# Accelerated Lloyd's Method for Resampling 3D Point Clouds

Yanyang Xiao, Tieyi Zhang, Juan Cao, *Senior Member, IEEE*, and Zhonggui Chen, *Senior Member, IEEE*

***Abstract*—We present an efficient approach to generating uniformly distributed resampling points of raw 3D point clouds. A key contribution for making such a resampling method both practical and efficient is the construction of the centroidal Voronoi tessellation on the given point cloud efficiently achieved by applying the proposed Anderson-accelerated Lloyd's method. The calculations involved in the method are mainly carried out over a group of locally approximated quadratic surfaces, instead of directly on the given point cloud, providing us a great advantage in filtering out the affection of distribution of original points on output results. Once the resampling points are initialized, the resampling quality can be improved progressively by optimizing resampling points and updating the local approximated surfaces. In addition, by restricting the movement of resampling points, we can deal with unclosed point clouds without any boundary detection. Our approach outperforms existing resampling methods in generating uniform results, and extensive experiments are conducted to demonstrate its efficacy.**

***Index Terms*—Anderson acceleration, Lloyd's method, point clouds, resampling.**

## I. Introduction

POINT clouds are one of the most popular representations of 3D objects and environments. Owing to its simplicity and flexibility, they have been widely used in various applications, including mobile mapping [1], virtual reality [2], and heritage digitalization [3]. With the rapid development of 3D sensing techniques and devices, the acquisition of point clouds is becoming much easier and more convenient.

Manuscript received 30 July 2022; revised 21 October 2023 and 8 February 2024; accepted 9 May 2024. Date of publication 27 May 2024; date of current version 21 February 2025. The work of Yanyang Xiao was supported in part by the National Natural Science Foundation of China under Grant 62102174 and in part by the Jiangxi Provincial Natural Science Foundation, China under Grant 20232BAB212010. The work of Juan Cao and Zhonggui Chen was supported in part by the National Natural Science Foundation of China under Grant 61972327, Grant 62272402, and Grant 62372389, in part by the Special Fund for Key Program of Science and Technology of Fujian Province, China under Grant 2022YZ040011, in part by the Natural Science Foundation of Fujian Province, China under Grant 2022J01001, and in part by the Fundamental Research Funds for the Central Universities, China under Grant 20720220037. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Mingqiang Wei. *(Corresponding author: Zhonggui Chen.)*

Yanyang Xiao is with the School of Mathematics and Computer Sciences, Nanchang University, Nanchang 330031, China (e-mail: xiaoyanyang@ncu.edu.cn).

Tieyi Zhang and Zhonggui Chen are with the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: rysx.zty@gmail.com; chenzhonggui@xmu.edu.cn).

Juan Cao is with the School of Mathematical Sciences, Xiamen University, Xiamen 361005, China (e-mail: juancao@xmu.edu.cn).

Digital Object Identifier 10.1109/TMM.2024.3405664

However, due to the device accuracy limitations and sensing conditions, most scanned point clouds usually suffer from artifacts, such as redundancy, noises, outliers, and uneven point distributions, challenging storage and processing in downstream applications. Although point clouds can be organized into other data structures, such as octree encoding [4], to reduce these artifacts significantly, many details will be lost and the distortion error will be much higher. On the other hand, it is required to generate denser points from given sparse points, or recover original points from compressed point clouds, i.e., upsampling [5], [6], [7]. Therefore, the conversion of the given point clouds into noise-free, uniform, and low-distortion results by resampling has become an important preprocessing stage for various point cloud-based tasks. An example of resampling is given in Fig. 1, where a large-scale point cloud with varying densities is uniformly resampled with a sampling rate of 2%.

Many studies have focused on point cloud resampling, most of which cannot generate uniform results from arbitrary raw data. The simplification-based methods (e.g., [8], [9], [10], [11]) select a subset of original points as the result and aim to preserve as much of the underlying shape represented by the point cloud as possible, while usually caring less about the point distribution in the subset. By contrast, some methods (e.g., [12], [13], [14]) directly act on the original points, so that their outputs are severely affected by the uneven distribution of original points, resulting in poor uniformity.

To attenuate the influence of the original distribution as much as possible, a class of proxy-based methods attempts to extract the underlying surface of the input points as the resampling domain. For instance, Li and Sun [15] construct a $k$-nearest neighbors (KNN) graph from an over-segmentation of the given point cloud for uniformly sampling new points, while Chen et al. [16] and Han et al. [17] replace the input point cloud with a set of local planes, on which the resampling points can be then optimized. A more straightforward way is to reconstruct an approximating surface of the input points, which is then used for the subsequent optimization [18].

The goal to generate randomized point distribution as uniform as possible is similar to blue-noise sampling [19], [20], [21], [22], [23], [24], [25], but it commonly runs on given continuous domains. Blue-noise sampling can provide benefit for many applications and has attracted considerable attention in computer graphics. One of its primary objectives is to avoid repetitive patterns in the resultant point set, which is usually ignored in point cloud resampling tasks. We focus on producing points with high uniformity from given point clouds. Among these sampling

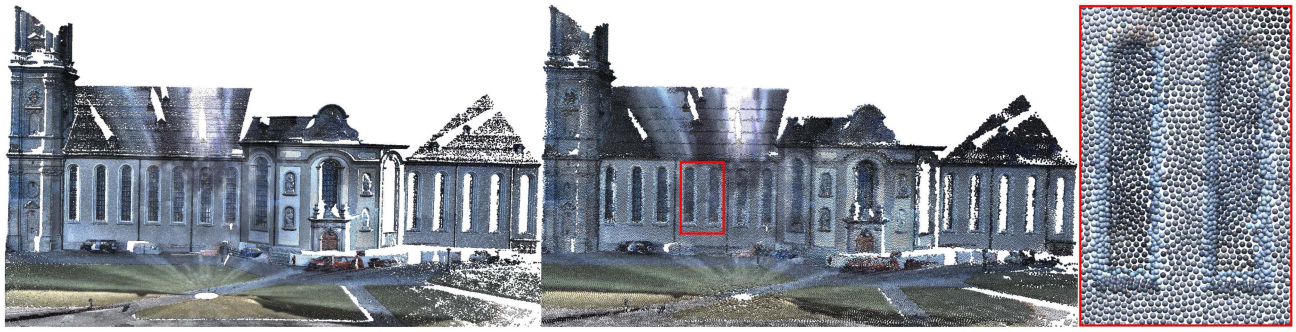


Fig. 1. Our method generates resampling points with high uniformity from a raw point cloud with varied densities. (a) Input with 5 M points; (b) Output with 100 K points and a close-up view of sampling distribution highlighted in the red box.

approaches, centroidal Voronoi tessellation (CVT) [26], [27] is one of the most frequently involved methods. The CVT-based methods typically minimize a tailored energy function to optimize the point locations so that the point set becomes progressively uniform. The minimization can be accomplished by different methods, where Lloyd's method [28] is the most commonly used. However, the CVT cannot be directly computed on discrete domains, such as point clouds. Benefiting from the construction of the abovementioned local planes, Chen et al. [16] have successfully extended the computation of CVT to point clouds.

This paper aims to produce uniform and feature-preserving resampling points for arbitrary raw data. To achieve this goal, we convert the resampling task into the efficient computation of CVT on an intermediate domain consisting of quadratic patches. Given that the plane causes high distortion error when representing points with widely differing normals, we introduce a similar domain consisting of a set of quadratic surfaces to better capture data features. Owing to the linear convergence of Lloyd's method, we incorporate the Anderson acceleration technique [29] to speed up the optimization of resampling points. The specific contributions of this paper are as follows.

- We propose a novel resampling method by generating CVTs on a set of quadratic surfaces that locally approximate the input point cloud. The resampling result has a uniform point distribution, and the feature of input point clouds are preserved.
- We tailor an efficient algorithm by combining Lloyd's method and the Anderson acceleration technique to speed up the computation of resampling, significantly improving the resampling quality within a given number of iterations.
- We set a move distance limit for each resampling point to prevent the resampling point on the boundary from deviating too much from the original data. Hence, our algorithm can handle boundaries and fill small holes in point clouds without any preprocessing.

The remainder of this paper is organized as follows. After a short review of related work in Section II, we describe the definition of our CVT-based energy function on point clouds in Section III and the efficient resampling algorithm in Section IV. Several experiments are demonstrated to evaluate our method in Section V. Finally we conclude the paper in Section VI.

## II. Related Work

### A. Resampling Methods

A thorough survey of point cloud resampling methods can be found in [30]. Here, we mainly review optimization- or proxy-based ones proposed in recent years.

Lipman et al. [12] presented a locally optimal projection (LOP) operator to generate results by minimizing the distance between resampling and original points and maximizing the space between resampling points. However, the point density of the resampling result generated by the LOP method remains similar to that of the input point cloud. By introducing density weights, Huang et al. [13] proposed the weighted LOP (WLOP) to deal with non-uniform raw data. Other variants of LOP have been developed to meet user requirements, such as the feature-preserving [31] and the edge-aware versions [32]. Given that LOP and its variants act directly on the given points, they are more or less affected by the non-uniform distribution of the raw data, hence difficult to produce highly uniform resampling points.

To remove the influence of the original point distribution, the proxy-based methods take an approximate surface of the given point cloud as the resampling domain. Li and Sun [15] build a KNN graph from an over-segmentation of the raw data, which is followed by a LOP-like optimization. Lv et al. [18] reconstruct a surface with global continuity as the domain for resampling. Some methods approximate the point clouds with a set of local planes. Song and Feng [8] selecte a subset of original points and update them with the closest points to the corresponding tangent planes, but the resulting point distribution is highly input-dependent. Chen et al. [16] focus on isotropic and anisotropic resampling of point clouds on smooth surfaces by computing CVTs on local fitting planes that are similarly employed in [17]. However, sharp features are ubiquitous in point clouds, and using planes to represent local points with such features is not reasonable. Therefore, we choose quadratic surfaces to construct the resampling domain to better capture features of the raw data. Given the power of the CVT method in generating uniform point sets, we further extend it to this domain for resampling arbitrary point clouds.

Since PointNet [33] successfully extends convolutional networks to unstructured datasets, there has been an increasing

amount of research on point cloud processing based on deep learning, a part of which involves denoising and upsampling [5], [6], [7], [34], [35], [36], [37], [38]. Rakotosaona et al. [36] propose the PointCleanNet to estimate correction vectors that project noisy points onto the original clean surfaces. PU-Dense [6] uses sparse networks to reconstruct the geometry of upsampled point cloud via progressive rescaling and multi-scale feature extraction. Chen et al. [7] represent a point cloud via its gradient field, which is enforced to be continuous and can be estimated by deep learning. Differently, Hermosilla et al. [35] treat the convolution as a Monte Carlo integration problem, making the learning robust to non-uniform distributions. These deep learning-based methods are usually task-driven, ignoring the quality of resultant points, which is not the same as the purpose of our method to obtain uniform points.

### B. CVT Methods

For a given continuous domain, CVT is a special case of Voronoi tessellation whose generating points coincide with the centroids of corresponding Voronoi regions. CVT can also be defined as the minimum of an energy function [26]. Lloyd's method [28], a type of fixed-point iteration that converges linearly, is one of the most commonly used methods for searching the minimum. Liu et al. [27] pointed out that the CVT energy function is $C^2$ continuous, and they apply a quasi-Newton method to speed up the minimization. In field of computer graphics, the CVT method is a popular tool to produce uniform point distributions, and has been extensively studied in a wide range of applications, such as the surface and volume remeshing [39], [40]. Chen et al. [16] successfully extended the CVT energy function to point clouds for isotropic and anisotropic resampling. However, their method is not feature-sensitive and not applicable to point clouds with complex boundaries unless the boundaries are extracted and processed in advance. We provide a more robust CVT-based resampling method to address these issues. In addition, the quasi-Newton methods that require high continuity of energy functions are not applicable for accelerating the CVT computation on point clouds. Thus, we incorporate the Anderson acceleration technique [29] to speed up the convergence.

### C. Anderson Acceleration

Anderson acceleration is a technique for accelerating the convergence of a fixed-point iteration and has been used in a broad context for a variety of algorithms. It was originally invented by Anderson [29] for the iterative solution of nonlinear integral equations. Coincidentally, Pulay [41] independently proposed this technique for self-consistent field iteration in electronic structure computations, which is also known in quantum chemistry as Pulay mixing or direct inversion in the iterative subspace. It was also presented by Washio and Oosterlee [42] as a Krylov subspace acceleration technique for solving nonlinear partial differential equations. The core idea of Anderson acceleration is to weight several previous iterates to derive a new one in each iteration, and it does not require the computation or approximation of Jacobians or Jacobian-vector products, providing an advantage over Newton methods. It can also be used as a type of quasi-Newton method for solving nonlinear equations [43]. Toth and Kelley [44] proved its local convergence for contractive fixed-point iterations. Anderson acceleration has been widely used in various applications, including numerical problems [45], fluid dynamics [46], geometry optimization and physics simulation [47]. In this paper, we extend Anderson acceleration to the computation of resampling points.

## III. CVT on Point Clouds

Given an arbitrary unstructured 3D point cloud $P = \{\mathbf{p}_i\}_{i=1}^N$, our goal is to produce resampling points that are as uniform and feature-preserving as possible. We present an efficient method to achieve the goal by obtaining a CVT on a set of local quadratic surfaces which approximate the input point cloud. First, we need to extend the definition of the CVT energy function to point clouds.

### A. Energy Function

Given a set of seed points $X = \{\mathbf{x}_i\}_{i=1}^n$ on a continuous surface $\Omega \subset R^3$, the Voronoi diagram divides the space $R^3$ into an equal amount of smaller regions, i.e., Voronoi cells $\{V_i\}_{i=1}^n$, where

$$V_i = \left\{\mathbf{x} \in R^3 \mid \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \forall j \neq i\right\}.$$

By restricting the Voronoi cells to the domain $\Omega$, we obtain a partition of the domain, and each sub-region $\Omega_i$ is called restricted Voronoi cell (RVC). In particular, $\Omega_i = V_i \cap \Omega$ for $\mathbf{x}_i$.

The CVT is a special type of Voronoi tessellation in which each seed point coincides with the centroid of the corresponding Voronoi region. From the variational point of view, a CVT can also be defined as the minimum of the following energy function

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

where $\rho(\mathbf{x})$ is a density function, controlling the sizes of Voronoi cells, and we set $\rho(\mathbf{x}) = 1$ for our case. The optimization of the positions of seed points $X$ will make them evenly distributed on $\Omega$, which is usually accomplished by using the Lloyd's method [28] or the BFGS method [27].

However, we cannot directly extend the computation of CVTs on the point cloud since the underlying surface is usually unknown. We have to first extract the domain that RVCs can be defined. Fortunately, by constructing a set of local approximating planes of the raw data (Fig. 2(a)), Chen et al. [16] successfully generalized the CVT energy function to point clouds. These bounded planes make up a discrete surface of the input point cloud, which is commonly adopted by existing methods. Using the plane to represent points with widely differing normals, such as sharp features of the point cloud, causes high distortion error. Therefore, we choose quadratic surfaces to locally approximate the input points because of their better performance in feature preservation, as illustrated in Fig. 2, at the cost of a light increase in computation.

In particular, the quadratic surface $S_i$ at point $\mathbf{x}_i$ is locally computed as the least squares fitting of the k-nearest neighbors of

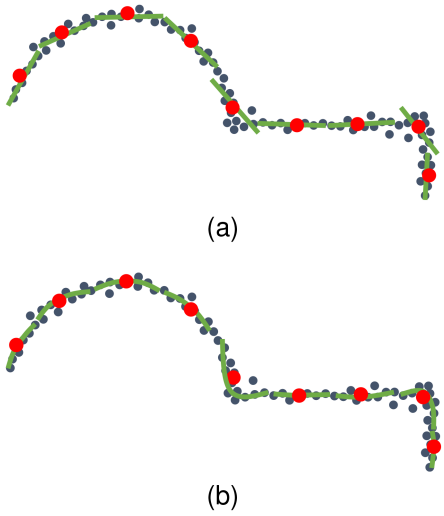


Fig. 2. Illustration for demonstrating the superiority of quadratic surfaces in preserving details and features over planes. The input points, resampling points, and approximating shapes are marked by dark grey points, red points, and green line segments/curves, respectively. (a) Approximating planes; (b) Approximating quadratic surfaces.

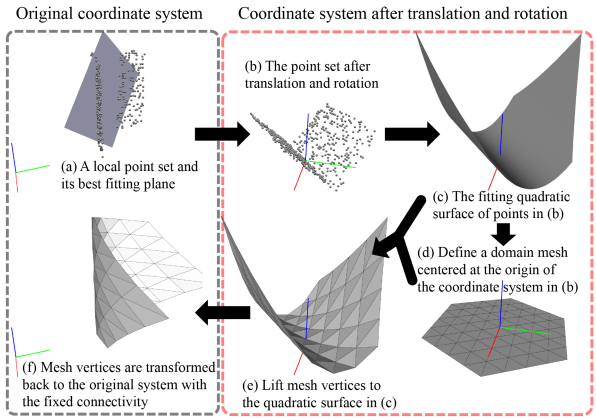


Fig. 3. Illustration for the construction of a local fitting quadratic surface.

$\mathbf{x}_i$ in the input point cloud. The details are given in Section III-B. Then the RVC of $\mathbf{x}_i$ is obtained by calculating the intersection between its 3D Voronoi cell $V_i$ and the corresponding surface $S_i$, as seen in Section III-C. With this configuration, the CVT energy function is then modified to

$$E(X) = \sum_{i=1}^{n} \int_{V_i \cap S_i} \|\mathbf{x} - \mathbf{x}_i\|^2 \, d\mathbf{x}. \tag{2}$$

The set $\{S_i\}_{i=1}^n$ is dynamically changed with the optimization of $\{\mathbf{x}_i\}_{i=1}^n$. We will adopt Lloyd's method with Anderson acceleration to obtain CVTs and uniform points on the given point cloud, which can be found in Section IV.

### B. Quadratic Surface Construction

In this section, we explain how to quickly construct an approximating quadratic surface at each seed point.

An implicit quadratic surface is formulated as

$$S(x, y, z) = \mathbf{f} \cdot \mathbf{\Lambda} = 0 \tag{3}$$

where $\mathbf{f} = (x^2, y^2, z^2, xy, yz, zx, x, y, z, 1)^T$ is the basis function and $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_{10})^T$ the parameter vector. For a seed $\mathbf{x}_i$, we find its k-nearest neighbors $P_i$ from the point cloud $P$ and suppose that $\mathbf{p}_{\mathbf{x}_i} \in P_i$ is the nearest one. The normal $\mathbf{n}_{\mathbf{p}_{\mathbf{x}_i}}$ of $\mathbf{p}_{\mathbf{x}_i}$ is estimated as the unit normal of the best fitting plane of $P_i$. The steps to construct the approximating surface of $\mathbf{x}_i$ are given as follows, accompanying with an illustration in Fig. 3.

*Step 1: transformation.* The point set $P_i$ is translated to the coordinate system originating at $\mathbf{p}_{\mathbf{x}_i}$ by left-multiplying a translation matrix

$$T = \begin{pmatrix} \mathbf{I}_{3\times3} & -\mathbf{p}_{\mathbf{x}_i} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix}$$

in homogeneous coordinates, where $\mathbf{I}_{3\times3}$ is an identity matrix. The translated points are rotated around $(a, b, c)^T = \mathbf{n}_{\mathbf{p}_{\mathbf{x}_i}} \times (0, 0, 1)^T$ with angle $\theta = arccos(\mathbf{n}_{\mathbf{p}_{\mathbf{x}_i}} \cdot (0, 0, 1)^T)$, equaling to left-multiplying a rotation matrix

$$R = \begin{pmatrix} \mathbf{R}_x^T & 0 \\ \mathbf{R}_y^T & 0 \\ \mathbf{R}_z^T & 0 \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix}$$

where $\mathbf{R}_x = (ac(1-cos\theta) + bsin\theta, ab(1-cos\theta) - csin\theta, a^2 + (1-a^2)cos\theta)^T$, $\mathbf{R}_y = (bc(1-cos\theta) - asin\theta, b^2 + (1-b^2)cos\theta, ab(1-cos\theta) + csin\theta)^T$ and $\mathbf{R}_z = (c^2 + (1-c^2)cos\theta, bc(1-cos\theta) + asin\theta, ac(1-cos\theta) - bsin\theta)^T$.

*Step 2: fitting.* A new point set $Q_i$ can be obtained after sequently applying the above two matrices to each point in $P_i$, indicated from (a) to (b) in Fig. 3, and (3) can be simplified to

$$z = \hat{S}(x, y) = \hat{\lambda_1} x^2 + \hat{\lambda_2} y^2 + \hat{\lambda_3} xy + \hat{\lambda_4} x + \hat{\lambda_5} y + \hat{\lambda_6}. \tag{4}$$

We calculate the parameters $(\hat{\lambda_1}, \ldots, \hat{\lambda_6})$ over the point set $Q_i$ in the least squares sense (Fig. 3(c)). We define the domain of $\hat{S}(x, y)$ to be a hexagon with radius $r$ centering at the origin, where $r$ is the average distance between $\mathbf{x}_i$ and its six nearest neighbors in $X$. Then, $\hat{S}(x, y)$ is completely constructed.

*Step 3: meshing.* We discretize the surface $\hat{S}(x, y)$ to a triangular mesh $\hat{M}$ for the subsequent RVC computation. In particular, we sample the vertices of $\hat{M}$ from the domain of $\hat{S}(x, y)$ using polar coordinates with gradually incremental radius and angles and connect them by the Delaunay triangulation method. These vertices are then lifted to $\hat{S}(x, y)$ with fixed connectivity, producing a bounded surface mesh, see Fig. 3(d) and (e).

*Step 4: inverse transformation.* The local approximating surface $S_i$ of $\mathbf{x}_i$ can be obtained through inverse transformation of $\hat{M}$ (Fig. 3(f)). By fixing the connectivity, each vertex of $\hat{M}$ is transformed back by left-multiplying

$$\begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{p}_{\mathbf{x}_i} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix} \cdot R^{-1},$$

where $R^{-1}$ is the inverse matrix of $R$.

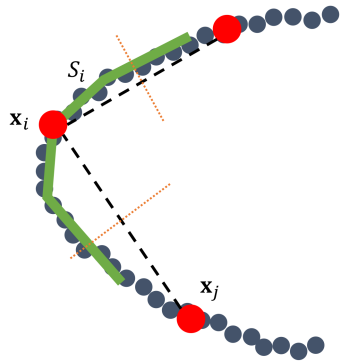


Fig. 4. Schematic for the RVC computation. Raw points, resampling points, approximate surface mesh, and bisectors between two consecutive resampling points are marked by grey points, red points, green polylines, and orange dotted lines, respectively.

These four steps provide us with the fast generation of local surface meshes, where the meshing step is easy to implement because of the transformation. By contrast, it is complicated if meshing occurs directly in the original space.

### C. RVC Computation

Once the approximating surface $S_i$ of point $\mathbf{x}_i$ is obtained, the RVC computation of $\mathbf{x}_i$ can be launched immediately. It requires the 3D Voronoi cell $V_i$ of $\mathbf{x}_i$. A common practice is to first build the Delaunay triangulation of the points $X$, and then compute the dual diagram of the triangulation as the Voronoi cells. Owing to the considerable time needed for the construction of Delaunay triangulation, Lévy and Bonneel [48] proposed a more efficient clipping algorithm based on a k-dimensional tree (KD-tree) of $X$. The algorithm is based on the observation that Voronoi cell $V_i$ is the intersection of half-spaces, bounded by the bisectors between $\mathbf{x}_i$ and its several nearest neighbors in $X$. We adopt this algorithm for our RVC computation. Specifically, the RVC of $\mathbf{x}_i$ is obtained by clipping the surface mesh $S_i$ by the bisectors of all point pairs $(\mathbf{x}_i, \mathbf{x}_j), j = 1, \ldots, n, j \neq i$. In practice, these bisectors are sorted in advance by their distances to $\mathbf{x}_i$. Starting from the nearest bisector, $S_i$ is gradually clipped into a smaller one, and it ends when the clipping generates no intersection. An illustration of the RVC computation is shown in Fig. 4.

## IV. Accelerated Lloyd's Method for Point Resampling

Our algorithm takes a point cloud $P$ as the input and generates uniform resampling points with the desired number $n$, which mainly consists of the initialization stage and optimization stage, proceeding as shown in Algorithm 1. The former allows us to generate favorable initial resampling points, as seen in Section IV-A. For the latter, we first present the details using Lloyd's method in Section IV-B and then provide an accelerated version in Section IV-C. Resampling points should be projected back to the underlying surface of the given point cloud once they have been updated, which is described in Section IV-D. Furthermore, we add a boundary restriction for the movement of resampling points, enabling our method to be available for unclosed data without any boundary detection, as seen in Section IV-E.

**Algorithm 1:** Accelerated Lloyd's Method for Resampling

**Input**: point cloud $P$, iteration number $K_{\max}$, and desired point number $n$.
**Output**: uniform resampling points $\{\mathbf{x}_i\}_{i=1}^n$.
1: initialize $\{\mathbf{x}_i\}_{i=1}^n$ (Section IV-A);
2: $k \leftarrow 1$;
3: **repeat**
4: construct surface for each $\mathbf{x}_i$ (Section III-B);
5: compute RVC for each $\mathbf{x}_i$ (Section III-C);
6: optimize $\{\mathbf{x}_i\}_{i=1}^n$ (Section IV-C);
7: project $\{\mathbf{x}_i\}_{i=1}^n$ back to the surfaces (Section IV-D);
8: $k \leftarrow k + 1$;
9: **until** $k > K_{\max}$

### A. Initialization

Generating uniform points is one of the main goals of our method. If the input points are unevenly distributed, our method can still be applicable, but it may cost more iterations. The efficiency of our approach in obtaining uniform results is highly dependent on initialization. This stage generates initial resampling points for subsequent optimization. A simple and fast initialization method is to randomly select $n$ points from the point cloud $P$, resulting in a distribution that is highly similar to the original points. To obtain a uniform result, the random initialization imposes a heavy burden on optimization when the input dataset is highly non-uniform, i.e., more iterations are required to optimize the resampling points.

To reduce the iteration number of the proposed algorithm on arbitrary input data, we employ an area-weighted initialization method. The steps are as follows.

*Step 1:* We randomly select $n$ points $\{\mathbf{x}_i\}_{i=1}^n$ from $P$, and compute the local surfaces $\{S_i\}_{i=1}^n$ and the RVCs $\{\Omega_i = V_i \cap S_i\}_{i=1}^n$.

*Step 2:* For each RVC $\Omega_i$, we compute its area $|\Omega_i|$, and sample $(|\Omega_i| \cdot n)/\sum_{j=1}^n |\Omega_j|$ points in a random fashion. A large RVC means sparse distribution of the original points, and the area-weighted sampling can generate more points here.

*Step 3:* These sampled points may deviate from the underlying surface of the given point cloud, and we update their respective approximating surfaces and project them back according to the operator described in Section IV-D. The projected points are our initial sampling points, still denoted by $\{\mathbf{x}_i\}_{i=1}^n$.

This area-weighted initialization is independent of the original distribution, and allows us to obtain uniform results with fewer iterations. We give a comparison shown in Fig. 5 that our algorithm runs on the random and the area-weighted initializations, respectively. The input point cloud (Fig. 5(a)) is unevenly distributed, and a random initialization (Fig. 5(b)) shows a similar distribution to the input, while an area-weighted initialization (Fig. 5(d)) indicates an equal possibility of generating initial resampling points everywhere. The resulting points (Fig. 5(c)) optimized from random initialization are still clustered heavily

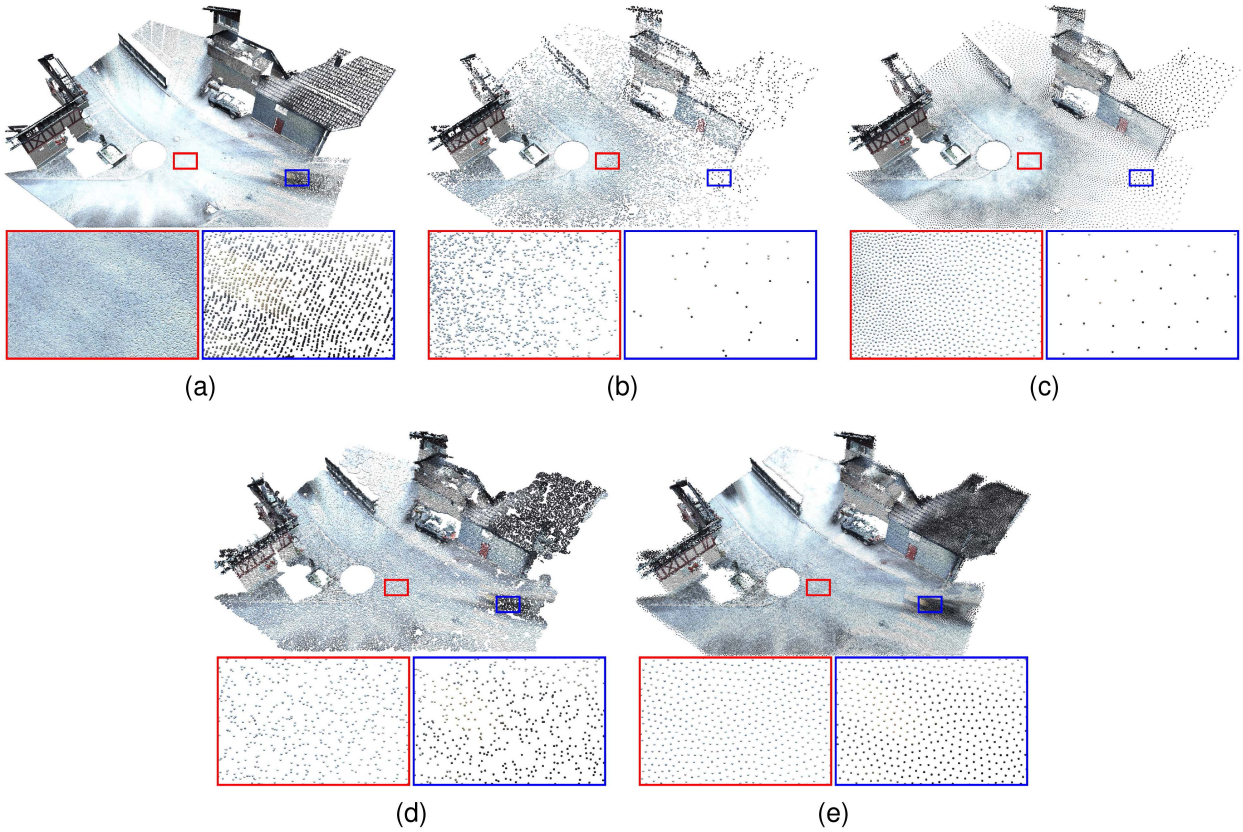


Fig. 5. Comparison of random initialization and area-weighted initialization. The latter can provide us better initial resampling points to decrease the burden of the subsequent optimization. (a) Input; (b) Random initialization; (c) Result from (b); (d) Area-weighted initialization; (e) Result from (d).

near the central region but sparsely on the borders. By contrast, the area-weighted initialization leads to a more uniform distribution after applying the same number of iterations (Fig. 5(e)).

### B. Optimization Using Lloyd's Method

After the initialization, the resampling points can be then optimized under the guidance of CVT generation. We employ Lloyd's method to achieve the goal, which iteratively moves resampling points to the corresponding RVC centroids. The movement repeats $K_{\max}$ times, where $K_{\max}$ is a preset maximum iteration number, and each resampling point is updated by

$$\mathbf{x}_i^{k+1} = \mathbf{c}_i^k = \frac{\int_{V_i \cap S_i} \mathbf{x} d\mathbf{x}}{\int_{V_i \cap S_i} d\mathbf{x}}, \tag{5}$$

where $k$ is the iteration index, $\mathbf{c}_i$ is the RVC centroid of $\mathbf{x}_i$.

After each movement of resampling points, we need to rebuild their approximating surfaces, meaning that (2) is defined over a dynamic domain. Hence, any local minima of the energy function implies nothing in our case, and the trend of the energy function value probably fluctuates dramatically, as shown in the top of Fig. 6(e), for example. An observation after our extensive experiments is that using (5) to optimize the resampling points is still capable of producing CVTs and uniform resampling points, an example is shown in Fig. 6.

Given that the uniformity of resampling points can no longer be measured by the energy function values, we define another intuitive metric below to quantify it:

$$\delta(X) = \sqrt{\frac{\sum_{i=1}^{n} \left(d_i/\bar{d} - 1\right)^2}{n}}, \tag{6}$$

where $d_i$ is the distance between $\mathbf{x}_i$ and its nearest neighbor in $X$, and $\bar{d} = \sum_{i=1}^{n} d_i/n$ is the average of the distance set $\{d_i\}_{i=1}^{n}$. $\delta(X)$ is the variance of regularized $\{d_i\}_{i=1}^{n}$, measuring how close from each $d_i$ to $\bar{d}$. A lower $\delta$ value means a more uniform point set.

In Fig. 6, we demonstrate the quality improvement during the optimization using Lloyd's method, as seen in the trend of uniformity metric at the bottom of Fig. 6(e). Starting from the initialization in Fig. 6(b), the distribution of resampling points becomes increasingly uniform during the optimization, which is indicated by the overall decline of the metric value. The final resampling result is shown in Fig. 6(c).

### C. Optimization Using Accelerated Lloyd's Method

Lloyd's method can be viewed as the fixed-point iteration that is with only linear convergence. Here, we incorporate Anderson acceleration to speed up the optimization. The domain of the proposed CVT energy function is dynamically changed during the optimization, resulting in the family of Newton or

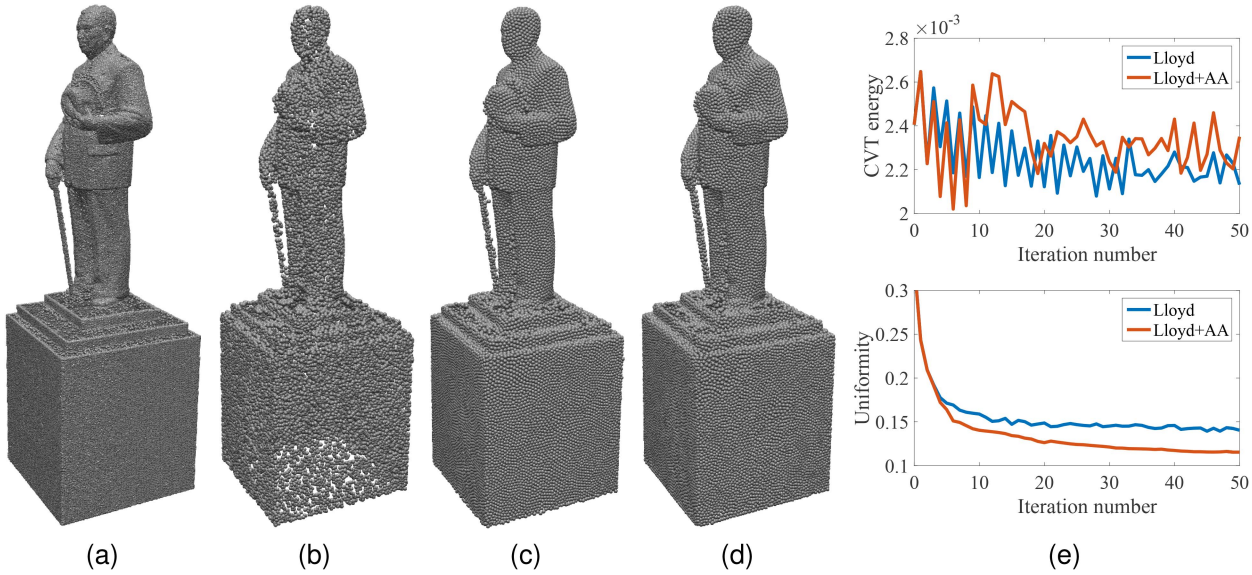


Fig. 6. Comparison between Lloyd's method and the proposed Lloyd+AA method. (a) Input point cloud; (b) Initialization; (c) Result from Lloyd's method; (d) Result using the proposed Lloyd+AA method; (e) Energy function graph (top) and uniformity metric graph (bottom) for the two optimization methods, respectively.

quasi-Newton methods that require high continuity of energy function being not applicable in our case. Given that Anderson acceleration has been utilized to speed up the searching for optimal solutions to various problems (e.g. [45], [46], [47]), and Peng et al. [47] found that it works well for the minimization of the CVT energy function defined over a continuous domain, it can still be extended to our case in practice.

The $k$-th solution $\mathbf{Y}^k = (\mathbf{x}_1^{kT}, \ldots, \mathbf{x}_n^{kT})^T$ can be updated from the previous solution $\mathbf{Y}^{k-1}$ in fixed-point iteration:

$$\mathbf{Y}^k = G(\mathbf{Y}^{k-1}), \tag{7}$$

where $G$ is a mapping function, and the *residual* is

$$\mathbf{D}^k = G(\mathbf{Y}^k) - \mathbf{Y}^k. \tag{8}$$

Anderson acceleration combines current iterate $\mathbf{Y}^k$ and the previous at most $m$ iterates $\mathbf{Y}^{k-1}, \ldots, \mathbf{Y}^{k-m}$ to derive a new iterate $\mathbf{Y}_{AA}^{k+1}$ that decreases the residual norm as much as possible, i.e.,

$$\mathbf{Y}_{AA}^{k+1} = \sum_{j=0}^{m} \alpha_j^* G(\mathbf{Y}^{k-j}), \tag{9}$$

where $(\alpha_0^*, \ldots, \alpha_m^*)$ is obtained by solving a linear least-squares problem:

$$(\alpha_0^*, \ldots, \alpha_m^*) = \underset{(\alpha_0,\ldots,\alpha_m)}{\arg\min} \left\| \sum_{j=0}^{m} \alpha_j \mathbf{D}^{k-j} \right\|^2, \text{s.t.} \sum_{j=0}^{m} \alpha_j = 1. \tag{10}$$

Walker and Ni [45] provides more details about the Anderson acceleration technique.

We follow Peng et al. [47] to set the mapping $G$ as

$$G\left(\mathbf{Y}^k\right) = \left(\mathbf{c}_1^{kT}, \ldots, \mathbf{c}_n^{kT}\right)^T \tag{11}$$

where $\mathbf{c}_i^k$ is the RVC centroid of $\mathbf{x}_i^k$ in the $k$-th iteration. Anderson acceleration needs to solve a $(m+1) \times (m+1)$ linear system for $(\alpha_0^*, \ldots, \alpha_m^*)$ in each iteration, involving more computation compared with Lloyd's method. However, $m$ is ordinarily set to a small value, which is $m = 4$ in this paper, the efficiency loss can be roughly ignored, as shown in Section V-E.

In addition, Anderson acceleration has been proven to become unstable and may lead to divergence if the iterates are far away from the solution [45], [49]. To improve the stability, we add a uniformity check:

$$\mathbf{Y}^{k+1} = \begin{cases} \mathbf{Y}_{AA}^{k+1}, \ \delta\left(\mathbf{Y}_{AA}^{k+1}\right) \geq \delta\left(G\left(\mathbf{Y}^k\right)\right) \\ G(\mathbf{Y}^k), \ \delta\left(\mathbf{Y}_{AA}^{k+1}\right) < \delta\left(G\left(\mathbf{Y}^k\right)\right) \end{cases}, \tag{12}$$

which means we always choose the one with better uniformity from the Lloyd's and the Anderson acceleration iterates. We denote the hybrid optimization as Lloyd+AA.

A comparison of Lloyd's method and Lloyd+AA is given in Fig. 6. From the same initialization and with the same iteration number, the hybrid optimization method generates a resampling result with higher uniformity, as shown in the curves in Fig. 6(e). In other words, Lloyd+AA requires fewer iterations to obtain the resampling points with a target uniformity than Lloyd's method. Without specific notion, we adopt Lloyd+AA to run the optimization.

### D. Projection

After the optimization in each iteration, the resampling points may deviate from the original point cloud because each centroid is probably not on the corresponding RVC, especially in the curved regions. Despite the deviation not accumulating during the optimization, it affects the result of the KNN search and we still pull the resampling points back to the point cloud.

A straightforward operator is to push each resampling point back to its nearest neighbor in the input point cloud, potentially resulting in the resampling points being a subset of the

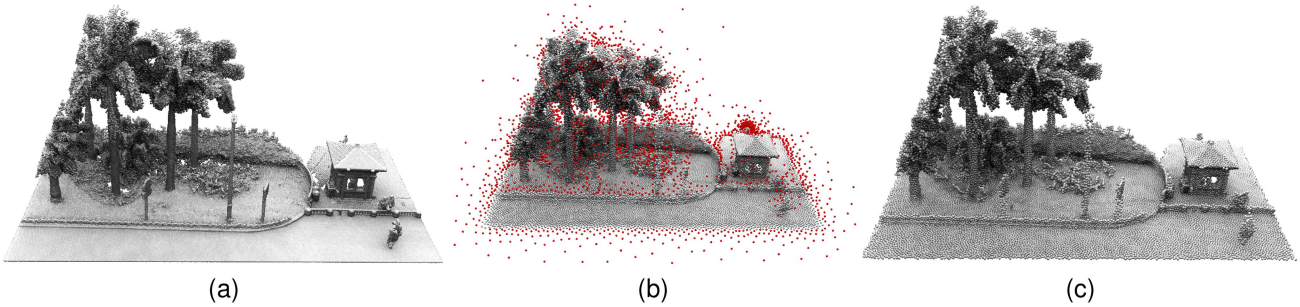


Fig. 7. Results without and with boundary restriction. The resampling points whose distance to the input data exceeds the threshold are marked in red. (a) Input; (b) Result without boundary restriction; (c) Result with boundary restriction.

original data and difficulty of uniformity improvement. Given that we already have local surface meshes to represent the point cloud, projecting resampling points to these meshes is reasonable. More specifically, for each resampling point, we find its nearest triangle in the local surface mesh and project it onto this triangle. The projection only relies on these surface meshes and guarantees the resampling points close to the original points. The projected points are then used to compute new approximating surfaces and RVCs in the next iteration.

### E. Boundary Restriction

The existing resampling algorithms can be applied to unclosed point clouds only when the boundary detection has been executed in advance (e.g., Chen et al. [16]). The main reason is that those resampling points initially located at boundaries will gradually float away from the input data. A natural solution is to find out all boundaries of the input and then fix resampling points on the border during the optimization. However, challenges remain in boundary detection for complex point clouds, narrowing the application range of these algorithms.

We observe that the movement of resampling points only relies on the local surface meshes, while the distance from each resampling point to the given point cloud has not been considered. We address this floating-issue by adding a distance restriction on each resampling point. If the distance between the resampling point and its nearest neighbor in $P$ is greater than a given threshold, the resampling point will stand still at the current iteration.

The distance threshold controls the magnitude of points drifting. A smaller value will make the optimization of resampling points stuck, while a larger one cannot eliminate the drifting although it fills gaps in the input data. Owing to the scale difference of various inputs, setting the threshold at a fixed value is difficult. Instead, we determine it using the intrinsic information of the input data. Specifically, the threshold is scaled with an average value of the distances between the original points and their nearest neighbors in $P$. With massive tests, we find that $0.5\sim1.5$ times this average distance is an acceptable tradeoff.

As shown in Fig. 7, we demonstrate the benefit of boundary restriction that significantly reduces the points drifting. Complex boundary information exists in the input point cloud (Fig. 7(a)). The result obtained by the proposed algorithm without boundary restriction is shown in Fig. 7(b), where the drifted resampling points are marked in red. By adding this distance constraint, we can produce uniform resampling points without the drifting phenomenon, as seen in Fig. 7(c).

For resampling point clouds with holes, the boundary restriction may be conflict with hole filling. If the hole size is larger than the distance threshold of the boundary restriction, the hole will not be filled completely, but it can be shrunk after applying our method. Therefore, a promising solution is to run the proposed algorithm few times, where the output of the algorithm is fed as the input in the next time.

## V. Results

In this section, we evaluate our method of generating uniform and feature-preserving resampling points and explore its applications. The proposed algorithm is realized in C++, and the neighboring relationship between points is recorded through a KD-tree implemented by the NanoFLANN library [50], [51], which also provides efficient KNN query functions. All experiments are executed on a PC with an Intel Xeon E3-1226 3.30 GHz CPU and 12 GB RAM.

For better visualization, the input data are rendered with their original colors if the color attribute is available, and the color of each resampling point is picked from its nearest point in the input data. The point clouds in Figs. 11–13 are scale-normalized to have a unit bounding box diagonal for point distances comparison. For numerical abbreviations, M is for million and K is for thousand.

### A. Uniform Resampling

All the examples in this paper demonstrate that our method performs outstandingly in generating high-quality, feature-preserving uniform resampling results. Fig. 1 shows our resampling result of a street dataset. The input point cloud (Fig. 1(a)) contains 5 M points with uneven density, and we resample 100 K points (2%) to represent the original environment. The output is given in Fig. 1(b), where we can see all geometric information is kept, and some small gaps are fulfilled (as seen in the roof and the ground on the left). The uniformly distributed resampling points also preserve the fine details and features of the input, which can be observed from the windowsill enlarged in the red box.

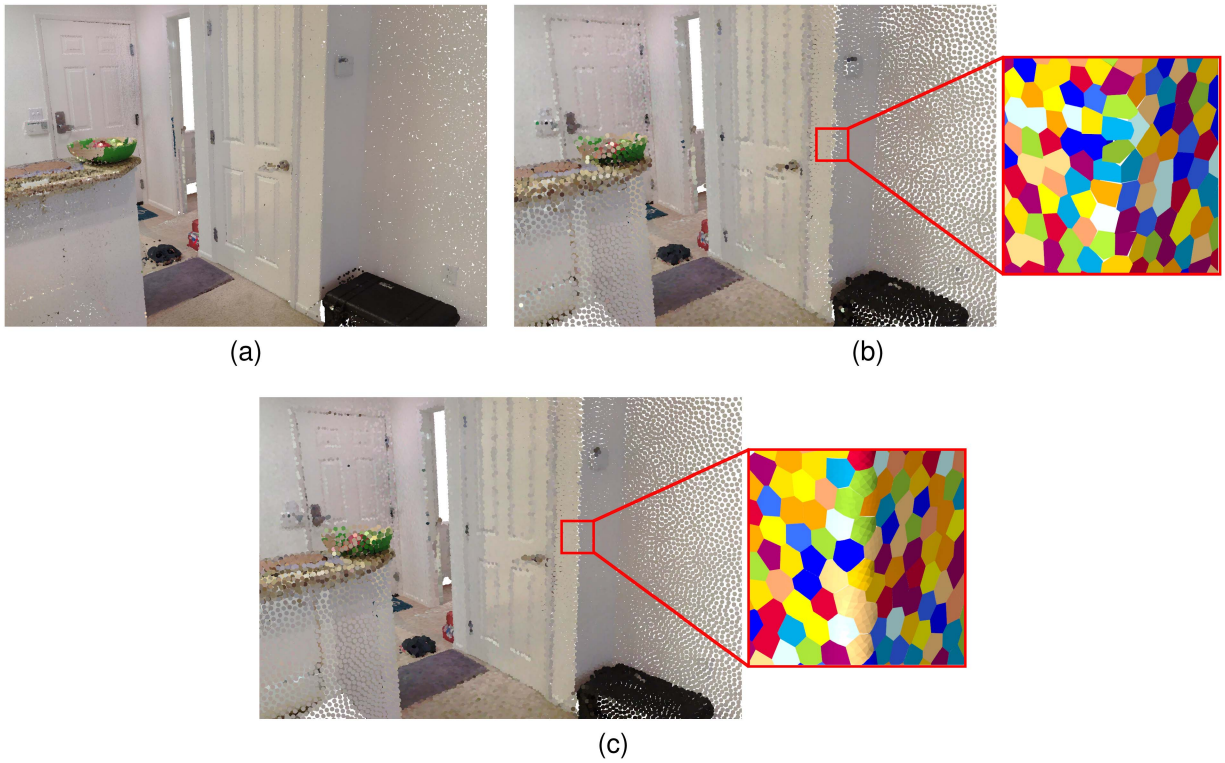


Fig. 8. Comparison with Chen et al. [16], another CVT-based resampling method whose CVT energy function is defined on a set of local planes. The boundary restriction has been added to Chen et al. [16]. (a) Input; (b) Result using method proposed by Chen et al. [16], $\delta = 0.15$; (c) Our result, $\delta = \mathbf{0.14}$. We possess better performance in feature preserving because of the more accurate representation of the input point cloud, see the zoomed-in details.

### B. Comparisons

Chen et al. [16] proposed another CVT-based point cloud resampling method in which their CVT energy function is defined on a set of local planes. Fig. 8 shows a comparison between their method and ours, and for fairness, we have added the boundary restriction to the implementation of [16]. The input is an indoor point cloud and the part shown in Fig. 8(a) is one of its corners. The resampling results of Chen et al. [16] and our method are shown in Fig. 8(b) and (c), respectively. The zoomed-in RVCs show that the quadratic surfaces represent the original points more accurately than linear planes, especially near the regions where the normals of original points change dramatically. The better approximation of the input points contributes more to decreasing the distortion error and increasing the uniformity of resampling points, where the resulting $\delta$ value of Chen et al. [16] is 0.15, while ours is 0.14.

We also compare our method with WLOP [13], Li and Sun [15], Chen et al. [16] and Lv et al. [18] to further evaluate our method. These methods are applied to different types of point clouds, respectively, the results are shown in Figs. 9 and 10. In Fig. 9, the input points exhibit a distribution of white noise. We resample 10 K points from the given 30 K points by using different methods. While the results show that all these methods succeed to increase the level of uniformity of output points, our method outperforms the others as evidenced by visualization and $\delta$ value of each result. In Fig. 10, the input dataset is highly non-uniform, with a higher point density closer to the scanning device, as shown in Fig. 10(a), and we resample 100 K points from the given 881 K points. Although WLOP adds weight to each input point to reduce the influence of the original distribution on the result, generating resampling results with globally high uniformity remains difficult with WLOP, as seen in Fig. 10(b). Similarly, the result of Li and Sun [15] (Fig. 10(c)) shows dense points in the central part and sparse distribution on the left and right sides, since their optimization of resampling points depends heavily on the nodes of the KNN graph, which are usually not uniformly distributed. By contrast, Chen et al. [16], Lv et al. [18] and our method produce much more uniform results even from the global perspective, as seen in the two zoomed-in details and $\delta$ values. Differ from Lv et al. [18] who adopt collapse and split operations, such non-uniform distribution is quickly eliminated by the area-weighted initialization in our method.

More comprehensive quality comparisons are given in Table I, where $\sigma$ and $\epsilon$ are another two metrics of point uniformity. $\sigma$ is defined as

$$\sigma(X) = \frac{1}{n}\sum_{i=1}^{n} \left| d_i - \bar{d} \right|,$$

it means the average Manhattan distance between the vectors $(d_1, \ldots, d_i, \ldots, d_n)^T$ and $(\bar{d}, \ldots, \bar{d}, \ldots, \bar{d})^T$, where $d_i$ and $\bar{d}$ are the same with (6). A lower $\sigma$ value corresponds to a more

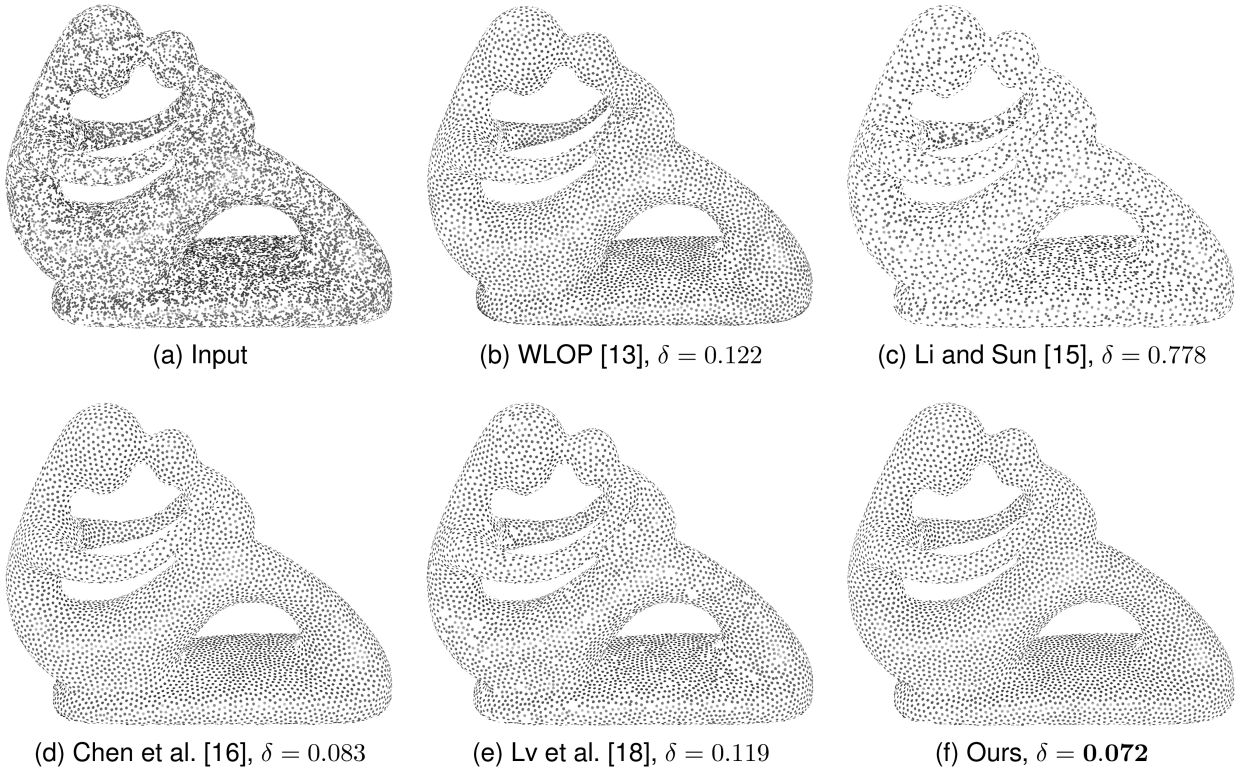


Fig. 9. Comparison between different resampling methods on point cloud of an object. From the quality statistics, it can be concluded that our method outperforms the others in obtaining points with high uniformity.

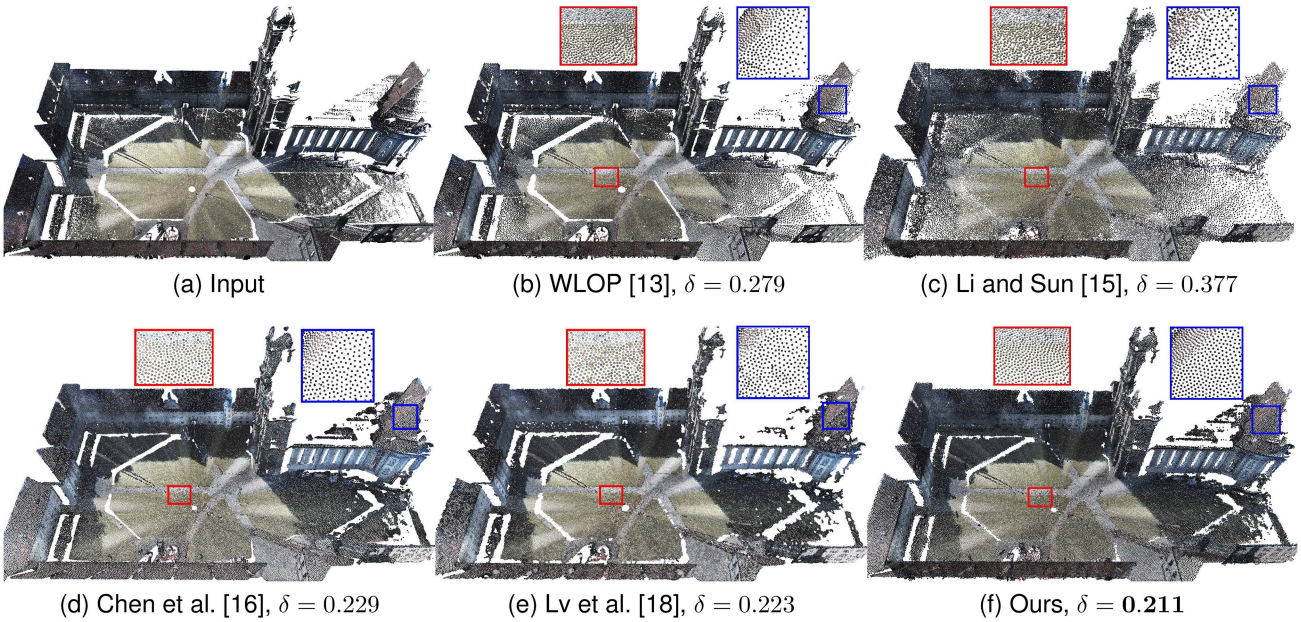


Fig. 10. Comparison between different resampling methods on LiDAR point cloud. Both the close-up view of the two regions highlighted in the red and blue boxes at the top of each result and the quality statistics indicate that our method achieves the best performance in the distribution uniformity.

uniform distribution. $\epsilon$ measures cosine value of the angle between the above two vectors and can be calculated by

$$\epsilon(X) = \frac{\sum_{i=1}^{n} \left(d_i \cdot \bar{d}\right)}{\sqrt{\sum_{i=1}^{n} d_i^2} \sqrt{\sum_{i=1}^{n} \bar{d}^2}},$$

it is in the range $[-1, 1]$, and the higher the better. Table I shows that our results are the best for most tests, supporting the conclusion that our method shows the best performance of resulting point uniformity compared with WLOP [13], Li and Sun [15], Chen et al. [16] and Lv et al. [18]. Although the method of Lv

TABLE I
QUANTITATIVE COMPARISONS WITH WLOP [13], LI AND SUN [15], CHEN ET AL. [16] AND LV ET AL. [18]

| Figs. | $\delta$ | | | | | $\sigma$ | | | | | $\epsilon$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [13] | [15] | [16] | [18] | Ours | [13] | [15] | [16] | [18] | Ours | [13] | [15] | [16] | [18] | Ours |
| 1 | 1.475 | 1.364 | 0.433 | 0.366 | **0.352** | 0.071 | 0.114 | 0.067 | 0.058 | **0.052** | 0.561 | 0.591 | 0.918 | 0.935 | **0.943** |
| 5 | 1.099 | 0.882 | 0.415 | **0.333** | 0.335 | 0.022 | 0.039 | 0.024 | **0.015** | 0.016 | 0.673 | 0.749 | 0.917 | **0.951** | 0.948 |
| 6 | 0.504 | 0.225 | 0.143 | 0.121 | **0.115** | 0.0052 | 0.0018 | 0.002 | **0.0016** | 0.0019 | 0.893 | 0.976 | 0.989 | 0.99 | **0.993** |
| 7 | 0.533 | 0.291 | 0.285 | 0.284 | **0.282** | 0.028 | **0.017** | 0.022 | 0.019 | 0.024 | 0.882 | 0.960 | 0.959 | 0.9623 | **0.9627** |
| 8 | 0.189 | 0.194 | 0.203 | **0.139** | 0.14 | 0.0027 | 0.0026 | 0.0024 | **0.0015** | 0.0017 | 0.983 | 0.982 | 0.979 | **0.995** | 0.991 |
| 9 | 0.122 | 0.778 | 0.083 | 0.119 | **0.072** | 0.197 | 0.951 | 0.148 | 0.204 | **0.131** | 0.992 | 0.788 | 0.996 | 0.992 | **0.997** |
| 10 | 0.279 | 0.377 | 0.229 | 0.223 | **0.211** | 0.062 | 0.084 | 0.059 | 0.058 | **0.057** | 0.963 | 0.935 | 0.971 | 0.967 | **0.972** |

*Note: The best results in the table are shown in bold font. $\delta$, $\sigma$ and $\epsilon$ are three different uniformity metrics, the lower $\delta$ and $\sigma$ the better, and the higher $\epsilon$ the better.*

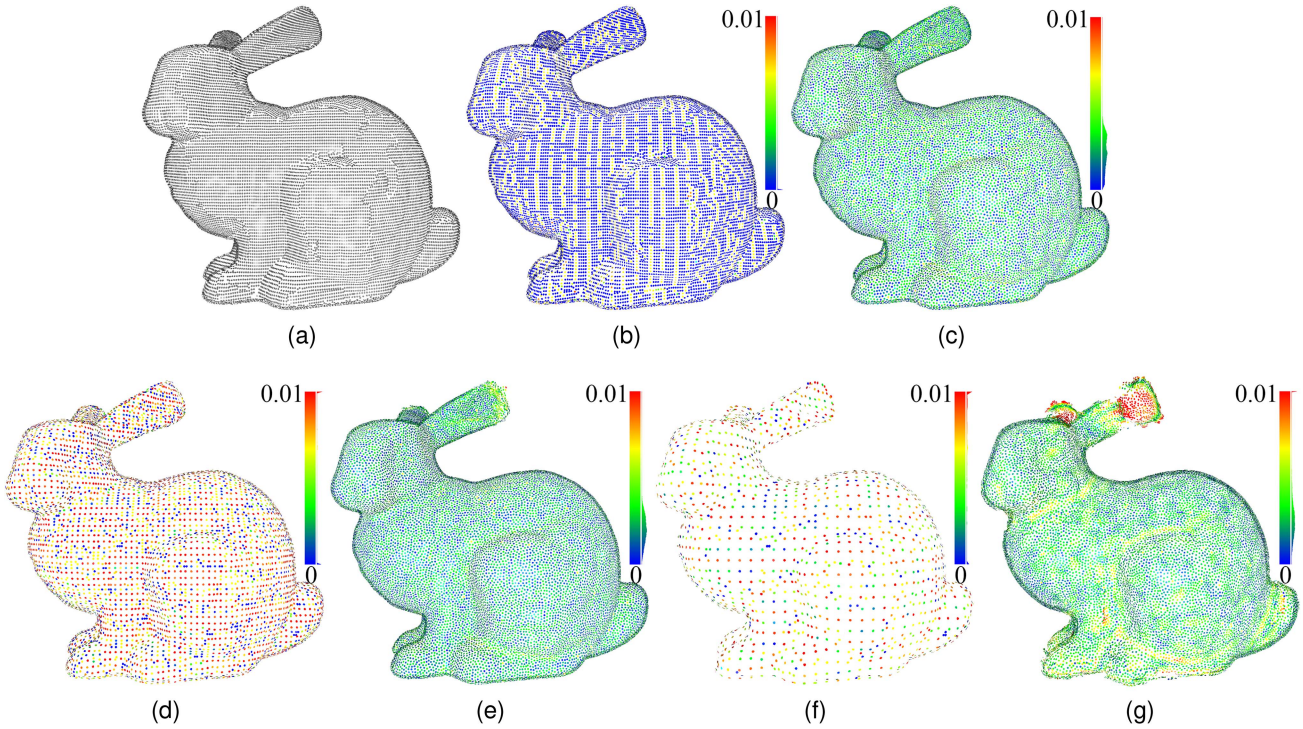


Fig. 11. Results that our method recovers point clouds from compressed point clouds. (a) original point cloud, 34 K points; (b), (d), (f) Octree encoded point clouds of (a), where compression rates are 11.7% (b), 67.6% (d), and 88.2% (f), respectively; (c), (e), (g) Recovered point clouds from (b), (d), (f), respectively, by using our method. The color of each point in (b)–(g) indicates its distance to the nearest point of cloud (a).

et al. [18] produces competitive results, it is with much higher time cost, as discussed in V-E.

### C. Upsampling

Point cloud upsampling predicts and generates a denser point cloud with fine details from the input points with low-density, which is also required when recovering a compressed point cloud. Benefiting from the construction of approximating surfaces of the given point cloud, our method is competent for the upsampling task. A minor modification in the initialization of the proposed algorithm can be made to generate more points than the input. Specifically, most points of the input data are selected and the disk radius $r$ (Step 2 in Section III-B) is enlarged to obtain the RVCs for full coverage of the point cloud surface. From here, the new points with the desired number can be then randomly sampled.

Recovering from compressed point clouds is a challenging task since they usually suffer from the loss of geometry information, including point vanishing and displacement. We provide recovered results from octree encoded point clouds with different compression rates by using our method in Fig. 11. Indicated by the distances to the original point cloud, we can see that our method is able to generate point clouds faithfully with the original one when the compression rate is moderate, see Fig. 11(c) and (e). However, our method cannot recover the details well enough if the input is heavily compressed, see the ear part of Fig. 11(g).

The point cloud upsampling also attracts much attention in the community of deep learning (e.g., [6], [7]). We compare with the PU-Dense method [6] in Fig. 12. Both methods succeed in accomplishing the upsampling task. Implied by the point colors, the result of the PU-Dense method seems globally closer to the original point cloud than our result. It is worth pointing

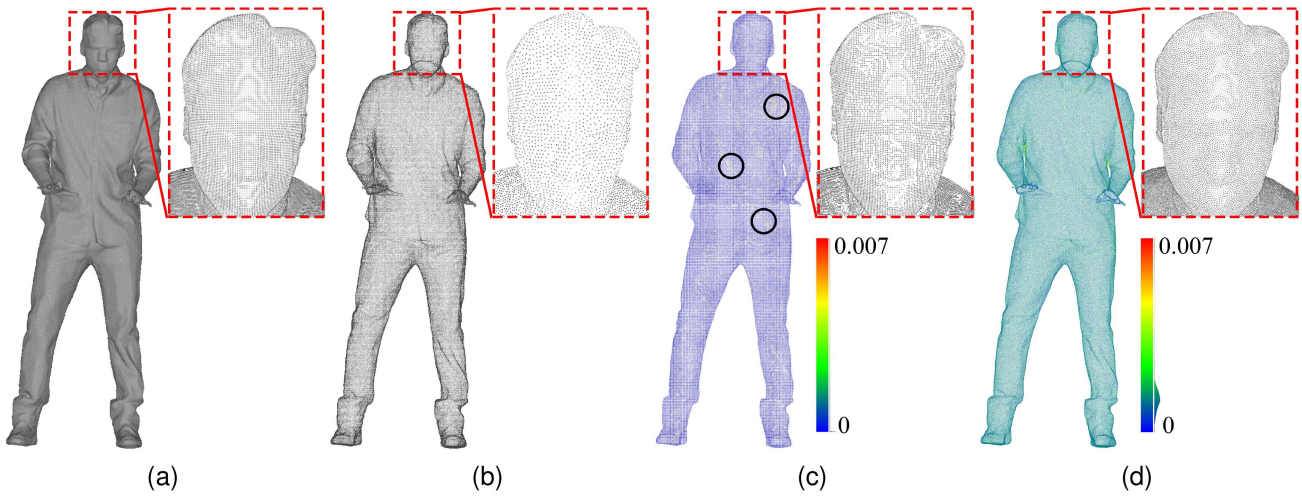


Fig. 12. Point cloud upsampling comparison with the PU-Dense [6]. (a) A point cloud from 8iVFBv2 dataset [52], 784 K points; (b) Down-sampled cloud from (a), 120 K points; (c) The upsampling result of the PU-Dense method from (b), 480 K points, $\delta = 0.099$; (d) Our upsampling result from (b), 480 K points, $\delta = \mathbf{0.087}$. The color of each point in (c) and (d) indicates its distance to the nearest point of cloud (a).

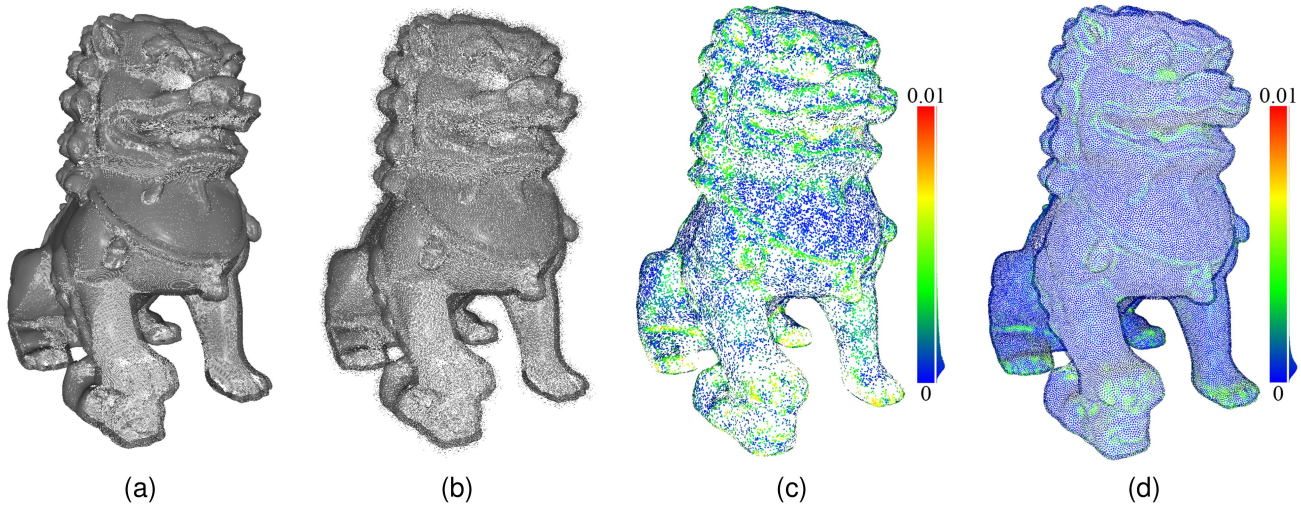


Fig. 13. Point cloud denoising comparison with the PointCleanNet [36]. (a) Original point cloud, 655 K points; (b) Noisy point cloud of (a); (c) The result of PointCleanNet, 100 K points, $\delta = 0.561$; (d) Our result, 100 K points, $\delta = \mathbf{0.08}$. The color of each point in (c) and (d) indicates its distance to the nearest point of cloud (a). All the noises are removed successfully and fine details are recovered faithfully for both results, and our result is more uniform and closer to the original point cloud than the result of PointCleanNet.

out that there is a part of points whose distance to the nearest point of the original cloud is large; see the points in black circles in Fig. 12(c). In contrast, our method focuses more on the uniformity of resultant points than the PU-Dense method (see the comparison of zoomed-in details and $\delta$ values), and each point will be moved in our method, resulting in non-zero but small distances between our result and the original point cloud.

### D. Denoising

The noise depression is one of the intrinsic attributes of our method. This is because the local surface construction filters out the noises and each resampling point is projected onto the corresponding local surface during the optimization, potentially smoothing the resampling points. We provide a comparison with the PointCleanNet [36], a deep learning-based method for point cloud denoising, in Fig. 13. 25% of the input points (Fig. 13(a)) are randomly perturbed within a sphere with a radius of 0.01 to generate the noisy data (Fig. 13(b)) (note that the point cloud is scale-normalized to have unit bounding box diagonal). Despite all the noises are removed successfully and fine details are recovered faithfully for both results, our result is more uniform and closer to the original point cloud than the result of PointCleanNet. In addition, our denoising result is affected by the number of point neighbors in the local surface construction. If the number is small, those resampling points surrounded by too many noises may not get free from them, whereas a large number will result in the accuracy loss of data representation. We

TABLE II
RUNNING TIME COMPARISONS WITH WLOP [13], LI AND SUN [15] AND LV ET AL. [18] (SECOND)

| Figs. | #P.I. | #P.O. | $K_{max}$ | [13] | [15] | [18] | Ours |
|---|---|---|---|---|---|---|---|
| 1 | 5M | 100K | 20 | 338.1 | 203.5 | 657.4 | **45.6** |
| 5 | 5M | 100K | 10 | 240.3 | 102.4 | 794.5 | **23.8** |
| 6 | 237K | 20K | 50 | 158.9 | 32.2 | 82.3 | **21.4** |
| 7 | 536K | 50K | 20 | 66.1 | 27.6 | 361.7 | **9.9** |
| 8 | 1.7M | 100K | 20 | 173.6 | 80.9 | 1690.2 | **42.3** |
| 9 | 30K | 10K | 20 | 4.1 | **0.5** | 35.9 | 2.6 |
| 10 | 881K | 100K | 20 | 83.2 | 56.3 | 622.5 | **41.1** |
| 11(e) | 11K | 34K | 20 | N/A | N/A | N/A | 5.7 |
| 12 | 120K | 480K | 10 | N/A | N/A | N/A | 132.8 |
| 13 | 655K | 100K | 20 | 117.6 | 63.4 | 1829.2 | **51.5** |

*Note: #P.I. denotes the number of input points, #P.O. denotes the number of output points, and $K_{max}$ is the maximum iteration number for [13], [15] and our method.*

recommend users determine the number according to the scale of noises, and it is suggested to be $50\sim300$ for most noisy cases. In Fig. 13, the number is set to 150.

### E. Timing

The running time of most examples in this paper is given in Table II, the proposed algorithm usually takes a few seconds to generate a satisfactory result. The efficiency of the algorithm mainly depends on the number of iterations and the number of resampling points, while the scale of the original points and the construction of the KD-tree contribute less. In each iteration, the computation of RVCs and centroids occupies a major part of the processing time, which is not linearly proportional to the number of resampling points.

The running time in the table is obtained by executing Lloyd+AA. For Fig. 6, the running time of Lloyd+AA is 21.4 s, while it is 20.2 s for Lloyd's method. This result clearly shows the minor difference in efficiency after incorporating Anderson acceleration, which is 24 ms per iteration for this example. Although a slight loss of efficiency occurs, we have a higher possibility to obtain better results with fewer iterations.

We also incorporate the running times for WLOP [13], Li and Sun [15] and Lv et al. [18] in Table II, showing efficiency advantage of our method. The former two are variants of LOP method [12]. The iterative update of each sample point involves original points and sample points in its optimization process, typically consuming more time than our method. In the case of Lv et al. [18], despite its ability to generate competitive results, there are extra operations, including point collapse, split and mappings, during the optimization, resulting in much higher time cost.

## VI. CONCLUSION

We propose a CVT-based point cloud resampling method in this paper, the conventional CVT energy function is extended to point clouds by introducing a set of local approximating quadratic surfaces. Therefore, the resampling points can be optimized for generating CVTs on the given point cloud, which is achieved efficiently by Lloyd's method with Anderson acceleration. Despite the domain of the proposed CVT energy function being dynamically changed during the optimization, the proposed resampling method still produces satisfactory results. We also add a boundary restriction on the movement of resampling points, enabling the proposed method to be available for arbitrary 3D point clouds without any boundary preprocessing. Our method outperforms the existing methods in generating uniform and feature-preserving resampling results.

However, we cannot output satisfactory results with a small number of resampling points, especially when the input data are complicated, because the local surfaces will poorly approximate the given points. Exploring more local representations is one of our future research directions. By contrast, due to the movement of resampling points, some small details of the input data are probably coarsened. Decreasing the distance threshold of boundary restriction may alleviate it, but it will result in the difficulty of smoothing resampling points. Introducing a local threshold for each resampling point is a promising solution. Finally, most of the steps in the proposed algorithm can be parallelized, and we intend to develop an accelerated version on GPUs.

## REFERENCES

[1] B. Schwarz, "LIDAR: Mapping the world in 3D," *Nature Photon.*, vol. 4, no. 7, pp. 429–430, 2010.
[2] D. Bonatto, S. Rogge, A. Schenkel, R. Ercek, and G. Lafruit, "Explorations for real-time point cloud rendering of natural scenes in virtual reality," in *Proc. IEEE Int. Conf. 3D Imag.*, 2016, pp. 1–7.
[3] W.-B. Yang, M.-B. Chen, and Y.-N. Yen, "An application of digital point cloud to historic architecture in digital archives," *Adv. Eng. Softw.*, vol. 42, no. 9, pp. 690–699, 2011.
[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.
[5] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.
[6] A. Akhtar, Z. Li, G. V. d. Auwera, L. Li, and J. Chen, "PU-Dense: Sparse tensor-based point cloud geometry upsampling," *IEEE Trans. Image Process.*, vol. 31, pp. 4133–4148, 2022.
[7] H. Chen, B. Du, S. Luo, and W. Hu, "Deep point set resampling via gradient fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2913–2930, Mar. 2023.
[8] H. Song and H.-Y. Feng, "A global clustering approach to point cloud simplification with a specified data reduction ratio," *Comput.-Aided Des.*, vol. 40, no. 3, pp. 281–292, 2008.
[9] B.-Q. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using k-means clustering," *Comput.-Aided Des.*, vol. 43, no. 8, pp. 910–922, 2011.
[10] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kovačević, "Fast resampling of three-dimensional point clouds via graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 666–681, Feb. 2018.
[11] Q. Deng, S. Zhang, and Z. Ding, "Point cloud resampling via hypergraph signal processing," *IEEE Signal Process. Lett.*, vol. 28, pp. 2117–2121, 2021.
[12] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 22–es, 2007.
[13] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–7, Dec. 2009.
[14] X. Cheng, M. Zeng, J. Lin, Z. Wu, and X. Liu, "Efficient $L_0$ resampling of point sets," *Comput. Aided Geometric Des.*, vol. 75, 2019, Art. no. 101790.
[15] M. Li and C. Sun, "Refinement of LiDAR point clouds using a super voxel based approach," *ISPRS J. Photogrammetry Remote Sens.*, vol. 143, pp. 213–221, 2018.
[16] Z. Chen, T. Zhang, J. Cao, Y. J. Zhang, and C. Wang, "Point cloud resampling using centroidal Voronoi tessellation methods," *Comput.-Aided Des.*, vol. 102, pp. 12–21, 2018.

[17] K. Han, K. Jung, J. Yoon, and M. Lee, "Point cloud resampling by simulating electric charges on metallic surfaces," *Sensors*, vol. 21, no. 22, 2021, Art. no. 7768.
[18] C. Lv, W. Lin, and B. Zhao, "Intrinsic and isotropic resampling for 3D point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3274–3291, Mar. 2023.
[19] T. Schlömer, D. K. Heck, and O. Deussen, "Farthest-point optimized point sets with maximized minimum distance," in *Proc. ACM SIGGRAPH Symp. High Perform. Graph.*, 2011, pp. 135–142.
[20] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational blue noise sampling," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 10, pp. 1784–1796, Oct. 2012.
[21] F. d. Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun, "Blue noise through optimal transport," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 1–11, Nov. 2012.
[22] D.-M. Yan and P. Wonka, "Gap processing for adaptive maximal poisson-disk sampling," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 1–15, Oct. 2013.
[23] D.-M. Yan, J.-W. Guo, B. Wang, X.-P. Zhang, and P. Wonka, "A survey of blue-noise sampling and its applications," *J. Comput. Sci. Technol.*, vol. 30, pp. 439–452, 2015.
[24] S. Zhang et al., "Capacity constrained blue-noise sampling on surfaces," *Comput. Graph.*, vol. 55, pp. 44–54, 2016.
[25] A. G. M. Ahmed et al., "A simple push-pull algorithm for blue-noise sampling," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 12, pp. 2496–2508, Dec. 2017.
[26] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637–676, 1999.
[27] Y. Liu et al., "On centroidal Voronoi tessellation-energy smoothness and fast computation," *ACM Trans. Graph.*, vol. 28, no. 4, Sep. 2009, Art. no. 101.
[28] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[29] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *J. ACM*, vol. 12, no. 4, pp. 547–560, Oct. 1965.
[30] X.-F. Han et al., "A review of algorithms for filtering the 3D point cloud," *Signal Process.: Image Commun.*, vol. 57, pp. 103–112, 2017.
[31] B. Liao, C. Xiao, L. Jin, and H. Fu, "Efficient feature-preserving local projection operator for geometry reconstruction," *Comput.-Aided Des.*, vol. 45, no. 5, pp. 861–874, 2013.
[32] H. Huang et al., "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, 2013.
[33] C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
[34] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "PointproNets: Consolidation of point clouds with convolutional neural networks," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 87–99, 2018.
[35] P. Hermosilla, T. Ritschel, P.-P. Vázquez, A. Vinacua, and T. Ropinski, "Monte Carlo convolution for learning on non-uniformly sampled point clouds," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–12, Dec. 2018.
[36] M.-J. Rakotosaona, V. L. Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "PointCleanNet: Learning to denoise and remove outliers from dense point clouds," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 185–203, 2020.
[37] G. Metzer, R. Hanocka, R. Giryes, and D. Cohen-Or, "Self-sampling for neural point cloud consolidation," *ACM Trans. Graph.*, vol. 40, no. 5, pp. 1–14, Sep. 2021.
[38] W. Feng, J. Li, H. Cai, X. Luo, and J. Zhang, "Neural points: Point cloud representation with neural fields for arbitrary upsampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 18612–18621. [Online]. Available: https://arxiv.org/abs/2112.04148
[39] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1445–1454, 2009.
[40] B. Lévy and Y. Liu, "Lp centroidal Voronoi tessellation and its applications," *ACM Trans. Graph.*, vol. 29, no. 4, 2010, Art. no. 119.
[41] P. Pulay, "Convergence acceleration of iterative sequences. The case of SCF iteration," *Chem. Phys. Lett.*, vol. 73, no. 2, pp. 393–398, 1980.
[42] T. Washio and C. W. Oosterlee, "Krylov subspace acceleration for nonlinear multigrid schemes," *Electron. Trans. Numer. Anal.*, vol. 6, pp. 271–290, Dec. 1997.
[43] T. Rohwedder and R. Schneider, "An analysis for the DIIS acceleration method used in quantum chemistry calculations," *J. Math. Chem.*, vol. 49, no. 9, pp. 1889–1914, 2011.
[44] A. Toth and C. T. Kelley, "Convergence analysis for anderson acceleration," *SIAM J. Numer. Anal.*, vol. 53, no. 2, pp. 805–819, 2015.
[45] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM J. Numer. Anal.*, vol. 49, no. 4, pp. 1715–1735, 2011.
[46] N. Ho, S. D. Olson, and H. F. Walker, "Accelerating the Uzawa algorithm," *SIAM J. Sci. Comput.*, vol. 39, no. 5, pp. S461–S476, 2017.
[47] Y. Peng et al., "Anderson acceleration for geometry optimization and physics simulation," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018, Art. no. 42.
[48] B. Lévy and N. Bonneel, "Variational anisotropic surface meshing with Voronoi parallel linear enumeration," in *Proc. 21st Int. Meshing Roundtable*, 2013, pp. 349–366.
[49] F. A. Potra and H. Engler, "A characterization of the behavior of the anderson acceleration on linear problems," *Linear Algebra Appl.*, vol. 438, no. 3, pp. 1002–1011, 2013.
[50] J. L. Blanco and P. K. Rai, "nanoflann: A C++ header-only fork of FLANN, a library for nearest neighbor (NN) with KD-trees," 2014. [Online]. Available: https://github.com/jlblancoc/nanoflann
[51] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.
[52] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," iSO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, vol. 7, no. 8, p. 11, 2017.

**Yanyang Xiao** received the Ph.D. degree in computer science from Xiamen University, Xiamen, China, in 2020. He is currently an Assistant Professor with the School of Mathematics and Computer Sciences, Nanchang University, Nanchang, China. His research interests include computer graphics and point cloud processing.

**Tieyi Zhang** received the master's degree in computer science from Xiamen University, Xiamen, China, in 2019. He is currently with NetEase Games, Hangzhou, China.

**Juan Cao** (Senior Member, IEEE) received the Ph.D. degree in applied mathematics from Zhejiang University, Hangzhou, China, in 2009. She is currently a Professor with the School of Mathematical Sciences, Xiamen University, Xiamen, China. Her research interests include computer aided geometric design and computer graphics.

**Zhonggui Chen** (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in applied mathematics from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively. He is currently a Professor with the School of Informatics, Xiamen University, Xiamen, China. His research interests include computer graphics, computational geometry, and digital image processing.