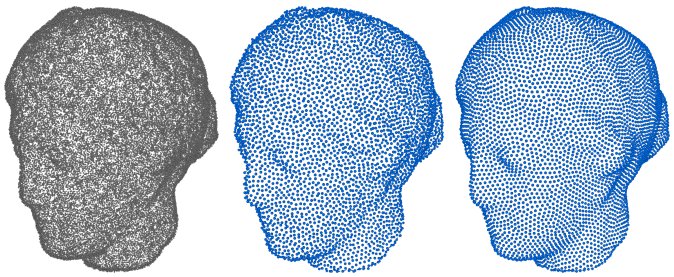


Figure 11: Noise depression. (a) A noisy input with 180K points; (b) result with 10K points from the graph Laplacian based method [22]; and (c) our result with 10K points.

point using CGAL's implementation [34]. Moreover, we use the Laplacian filter to smooth the discrete curvatures such that the density function varies smoothly. Figure 9 shows an adaptive resampling result of scan data of a dragon model from [35]. Our method resamples it from 3M to 50K points in 16.4 seconds. Our resampling result adapts to the given density function by distributing fewer (more) points in flat (curved) regions.

5.3. Anisotropic resampling

To obtain anisotropic resampling results, we minimize the energy function in Equation (4), where the tensor field $G(\mathbf{x}) = M^T(\mathbf{x})M(\mathbf{x})$ at each point of the input data should be specified. In our experiments, we use the curvature tensors as the tensor field. Let $U_i(\mathbf{x})$ be the principal directions and $N(\mathbf{x})$ be the surface normal at $\mathbf{x}$. Then, we set $M(\mathbf{x}) = Diag(\sqrt{\kappa_1(\mathbf{x})}, \sqrt{\kappa_2(\mathbf{x})}, 0)Q(\mathbf{x})^T$, where $\kappa_i(\mathbf{x})$ are the principal curvatures, and $Q(\mathbf{x}) = [U_1(\mathbf{x}), U_2(\mathbf{x}), N(\mathbf{x})]$. Here, the curvatures, principal directions and surface normals are estimated by the aforementioned CGAL's implementation [34]. Figure 10 shows an example of anisotropic resampling using an anisotropic L_2 norm (i.e., $p = 2$ in Equation (4)), where resampling points tend to gather along the high-curvature directions and vice versa. It also demonstrates the feasibility of the combination of our alternative optimization framework and the BFGS method for energy minimization.

Note that if the axes of anisotropy have the same length, the minimizer of the energy function in Equation (4) under L_2 norm corresponds a "honeycomb" pattern of RVCs. Nevertheless, minimizing Equation (4) with a large value of p leads to rectangle-shaped RVCs [29]. As a consequence, the resampling points have a lattice-like distribution, with lattice directions aligning with the tensor field. Figure 10(b) shows a point resampling result using an L_8-CVT energy function with $M(\mathbf{x}) = Diag(1, 1, 0)[U_1(\mathbf{x}), U_2(\mathbf{x}), N(\mathbf{x})]^T$. Though the lengths of the two anisotropic axes are set to the same value, the directions of the axes guide the point distribution, which is entirely different from the L_2 case.

5.4. Noise depression

During optimization the resampling points are repeatedly projected onto a plane that best fits their k-nearest neighbors at the input point cloud. This procedure is similar to that in a standard point cloud smoothing approach, in which the points are projected onto the average plane of neighbors as one round of smoothing. Hence, our method presents a good anti-noise ability when the input data contains moderate levels of noise, as shown in Figure 11. In contrast, the result from the graph Laplacian based method [22] is more easily affected by noise. For input data with large scale noise, we can apply our algorithm after several rounds of smoothing to achieve visually more pleasant results.

5.5. Boundary handling

Note that, by simply following the procedure presented above without any special processing on the boundaries, the

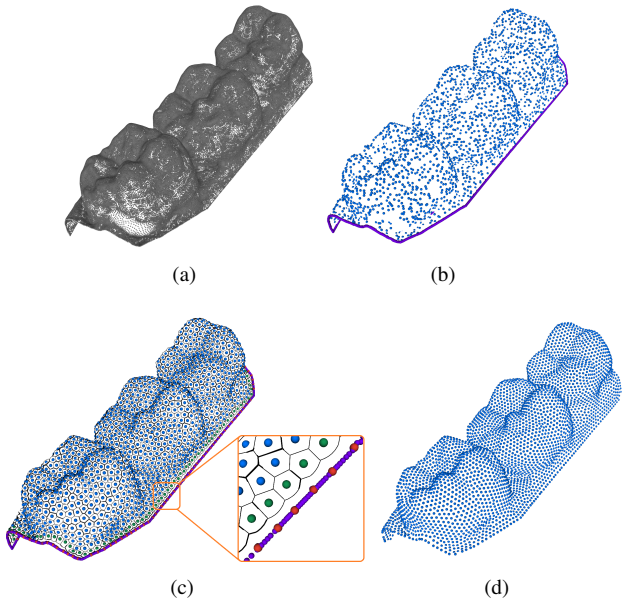


Figure 12: Boundary handling. (a) Input point cloud; (b) detected boundary points (in purple) and initial resampling points (in blue); (c) resultant RVCs after Lloyd's relaxation and projected points (in red) on the boundary; and (d) final resampling result.

RVC of a resampling point at the boundary usually contains a region beyond the scope of the underlying surface. Thus, the corresponding centroid tends to drift away from the underlying surface. Here we propose a simple method to handle this problem.

First, the boundary points of the input point cloud are extracted, by using the boundary point detection method provided in the point cloud library (PCL) [36]. The basic steps are as follows: (1) For each point $\mathbf{p}$ in the point cloud P, find the plane $\tau(\mathbf{p})$ which best fits its k-nearest neighbors from P. (2) The point $\mathbf{p}$ and its k-nearest neighbors $N_k(\mathbf{p})$ are projected onto the plane $\tau(\mathbf{p})$. (3) Obtain k vectors on $\tau(\mathbf{p})$ by connecting the projection of $\mathbf{p}$ with the projections of its k-nearest neighbors, respectively. (4) Find the maximum turning angle between each pair of adjacent vectors. If the maximum turning angle is greater than a threshold, then $\mathbf{p}$ is considered to be a boundary point. Figure 12(b) shows the boundary points detected by the above method. Second, the boundary points are considered as virtual resampling points, which participate in the RVC computation but are fixed during optimization. By this way, RVCs of real resampling points and their centroid will always stay within the regions covered by the underlying surface, see Figure 12(c). Finally, as a post-processing step, we further resample the boundary points to obtain a consistent point distribution on the boundary as follows. For each pair of adjacent resampling points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are adjacent to the boundary (namely, their RVCs share a common bisector with the virtual resampling points), see Figure 12(c), the middle point of the segment $\mathbf{x}_i\mathbf{x}_j$ is projected onto the line that best fits its k-nearest points from the boundary point set. The projected point is included in the resampling point set only if its distance to the nearest resampling point is greater than $\|\mathbf{x}_i - \mathbf{x}_j\|/2$.

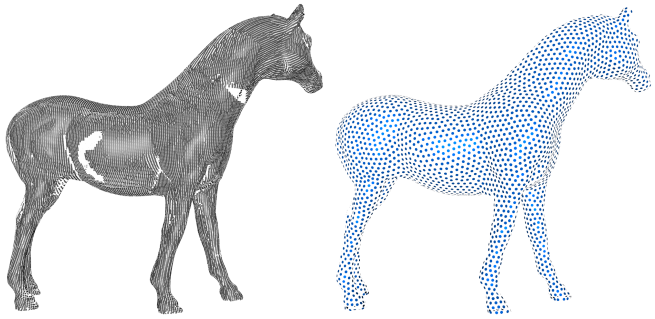


Figure 13: Hole filling. (a) Input point cloud with holes; and (b) resampling result with holes filled.

5.6. Hole filling

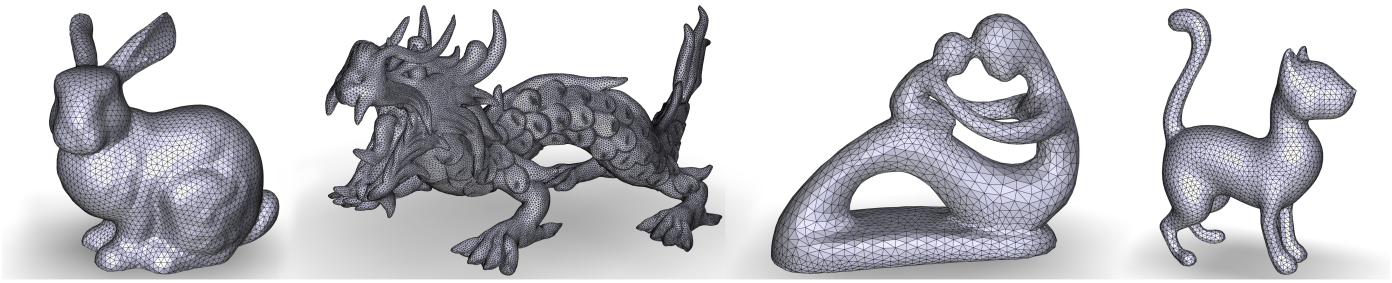
Our algorithm can automatically infer substitute regions for the missing parts and place an appropriate number of resampling points on it. As based on local linear approximation, our algorithm works well for holes which can be repaired by a simple surface extrapolation, as shown in Figure 13, but it has problem in recovering highly curved regions. A possible solution to improve the hole filling results is to use high-order surfaces, instead of the linear plane, to locally approximate the input data. Another difficulty is that it is hard to distinguish hole boundaries from the open boundaries that the users want to preserve. For simplicity, we just let the users decide which holes need to be filled.

5.7. Surface reconstruction

As a by-product, the RVCs generated by our algorithm gives a piecewise linear approximation to the underlying surface, as shown in Figures 5 and 12. This polygon soup representation may be sufficient for the application of geometric proximity analysis and 3D visualization. Furthermore, the obtained polygon soup can easily be converted into a widely used triangular mesh, using the duality between RVCs and restricted Delaunay triangulation. Figure 14 shows the reconstructed surface meshes of point clouds used in this paper. Note that the surface meshes constructed by using only the duality of RVCs and restricted Delaunay triangulation may contain holes and non-manifold faces. A clean and orientable 2-manifold mesh can be recovered by using a manifold extraction algorithm [37]. Based on our resampling results, we get the isotropic/anisotropic meshing results of the point clouds directly, which is more efficient than reconstructing the surface from the point cloud first and then remeshing the surface.

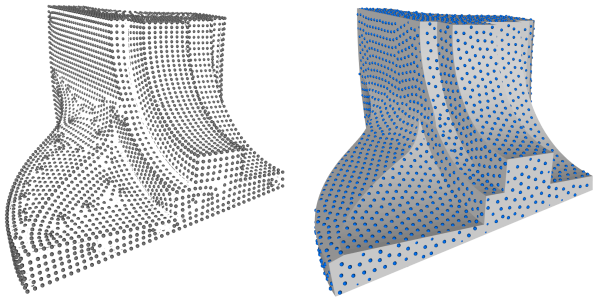
6. Limitation and Discussion

We have generalized the definition of the CVT energy function to point clouds. Based on the generalized energy function, we propose an efficient algorithm for energy function minimization and provide a versatile CVT-based framework on which we can generate uniform/adaptive or isotropic/anisotropic point resampling results.



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

Figure 14: Surface reconstruction using the duality between RVCs and restricted Delaunay triangulation. (a) Uniform meshing; (b) non-uniform meshing; (c) anisotropic meshing; and (d) L_p meshing.



(a) (b)

Figure 15: An example with sharp features. (a) Input point cloud with hidden sharp features; and (b) resampling result, with the underlying surface rendered for a better visualization of the locations of the resampling points.

The proposed algorithm assumes that the point cloud is sampled from a smooth surface and is free of outliers and large scale noise. We locally approximate the underlying surface using planes. Hence, if the number of resampling points is too small, especially for curved regions, RVCs will give a poor approximation to the underlying surface, leading to unsatisfying resampling results. Besides, our algorithm does not perform well on models with sharp features, as the estimated tangent planes at the sharp features are inaccurate. Figure 15 shows a resampling result of a CAD model, where the sharp features are not well preserved. If the sharp features are well sampled and can be clearly detected, we can handle the sharp features in the same fashion as we do for boundaries. However, it is hard to detect clear sharp features from a point cloud in practice, due to the inherent noise and incompleteness in the acquired data. Integrating robust feature preserving techniques into our resampling framework will be our future work.

Acknowledgements

The research of Zhonggui Chen and Juan Cao was supported by the National Natural Science Foundation of China (No. 61472332, 61572020, 61728206), the Natural Science Foundation of Fujian Province of China (No. 2018J01104), and the program of China Scholarship Council. The research of Cheng Wang was supported in part by the National Natural Science Foundation of China (No. U1605254). The research of Yongjie Jessica Zhang was supported in part by the PECASE Award N00014-16-1-2254 and NSF CAREER Award OCI-1149591.

References

[1] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, L. Xiao, A review of algorithms for filtering the 3D point cloud, Signal Processing: Image Communication 57 (Supplement C) (2017) 103 – 112.

[2] P. S. Heckbert, M. Garland, Survey of polygonal surface simplification algorithms, Multiresolution Surface Modeling Course, SIGGRAPH 97.

[3] D. P. Luebke, A developer's survey of polygonal simplification algorithms, IEEE Computer Graphics and Applications 21 (3) (2001) 24–35.

[4] M. Pauly, M. Gross, L. P. Kobbelt, Efficient simplification of point-sampled surfaces, in: Proceedings of IEEE Conference on Visualization, 2002, pp. 163–170.

[5] H. Song, H.-Y. Feng, A global clustering approach to point cloud simplification with a specified data reduction ratio, Computer-Aided Design 40 (3) (2008) 281–292.

[6] Y. Miao, R. Pajarola, J. Feng, Curvature-aware adaptive re-sampling for point-sampled geometry, Computer-Aided Design 41 (6) (2009) 395–403.

[7] Z. Yu, H.-S. Wong, H. Peng, Q. Ma, ASM: an adaptive simplification method for 3D point-based models, Computer-Aided Design 42 (7) (2010) 598–612.

[8] B.-Q. Shi, J. Liang, Q. Liu, Adaptive simplification of point cloud using k-means clustering, Computer-Aided Design 43 (8) (2011) 910–922.

[9] X. Ma, R. J. Cripps, Shape preserving data reduction for 3D surface points, Computer-Aided Design 43 (8) (2011) 902–909.

[10] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Computing and rendering point set surfaces, IEEE Transactions on Visualization and Computer Graphics 9 (1) (2003) 3–15.

[11] S. Liu, K.-C. Chan, C. C. L. Wang, Iterative consolidation of unorganized point clouds, IEEE Computer Graphics and Applications 32 (3) (2012) 70–83.

[12] J. Wang, K. Xu, L. Liu, J. Cao, S. Liu, Z. Yu, X. D. Gu, Consolidation of low-quality point clouds from outdoor scenes, Computer Graphics Forum 32 (5) (2013) 207–216.

[13] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, ACM Transactions on Graphics (TOG) 26 (3) (2007) 22.

[14] H. Huang, D. Li, H. Zhang, U. Ascher, D. Cohenor, Consolidation of unorganized point clouds for surface reconstruction, ACM Transactions on Graphics 28 (5) (2009) 1–7.

[15] B. Liao, C. Xiao, L. Jin, H. Fu, Efficient feature-preserving local projection operator for geometry reconstruction, Computer-Aided Design 45 (5) (2013) 861–874.

[16] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, H. Zhang, Edge-aware point set resampling, ACM Transactions on Graphics 32 (1) (2013) 1–12.

[17] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, M. Wimmer, Continuous projection for fast L_1 reconstruction, ACM Transactions on Graphics 33 (4) (2014) 47.

[18] A. C. Öztireli, M. Alexa, M. Gross, Spectral sampling of manifolds, ACM Transactions on Graphics (TOG) 29 (6) (2010) 168.

[19] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, W. Wang, Variational blue noise sampling, IEEE Transactions on Visualization and Computer Graphics 18 (10) (2012) 1784–1796.
[20] J. Chen, X. Ge, L.-Y. Wei, B. Wang, Y. Wang, H. Wang, Y. Fei, K.-L. Qian, J.-H. Yong, W. Wang, Bilateral blue noise sampling, ACM Transactions on Graphics (TOG) 32 (6) (2013) 216.
[21] D.-M. Yan, J.-W. Guo, B. Wang, X.-P. Zhang, P. Wonka, A survey of blue-noise sampling and its applications, Journal of Computer Science and Technology 30 (3) (2015) 439–452.
[22] C. Luo, X. Ge, Y. Wang, Uniformization and density adaptation for point cloud data via graph Laplacian, Computer Graphics Forum 37 (1) (2018) 325–337.
[23] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tessellations: applications and algorithms, SIAM Review 41 (1999) 637–676.
[24] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.
[25] Y. Liu, W. Wang, F. Sun, D. M. Yan, L. Lu, C. Yang, Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, C. Yang, On centroidal Voronoi tessellationenergy smoothness and fast computation, ACM Transactions on Graphics (TOG) 28 (4) (2009) 101.
[26] P. Alliez, É. C. De Verdière, O. Devillers, M. Isenburg, Centroidal Voronoi diagrams for isotropic surface remeshing, Graphical Models 67 (3) (2005) 204–231.
[27] D. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang, Isotropic remeshing with fast and exact computation of restricted Voronoi diagram, Computer Graphics Forum 28 (5) (2009) 1445–1454.
[28] B. Lévy, N. Bonneel, Variational anisotropic surface meshing with Voronoi parallel linear enumeration, Proceedings of The 21st International Meshing Roundtable (2013) 349–366.
[29] B. Lévy, Y. Liu, L_p centroidal Voronoi tessellation and its applications, ACM Transactions on Graphics 29 (4) (2010) 1–11.
[30] J. L. Blanco, P. K. Rai, NanoFLANN: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) wih kd-trees, `https://github.com/jlblancoc/nanoflann` (2014).
[31] M. Muja, D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2227–2240.
[32] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, 2006.
[33] G. Parigi, M. Piastra, Gradient of the objective function for an anisotropic centroidal Voronoi tessellation (CVT)-a revised, detailed derivation, arXiv:1408.5622 (2014).
[34] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, Computer Aided Geometric Design 22 (2) (2005) 121–146.
[35] Stanford computer graphics laboratory.
URL `http://graphics.stanford.edu/data/3Dscanrep/`
[36] R. B. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in: Proceedings of the IEEE International Conference on Robotics and Automation, 2011, pp. 1–4.
[37] D. Boltcheva, B. Lévy, Surface reconstruction by computing restricted Voronoi cells in parallel, Computer-Aided Design 90 (Supplement C) (2017) 123–134, SI:SPM2017.