# Approximate Intrinsic Voxel Structure for Point Cloud Simplification

Chenlei Lv, *Member, IEEE,* Weisi Lin, *Fellow, IEEE,* and Baoquan Zhao, *Member, IEEE*

***Abstract*—A point cloud as an information-intensive 3D representation usually requires a large amount of transmission, storage and computing resources, which seriously hinder its usage in many emerging fields. In this paper, we propose a novel point cloud simplification method, Approximate Intrinsic Voxel Structure (AIVS), to meet the diverse demands in real-world application scenarios. The method includes point cloud pre-processing (denoising and down-sampling), AIVS-based realization for isotropic simplification and flexible simplification with intrinsic control of point distance. To demonstrate the effectiveness of the proposed AIVS-based method, we conducted extensive experiments by comparing it with several relevant point cloud simplification methods on three public datasets, including Stanford, SHREC, and RGB-D scene models. The experimental results indicate that AIVS has great advantages over peers in terms of moving least squares (MLS) surface approximation quality, curvature-sensitive sampling, sharp-feature keeping and processing speed. The source code of the proposed method is publicly available[1].**

***Index Terms*—Point cloud simplification, isotropic simplification, intrinsic control, MLS surface, curvature-sensitive sampling, sharp-feature keeping.**

## I. Introduction

With the rapid development of 3D scanning technology, using 3D point clouds to represent real-world objects is becoming increasingly popular in recent years. A point cloud comprises a set of points that carry geometry and possible attribute information such as point position, color, and texture. Unlike a polygonal mesh, it does not contain any edges and facets. Such a data structure is often the primary data format from the sensors, and more versatile and flexible in creating and representing 3D models (thus is being widely adopted in a wide range of applications including 3D object recognition, scene reconstruction, industrial modeling and so on). Although it exhibits unique advantages, there have been several challenges in practice. Firstly, point clouds are generally made up of high-density points. As a result, they can be with huge data volume, which significantly increases the cost of storage and transmission. Secondly, a dense point cloud may contain a considerable number of redundant points. These points are not only dispensable in many cases but could have a severe impact on the efficiency of downstream tasks such as visualization, geometric feature training [1], and localization [2].

This research is supported by the Ministry of Education, Singapore, under its Tier-2 Fund MOE2016-T2-2-057(S).

Chenlei Lv, Weisi Lin and Baoquan Zhao were with the School of Computer Science and Engineering, Nanyang Technological University. The correspoding author is Weisi Lin, (e-mail: wslin@ntu.edu.sg).

[1]https://github.com/vvvwo/AIVS-project.

To tackle the aforementioned challenges, point cloud simplification [3] and compression [4] have been extensively studied to eliminate the redundant information of point clouds. Although these two approaches share similar ultimate goal, their mechanisms and application scenarios are quite different. Compression aims to reduce the size of a point cloud by exploiting the spatial correlation among adjacent points. It can be classified into two categories: lossless compression and lossy compression. Lossless compression is a compression technique that does not lead to a change in the original data of a point cloud, while lossy compression, typified by the trisoup scheme of the MPEG G-PCC, attempts to reduce the data volume of an original point cloud by eliminating some less important information. Simplification, however, reduces the information redundancy of a given point cloud by removing redundant points without significantly affecting the performance of target tasks. These two techniques, in a sense, are complementary and can be integrated into one pipeline. That is, the simplification is first used to obtain a simplified representation of a point cloud, and the compression is then performed on the simplified point cloud to further reduce the overhead of data storage, transmission and processing. In this paper, we mainly focus on point cloud simplification.

There is a large body of research in the literature to suggest effective point cloud simplification schemes, which can be broadly divided into the following two categories, depending on different targets. One kind of simplification work focuses on retaining important local regions in a point cloud, such as key-points [5], and visually sensitive areas [6], but other geometric features of a point cloud may be lost. The other category aims to construct a simplified representation from an original point cloud while keeping important global geometric features. An ideal simplification method that falls into this category is expected to satisfy the following four requirements: 1) distribution uniformity of points in local regions; 2) geometric consistency with the original point cloud; 3) low computational cost; 4) flexibility toward user specifications. To be more specific, distribution uniformity of points in local regions means that the points of simplification result are approximately evenly spread along the underlying local surface of the original point cloud, and this would be beneficial to point cloud visualization and maintaining the inherent adjacency relationship among points. Geometric consistency is referred to a condition that the simplification result holds the important geometric features of the original point cloud. Low computational cost requires that the run-time complexity of a simplification algorithm should be efficient enough to ensure its usability in real-world applications. It also would

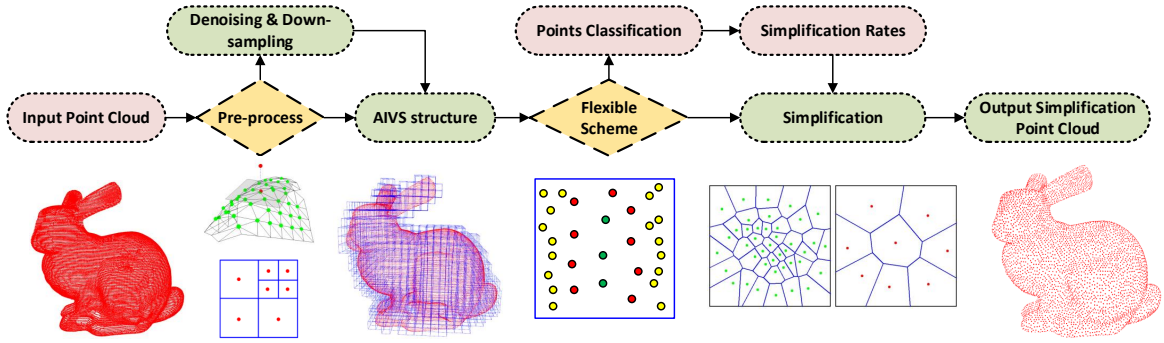



Fig. 1. The pipeline of the proposed AIVS-based point cloud simplification method.

be preferable to allow users to flexibly specify different simplification settings such as curvature-sensitive sampling and sharp-feature keeping. Currently, the most popular methods such as Laplace graph-based simplification [3] and Centroidal Voronoi Tessellation (CVT) [7] resampling cannot satisfy the four conditions at the same time.

To meet these requirements, we propose a point cloud simplification framework called Approximate Intrinsic Voxel Structure (AIVS) that is constructed by two core components: global voxel structure and local farthest point sampling (local FPS). The global voxel structure is used not only to decompose the simplification task of a point cloud into a collection of subtasks that can be executed in parallel, but also to provide intrinsic control of point distance. Based on the intrinsic control of point distance, the simplification can be regarded as an approximately intrinsic one and this keeps better geometric consistency. In our framework, the local FPS is developed to perform the simplification task in each individual voxel box and sample a subset of points that are locally uniform. By combining the global voxel structure and local FPS, the AIVS can efficient simplify point clouds while keeping the isotropic property (globally uniform density). The AIVS also provides a flexible scheme to meet diversified simplification demands, such as curvature-sensitive sampling and sharp-feature keeping. It satisfies all aforementioned four requirements for simplification tasks. The proposed pipeline is shown in Figure 1. The contributions of this paper can be summarized as:

- We propose an Approximate Intrinsic Voxel Structure (AIVS), which provides a voxel box-based organization for a point cloud. It makes a simplified point cloud to be isotropic with the intrinsic control of point distance, even when the input point cloud is non-uniform. Benefited from the proposed intrinsic control, our simplification result maintains better geometric consistency with the original point cloud in terms of MLS surface.
- We provide an efficiently flexible simplification scheme based on AIVS. The points are classified into different categories according to user-specified classification and related sampling rates. With different sampling rates for categories, the different kinds of flexible simplifications such as curvature-sensitive sampling and sharp-feature keeping can be realized.
- We implement the parallel computation for AIVS-based simplification to accelerate the process while keeping the globally or locally uniform density of simplified point clouds.

The rest of the paper is organized as follows. In Sec. II, we introduce some classical works for point cloud simplification. In Sec. III, we propose the formulation of our point cloud simplification, followed by the details of AIVS-based realization in Sec. IV. We demonstrate the effectiveness and efficiency of our method with extensive experimental evidence in Sec. V. Sec. VI concludes the paper.

## II. Related Works

There are many methods in the literature for point cloud simplification, which can be classified into three groups: global simplification, local simplification, and deep learning-based simplification.

The global simplification methods are constructed by different global point sampling strategies. Classical methods include geometric errors optimization [8][9][10], Voronoi diagram [11][12], global clustering [13][14][15][16], and Laplace graph [17][7][3][18]. Alexa et al. provided a point cloud-based surface representation [8], which could be regarded as a basic fundamental for point cloud simplification. This method first defines a 2-manifold from a point cloud using moving least squares (MLS) fitting, and then achieves simplification by optimizing MLS errors. Another important fundamental is intrinsic point cloud simplification, proposed by Moening et al. [11]. The contribution of the work is that it uses the point cloud-based geodesic rather than the Euclidean distance in Voronoi diagram to obtain more reasonable simplification results that are in line with mathematical interpretation and intuitive perception. Laplace graph-based point cloud simplification methods [7][3] are becoming more popular in recent years. Such methods optimize a weighted adjacency matrix which is used to produce a simplified point cloud with uniform density. Although more accurate simplification is achieved, these methods are of high computational complexity and are generally incapable of providing flexible local simplification solutions.

The local simplification methods have also been extensively investigated to improve the effectiveness of simplification and

retain local geometric features. Lee et al. [19] constructed 3D grid structure for point cloud simplification. Xiao et al. [20] proposed an efficient point cloud simplification method based on kd-tree structure. Han et al. [21] used the similar idea for edge-keeping simplification. Some shape descriptors were devised for local geometric analysis and simplification, such as DSO feature [22], sharp-feature keeping [23][24], Gaussian curvature [25], and saliency detection [26]. The extraction of these features is performed on the local regions of a point cloud, and the simplification of the whole point cloud is an aggregation of the simplified points of all local regions. Although local simplification methods provide a more flexible and efficient framework for geometric features keeping, they may run the risk of losing global features and fail to maintain the intrinsic and isotropic properties of a point cloud. For extremely non-uniform regions in a point cloud, those methods without global density analysis can easily result in unsatisfactory simplification outcomes.

Both the aforementioned global and local simplification methods rest on the fact that the simplified points are derived from the original point cloud, without any modification of point positions. In contrast, there are many methods that reduce points of point clouds using resampling schemes such as Centroidal Voronoi Tessellation (CVT) [27][28][29] and particle-based resampling [30][31][32]. These methods resample the points in local tangent space and change the positions of points. Strictly speaking, point resampling can not be regarded as point cloud simplification. The geometric features are smoothed by the resampling and the neighbor points' structures are updated.

Recently, the deep learning-based simplification or sampling methods that are designed for special tasks such as classification, registration and recognition are becoming more popular. These methods, including FoldingNet [33], KCNet [34], SampleNet [35][36], PAT [37], CPL [38], MOPS-Net [39], PIE-NET [40], PointASNL [41], SK-Net [42], etc., attempt to sample the points with sensitive characteristics for effective feature learning. However, they do not achieve a good balance between uniform density and geometric feature keeping. The MLS surfaces of original point clouds are changed after sampling.

In this paper, we focus on the simplification without changing the positions of points and attempt to strike a balance between global and local simplification schemes. Before introducing the details our simplification framework, we propose the formulation of our point cloud simplification in the following section.

## III. Proposed Formulation of Point Cloud Simplification

In this section, we introduce the proposed formulation of our point cloud simplification, which mainly includes two core components: isotropic simplification and flexible simplification. The intrinsic control of point distance is also discussed. It also intends to introduce notations needed in the sections that follow.

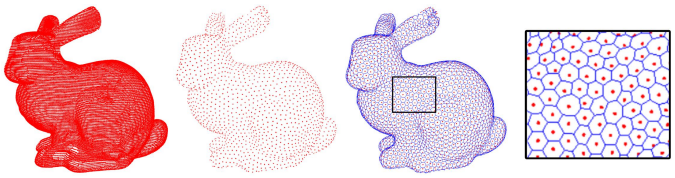


Fig. 2. An instance of isotropic simplification and its 3D Voronoi diagram,

### A. Isotropic Simplification

A point cloud is regarded as a discrete form of a 2-manifold, which is represented by an MLS surface [9]. The contribution of different points to MLS surface representation is different. Some points can be removed without significantly affecting the accuracy of MLS surface representation. Therefore, the goal of simplification is to remove redundant points while keeping the geometric consistency of MLS surface as much as possible. An MLS surface depends on the adjacency among points, which can be represented with a distance field. The simplification of a point cloud can be regarded as a process of selecting a subset of points from the MLS surface while optimizing its distance field. To this end, the isotropic simplification is proposed.

The isotropic simplification means that the distances between any adjacent points are approximately equal in simplification result. It guarantees the correct adjacency among points in a 1-ring region, which avoids the error-fitting in MLS approximation. Figure 2 illustrates an instance of isotropic simplification and its 3D Voronoi diagram. As can be seen from the figure, the Voronoi cells of different points are similar in an isotropic simplification. To achieve isotropic simplification, it needs to optimize a distance field energy, which is defined as

$$E_D = \sum_{i=1}^{n} \int_{r \cap M_s} d(p_i, p_j)\, \mathrm{d}\,\sigma, \tag{1}$$

where $M_s$ represents the MLS surface defined by a point cloud $P$, $n$ is the simplified point number of $P$, $r$ is the 1-ring region of $p_i$, $d(p_i, p_j)$ is the distance between a point $p_i$ and its neighboring point $p_j$ in $r$. According to Equation 1, $E_D$ is mainly determined by $d$ and $r$. In most previous studies [7][17][32], these two variables are defined in a local tangent space. However, it is likely to introduce some errors if there is sharp curvature in local regions. As shown in Figure 3, the Euclidean distance between different points cannot accurately represent the inherent adjacency among points. To avoid this issue, we propose the intrinsic control of point distance. It means that we use the geodesic distance $g$ rather than the Euclidean distance as the distance metric to define the 1-ring region for each point. Then the 1-ring region of a point is regarded as a geodesic Voronoi diagram $G_r$, and the distance field energy can be reformulated as

$$E_G = \sum_{i=1}^{n} \int_{G_r \cap M_s} g(p_i, p_j)\, \mathrm{d}\,\sigma. \tag{2}$$

In [11], a classical simplification method, global farthest point sampling (global FPS), was proposed to simplify point clouds. Theoretically, it provides a reasonable solution to

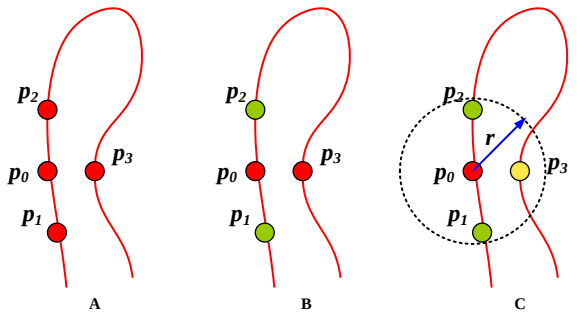



Fig. 3. An instance of the wrong adjacent point based on Euclidean distance. A: the red curve represents the MLS surface; B: the correct adjacent points of point $p_0$ are $p_1$ and $p_2$; C: $p_3$ would be mistakenly identified as a adjacent point of $p_0$.

optimize Equation 2. However, the computation cost of global FPS is huge and this hinders its use in practical applications. To solve the problem, we propose a new simplification framework. Comparing to global FPS, our method improves efficiency of simplification while keeping the intrinsic control of point distance. The implementation will be discussed in Sec. IV-B.

### B. Flexible Simplification

Flexible simplification is used to satisfy special requirements such as curvature-sensitive sampling and sharp-feature keeping. An existing solution is to sample points in a high-dimensional manifold that is constructed by points with normal vectors. In the manifold, the point distance considers the influence of normal vectors. Such a distance reflects the curvature changes in an MLS surface. The flexible simplification is achieved based on the distance $d_h$ in the high-dimensional manifold. In this context, the distance field energy $E_F$ is defined as

$$E_F = \sum_{i=1}^{n} \int_{r \cap M_h} d_h(v_i, v_j)\, \mathrm{d}\,\sigma, \tag{3}$$

and

$$d_h(v_i, v_j) = \rho(v)\, ||v_i, v_j||^L, \tag{4}$$

where $M_s$ represents the MLS surface as mentioned before, $M_h$ is the representation of $M_s$ in the high-dimensional space. $E_F$ for flexible simplification based on curvature is achieved in $M_h$. In $M_h$, a 6-dimensional point $v_i$ is formed by the position (3-dimensional) and normal (3-dimensional) vectors of a point $p_i$ in $M_s$. The point distance in $M_h$ is $d_h$ with $L$ norm. As mentioned before, the isotropic simplification in $M_h$ considers the influence of normal vectors of different points. The density function $\rho(v)$ defines impact of the normal changes for $d_h$. Mapping the isotropic simplification from $M_h$ into $M_s$, the simplification result can be transferred into a curvature-sensitive flexible one. With different kinds of $\rho(v)$ and related $d_h$, various flexible simplification results can be achieved. In our framework, we propose an efficient scheme to implement the flexible simplification.

In our framework, we realize the isotropic and flexible simplification based on AIVS. Benefited from the AIVS, the intrinsic control of point distance can be realized with an efficient way and the simplification can be processed by parallel computing while keeping the globally or locally uniform density. In the following parts, we introduce the details of our simplification methods.

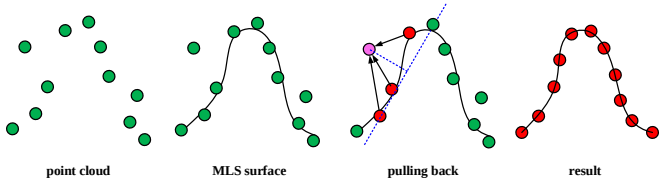



Fig. 4. An instance of the smoothing process of a 2D noisy point cloud.

## IV. AIVS-based Realization

In this section, we elaborate on the proposed AIVS-based point cloud simplification framework from the following three aspects: point cloud pre-processing, AIVS-based isotropic simplification, and AIVS-based flexible simplification. It is worthy noting that these proposed AIVS-based simplification schemes are under the intrinsic control of point distance.

### A. Point Cloud Pre-processing

Many raw point clouds harvested from 3D scanning devices contain noisy or/and non-uniform points. These points could have an adverse impact on the performance of point cloud simplification. Therefore, the pre-processing is needed before simplification. In our framework, the pre-processing includes denoising and density-adjusting.

For denoising, the theory of MLS surface representation is utilized to smooth a point cloud. The smoothing operation updates the location of a point by taking a weighted average of its adjacent points, and this process can be thought of as pulling a noisy point back to the MLS surface. More specifically, for a point $p_i$, it is replaced by a new point $p_i^{'}$ after smoothing, and $p_i^{'}$ is calculated as

$$p_i^{'} = \sum_{j=1}^{k} \frac{\theta(||p_i - p_j||)p_j}{\left|\left|\sum_{j=1}^{k} \theta(||p_i - p_j||)\right|\right|}, \tag{5}$$

and

$$\theta(d) = e^{-\frac{d^2}{h^2}}, \tag{6}$$

where $p_j$ is an adjacent point of $p_i$, $k$ is the total number of adjacent point of $p_i$ in the search radius $h$, and $\theta$ is a distance weighting function. It is suggested to fit a Gaussian function [43] in Equation 6. Without loss of generality, we show an example of the smoothing process of a 2D point cloud in Figure 4.

Next, we adjust the densities of different regions of the obtained point cloud before performing simplification. The reason is that our simplification framework is based on a voxel structure. If the densities in different voxel boxes are not similar, it is difficult to achieve a globally uniform simplification. To adjust the density of a point cloud, we carry out down-sampling using octree and this is widely used in

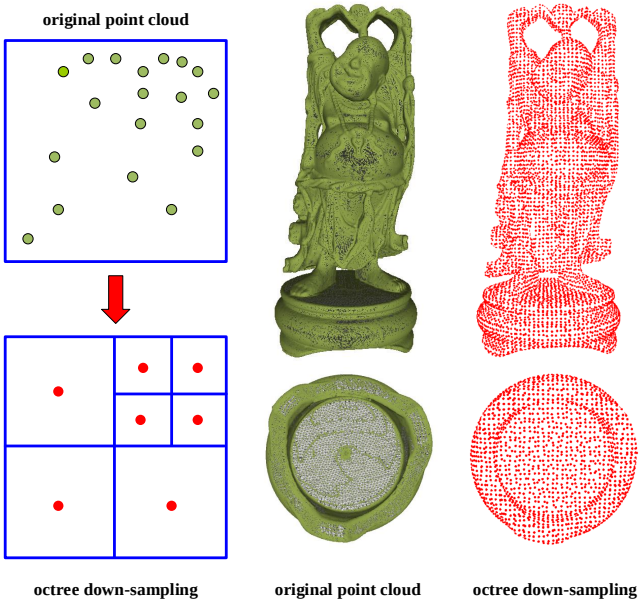



Fig. 5. An instance of octree-based point cloud down-sampling.

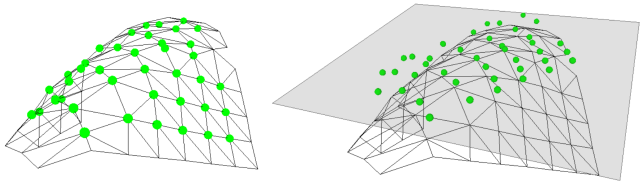


Fig. 6. The original points in a voxel box (left) and the fitting plane (right). The points are approximately located in a local tangent plane.

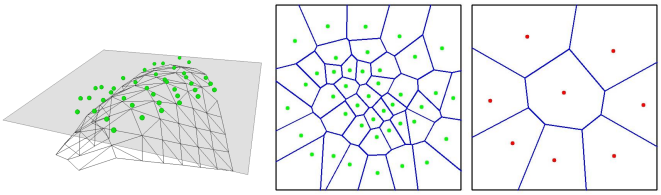


Fig. 7. The simplification of points using the local FPS algorithm. Left: a collection of points on the fitting plane; Middle: the Voronoi diagram of the original points; Right: the Voronoi diagram of the simplified points. The Voronoi diagram shows that the simplification is an isotropic one.

point cloud compression [4] and processing [44]. The down-sampling process reduces the scale of a point cloud and thus improves the efficiency of the whole processing pipeline, especially when dealing with large-scale, high-density point clouds. In Figure 5, we show an instance of octree-based down-sampling. Let $|P|$ and $|P_s|$ be the number of points of an original point cloud $P$ and a simplified point cloud $P_s$, respectively, we estimate the octree voxel size $O_s$ as

$$O_s = max\{d_k(p_i, p_j), p_j \in K(p_i)\}, \tag{7}$$

and

$$k = max(\frac{|P|}{5|P_s|}, 1), \tag{8}$$

where $K(p_i)$ is the set of the $k$-nearest neighbour points of a point $p_i$. The point number $k$ of $K(p_i)$ is estimated by the point number $|P|$ of $P$ and the simplification point number $|P_s|$. The value of $k$ should balance the effectiveness and efficiency of down-sampling. If $k$ is too large, the voxel scale will be increased which makes the point cloud without enough points for simplification with user-specified point number. On the contrary, the point number of input point cloud will not be reduced when the $k$ is too small, which reduces the efficiency of the down-sampling. To keep the balance, we empirically control the value of $k$ by Equation 8. That is, when the original point number is significantly (5 times in this work) more than the targeted simplification point number, the value of $k$ is increased according to 5 integer multiples of the $|P|/|P_s|$; otherwise, it is chosen as 1. The equation makes the balance between effectiveness and efficiency in practice.

### *B. AIVS-based Isotropic Simplification*

Following the formulation in Sec. III-A, we propose an efficient AIVS-based isotropic simplification scheme with the intrinsic control of point distance. The proposed AIVS is a structure similar to voxelization and octree. The points of a given point cloud are divided into different voxel boxes of AIVS. Instead of using the global FPS [11], we use the local FPS as the basic simplification strategy, with which we obtain a simplified point cloud by iteratively searching the target point from a given point cloud. Mathematically, let $P_v$ be the point set in voxel box $v$ and $P_s$ be the simplified point set, the target point $p_s$ in $P_v$ in each iteration is found by

$$p_s = \underset{p_i}{\operatorname{argmax}}\, d(p_i,\ P_s),\ p_i \in P_v,\ p_i \notin P_s, \tag{9}$$

and

$$d(p_i,\ P_s) = \min\{d(p_i,\ p_{s'})\},\ \ p_{s'} \in P_s. \tag{10}$$

That is, the local FPS is an iterative process that finds a point in $P_v$ but not in $P_s$ such that it is farthest from the points in $P_s$. It should be noted that there could be multiple points achieving the maximum in Equation 9. In this case, we randomly select one maximum point and add it to $P_s$. Compared to the global FPS [11], the local FPS uses Euclidean distance to approximate the geodesic distance and processes each voxel independently, and this improves the computational efficiency. Besides, since there is rarely a sharp curvature variation between points in a voxel box, it reduces the possibility of inaccurate MLS surface fitting. We show an example of MLS fitting of a voxel box and its simplification using the local FPS algorithm in Figures 6 and 7, respectively.

The simplification of the whole point cloud is obtained by combining the local FPS results of different voxel boxes. The intrinsic control of point distance is added based on voxel boxes. In Figure 8, we show an example of the intrinsic control of point distance provided by AIVS. The distances between different points are controlled inside the voxel box or adjacent voxel boxes. The points (black) from non-adjacent voxel boxes will not be considered by local FPS in current

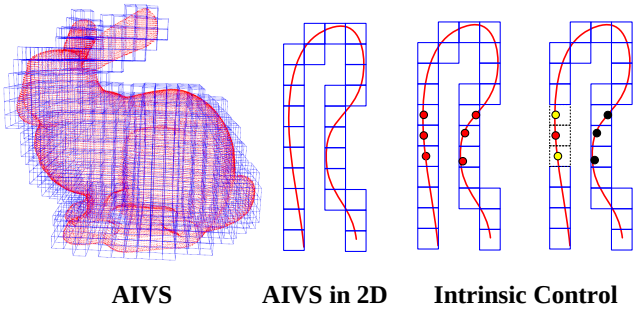



Fig. 8. The intrinsic control of point distance provided by AIVS. The voxel box (blue) controls the adjacent points in the local space (black dotted squares).

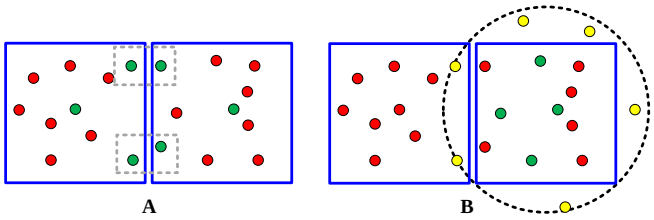



Fig. 9. A: The distances between the points in two dotted grey boxes are not optimized. The distances are too small. B: adding adjacent limit points (yellow) from the adjacent voxel boxes, the isotropic simplification is achieved.

voxel box. The intrinsic control of point distance keeps the geometric consistency between the simplification result and the original point cloud. However, the points' distances are not optimized in the boundary of adjacent boxes, which may result in non-uniform sampling near the boundary of the voxel boxes, as shown in Figure 9A. To solve the problem, we add some points that have been selected into simplification result from the adjacent voxel boxes to limit the non-uniform sampling. These points, as shown in Figure8 and 9B (yellow), are called adjacent limit points. The local FPS considers the adjacent limit points around the boundary when performing simplification. Then, the uniform density can be maintained near the voxel boxes' boundary.

Naturally, the proposed AIVS-based isotropic simplification can be accelerated by parallel computation. Each voxel box corresponds to a parallel computing unit. Considering the adjacent limit points selection, the points from adjacent voxel boxes are not simplified in a parallel computing step, i.e., the adjacent voxel boxes should be arranged in different parallel computing steps. In Figure 10, we show an instance of simplification in different parallel computation steps. Once the sampling point number in each voxel box is determined, the local FPS results can be obtained by parallel computation. Based on the octree-based down-sampling, the point number is proportional to the corresponding MLS surface area. The simplification rate $R_v$ in each voxel box is set to $|P_s|/|P|$. Then the simplification point number $|P_{sv}|$ in a voxel box is $|P_v| \cdot R_v$, where $|P_v|$ is the point number of a voxel box before simplification.

For some small-scale point clouds ($|P| \leqslant 100,000$), the octree-based down-sampling as an independent part is not

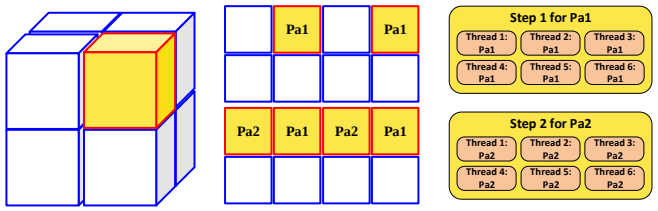



Fig. 10. An instance of parallel computing for simplification. The red box represents the voxel box which is being simplified. The Pa1 and Pa2 are different steps of parallel computing.

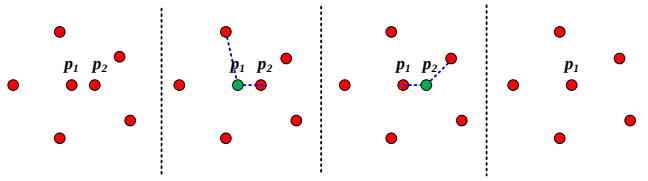



Fig. 11. A point cropping process. Compared to $p_1$, the point $p_2$ satisfies the definition of redundant point. The sum of distance between $p_2$ and its adjacent points is smaller than that of $p_1$. Therefore, the point $p_2$ is removed.

necessary. We simplify such point clouds directly. The simplification rate $R_v$ of a voxel box should be changed. Without the pre-processing of octree-based down-sampling, the uniform density cannot be guaranteed. To solve the problem, we add an MLS surface area estimation for all voxel boxes. The principle is similar to octree. For each voxel box, the points are divided into smaller secondary voxel boxes. If there are no points in a secondary voxel box, the box is deleted. The simplification rate $R_v$ for voxel box $v$ is replaced by the ratio of total secondary voxel boxes number and secondary voxel boxes number in $v$.

By combining the local FPS results of all voxel boxes, the AIVS-based isotropic simplification of the whole point cloud is obtained. However, the number of obtained simplified point cloud may be larger than the one specified by the user. The reason is that the simplification rate $R_v$ for each box produces remainder. Therefore, an accurate point cropping process is required to remove some redundant points so that the outcome is in line with the user-specified number of points. Following the Equation 1, a point $p_i$ that has the closest distance to its adjacent point set is detected as the redundant point $p_d$ and removed; note that the distance between a point and its adjacent point set is defined by the sum of distances between the point and each points in the adjacent point set; i.e., $p_d$ is determined as

$$p_d = \underset{p_i}{\operatorname{argmin}} \sum_{p_j \in R(p_i)} d(p_i, p_j), \tag{11}$$

where point $p_j$ is an adjacent point of $p_i$, and the adjacent point set of $p_i$ is represented as $R(p_i)$. In our implementation, $|R(p_i)|$ is set to 2, to avoid the case that two points are too close. The point cropping process searches redundant points iteratively and deletes them until the number of the simplified point cloud is equal to the user-specified one. In each iteration, if several points share a same minimum distance, one of them

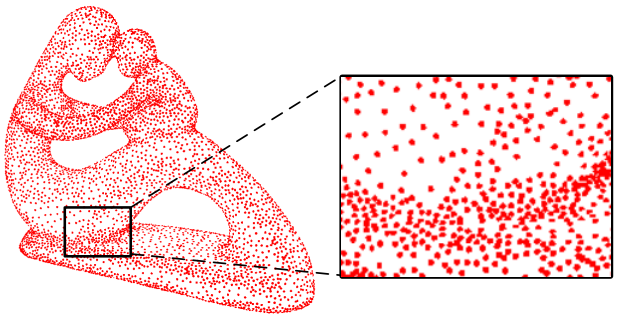


Fig. 12. An instance of curvature-sensitive sampling using the proposed AIVS-based flexible simplification scheme.

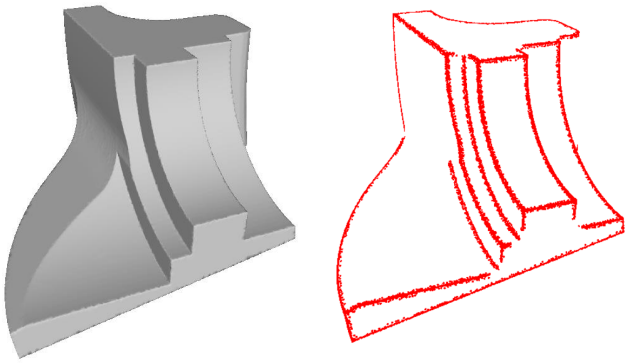


Fig. 13. Edge point detection. Left: a 3D model. Right: The detected edge points are highlighted in red.

is selected to be $p_d$ randomly. In Figure 11, we show an instance of point cropping process.

In a nutshell, the proposed AIVS-based isotropic simplification method can be summarized as follows: 1. build the AIVS for the point cloud after pre-processing so that each point is assigned to a voxel box; 2. divide voxel boxes into different steps for parallel computation; 3. compute the simplified number of points of each voxel box for simplification; 4. perform local simplification on each voxel box using the local FPS algorithm; 5. obtain the simplified results of the whole point cloud by combining the local simplification results of all voxel boxes; 6. reduce the number of points of the simplified point cloud to the user-specified number using the proposed point cropping scheme. In Algorithm 1, we show the related details.

### C. AIVS-based Flexible Simplification

Following the formulation in Sec. III-B, we also propose an AIVS-based flexible simplification method. It constructs a point classification-based simplification scheme to simulate the density function discussed in Equation 4. The density of the simplified points in a voxel box is decided by the simplification rate $R_v$. If the points are classified into different categories with different simplification rates, the flexible simplification result can be achieved. The simplification rate $R_{vi}$ of a category $c_i$ can be computed

$$R_{vi} = \beta a_i, \beta = \frac{|P_s|}{\sum_{i=1}^{k} a_i \, |P_i|}, \tag{12}$$

where $a_i$ is the user-specified rate for category $c_i$, $|P_i|$ is the number of points that belong to $c_i$, and $|P_{vi}|$ is the number of points that belong to $c_i$ in voxel box $v$. For each category, we use the user-specified rate $a_i$ to control the simplification rate $R_{vi}$. With the different simplification rates, the flexible simplification result can be achieved. Considering the point numbers from different categories are not equal, the simplification rate $R_{vi}$ should be limited to a range ($R_{vi} \in [0.1, 1]$) to avoid sampling errors. For example, the input sampling number is larger than exist points or equal to zero. With the range restriction of $R_{vi}$, users can specify different $a_i$ to perform simplification in each voxel box flexibly. In voxel box $v$, $P_{sv}$ represents the simplification result, which is combined from simplification results of related categories. The flexible simplification results of the whole point cloud can be obtained by combining $P_{sv}$ of each voxel box.

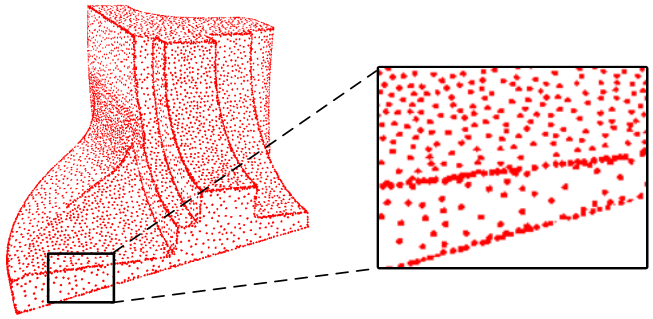


Fig. 14. An instance of sharp-feature keeping using the proposed AIVS-based flexible simplification scheme.

Based on Equation 12, we introduce the implementation of different kinds of flexible simplification, including curvature-sensitive sampling and sharp-feature keeping. For the curvature-sensitive sampling, points are classified into different categories according to their corresponding curvature. The curvature at each point is computed by averaging the normal angles between it and its adjacent points. Then, the point set $P_i$ of category $c_i$ is defined as

$$P_i = \{p_{ci} | Cu(p_{ci}) \in [Cu_{min}(P_i),\ Cu_{max}(P_i)]\}, \tag{13}$$

where $Cu(p_{ci})$ is the curvature of point $p_{ci}$, and $Cu_{min}(P_i)$ and $Cu_{max}(P_i)$ are the lower and upper curvature bound of category $c_i$, respectively. If we equally divide the curvature range of a point cloud into $|c|$ intervals, then the lower and upper bounds of the $i$-th category are: $Cu_{min}(P_i) = (i-1)\pi/(2|c|)$, $Cu_{max}(P_i) = i\pi/(2|c|)$. Figure 12 illustrates an instance of curvature-sensitive sampling using the proposed AIVS-based flexible simplification scheme.

For sharp-feature keeping, the points of a point cloud are classified into two sets: edge point set and ordinary point set. As shown in Figure 13, edge points with sharp features are generally located at the intersection of two MLS surfaces,

**Algorithm 1** AIVS-based isotropic and flexible simplification.

**Require:** Input an original point cloud $P$.
Pre-processing for $P$ (Eq.5 and Eq.7).
Build AIVS for $P$, assign points of $P$ into voxel boxes (Eq.14).
Divide voxel boxes into different steps (like Fig.10).
Input simplification number $|P_s|$.
**if** (isotropic simplification)
  Compute simplification rate $R_v = |P_s|/|P|$.
**else** (flexible simplification)
  Input $\{a_i\}$ and specify classification (Eq.12);
  Classify point cloud $P$ into $\{P_i\}$;
  Count point number $|P_i|$ from $P_i$;
  Compute simplification rate $R_{vi}$ (Eq.12);
  Compute $\{P_{vi}\}$ for each voxel box, $P_{vi} = \{P_v \cap P_i\}$.
**for** s **do** steps
  Add adjacent limited points (Fig.9).
  **if** (isotropic simplification)
    Parallel simplification to sample $P_{sv}$.
    $P_{sv} = \{\text{localFPS}(P_v, R_v)\}$.
  **else** (flexible simplification)
    Parallel simplification to sample $P_{sv}$.
    $P_{sv} = \{\sum_{i \in c} \text{localFPS}(P_{vi}, R_{vi})\}$.
  Add simplification points into $P_s$.
**end for**
Accurate point cropping of $P_s$ (Eq.11).
**Ensure:** Output the simplified point cloud $P_s$.

TABLE I
GEOMETRIC MAXIMUM & AVERAGE ERRORS AND TIME COST WITH DIFFERENT VALUES OF $\varphi$. (10,000 SIMPLIFICATION POINTS).

| Model | Bunny | | | Armadillo | | |
|---|---|---|---|---|---|---|
| $\varphi$ | ΔMax | ΔAvg | Time (s) | ΔMax | ΔAvg | Time (s) |
| 10 | 0.02172 | 0.00260 | 19.577 | —— | —— | —— |
| 20 | 0.02081 | 0.00258 | 7.065 | —— | —— | —— |
| 30 | 0.02284 | 0.00269 | 6.242 | 0.02141 | 0.00331 | 53.915 |
| 40 | 0.02180 | 0.00279 | 4.429 | 0.02402 | 0.00329 | 35.273 |
| 50 | —— | —— | —— | 0.02433 | 0.00332 | 26.951 |
| 60 | —— | —— | —— | 0.02113 | 0.00326 | 24.620 |

where there is a wide variation of curvature. In our implementation, we employ the Voronoi covariance measure (VCM) [45], to detect the edge points of a point cloud. Once the simplification rate for edge points is larger than the rate for ordinary points, the edges in a point cloud are revealed in visualization. We show an instance of sharp-feature keeping using the proposed AIVS-based flexible scheme in Figure 14.

In summary, AIVS-based simplification provides a general and efficient framework for isotropic simplification and flexible simplification. We summarize the proposed simplification framework in Algorithm 1.

## V. EXPERIMENTS

In this section, we carry out extensive experiments to evaluate the efficiency, robustness and complexity of the proposed method. The test point cloud datasets are selected from Stanford [46], SHREC [47], and RGB-D scene [48] models. We implemented the proposed method in C++ with Visual Studio 2019 (64 bit). The experiments were carried out on a PC equipped with a 3.6 GHz Intel Xeon W2133 processor and 32 GB of RAM, and with Windows 10 as its operating system. Firstly, we evaluate the influence of voxelization for our method by the geometric maximum & average error [49][15]. Secondly, we estimate the quality of simplification results from different simplification methods based on the geometric maximum & average and density error analysis. Next, we show the flexible simplification results based on AIVS. Then we evaluate the robustness of our method on different kinds of point clouds, including noisy and multi-objects point clouds. Finally, we provide a further analysis for different methods, including time cost report and evaluation of intrinsic control of point distance.

TABLE II
DENSITY ERRORS OF DIFFERENT SIMPLIFICATION METHODS ON THE STANFORD MODELS (10,000 SIMPLIFICATION POINTS).

| Methods | MLS | Hierarchy | Wlop | Laplace | AIVS |
|---|---|---|---|---|---|
| Bunny | 0.06855 | 0.10612 | 0.05162 | 0.04692 | **0.03271** |
| Horse | 0.05517 | 0.05632 | 0.03084 | 0.02548 | **0.01595** |
| Armadillo | 0.05952 | 0.13631 | 0.04584 | 0.02895 | **0.01753** |
| Angel | 0.03715 | 0.09064 | 0.05739 | 0.03701 | **0.01247** |
| Dragon | 0.06349 | 0.13661 | 0.05912 | 0.05062 | **0.02777** |
| Buddha | 0.07860 | 0.11272 | 0.08526 | 0.05292 | **0.02306** |
| Asian-Dragon | —— | 0.03623 | 0.03731 | —— | **0.02176** |

TABLE III
DENSITY ERRORS OF DIFFERENT SIMPLIFICATION METHODS ON THE STANFORD MODELS (0.1 SIMPLIFICATION RATE).

| Methods | MLS | Hierarchy | Wlop | Laplace | AIVS |
|---|---|---|---|---|---|
| Bunny | 0.16212 | 0.17510 | 0.07571 | 0.07091 | **0.04268** |
| Horse | 0.10162 | 0.11932 | 0.04542 | 0.03771 | **0.02153** |
| Armadillo | 0.04912 | 0.08707 | 0.03315 | 0.02484 | **0.01446** |
| Angel | 0.02653 | 0.07047 | 0.03593 | 0.03167 | **0.00907** |
| Dragon | 0.03402 | 0.06741 | 0.03679 | 0.03683 | **0.02019** |
| Buddha | 0.04007 | 0.10051 | 0.05093 | 0.04651 | **0.01191** |
| Asian-Dragon | —— | 0.03147 | 0.01336 | —— | **0.00996** |

### A. Voxelization

According to the discussion in Sec. IV-A, the density of voxelization could have an influence on the quality of simplification. Let's consider two extreme cases: 1. there is only one voxel box for AIVS, and all points of the input point cloud are assigned to the box; 2. the size of voxel box is too small and each voxel box contains no more than one point. For the first case, the simplification algorithm degenerates to the global FPS simplification and the intrinsic control of point distance is disable. For the second case, we are unable to perform simplification in each voxel box independently. The point number of simplification in each voxel box is equal to zero. Therefore, the selection of voxelization scale should avoid the two cases.

Actually, it is difficult to find a golden rule that can be used for voxel box size selection in point cloud simplification. The reason is that the point distributions of different point clouds vary greatly. In our implementation, we roughly estimate the voxel box size of a point cloud as follows. For a given point cloud, we first find its rectangular bounding box and define

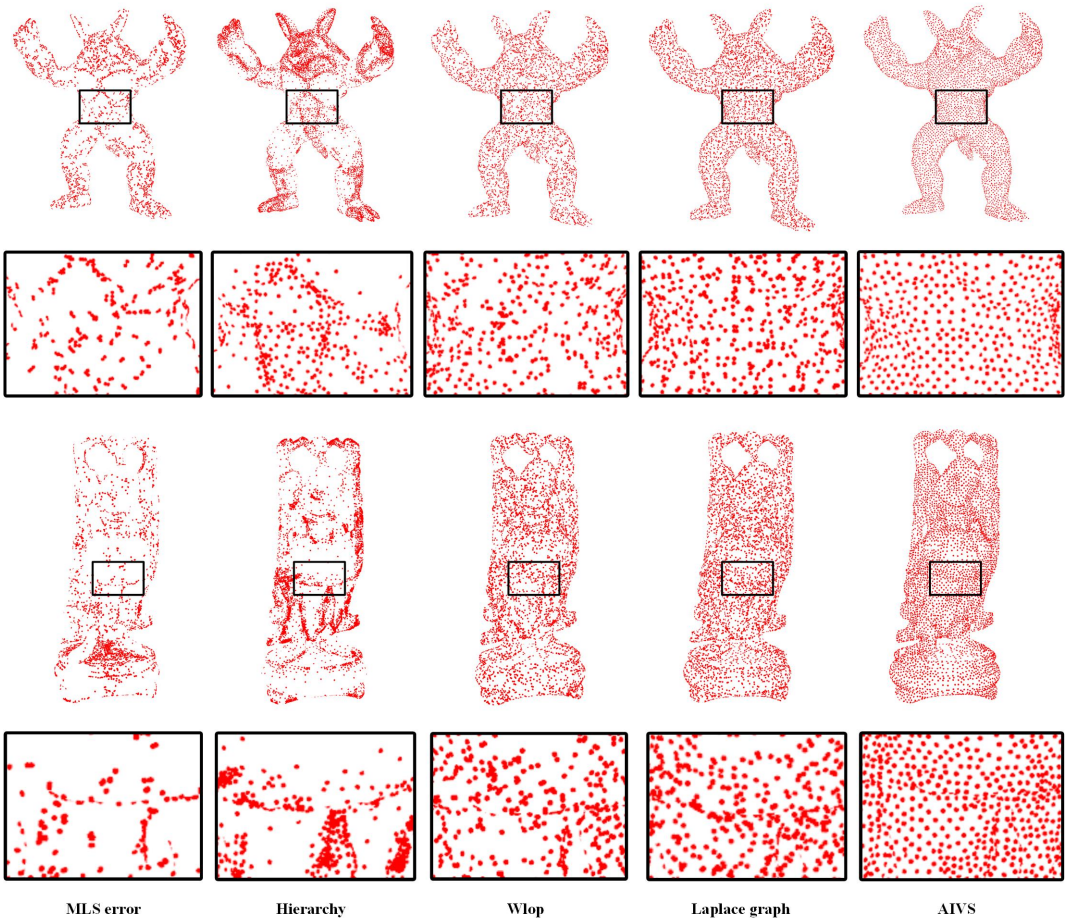



Fig. 15. Comparison of the simplification results obtained with different methods (10,000 simplification points from the Armadillo model and Buddha model).

the long axis of the box as the standard axis. Let $l$ be the length of the standard axis, the size of a voxel box is $l/\varphi$ if we equally divide the axis into $\varphi$ intervals. In Table I, the geometric maximum & average error and time cost are shown for simplification with different values of $\varphi$ (some results with extremely value of $\varphi$ are removed). By analyzing the results in Table I, we found that we are able to achieve the aforementioned balance if $\varphi$ is estimated by

$$\varphi = \left[ \frac{\sqrt[3]{|P|}}{2} \right], \tag{14}$$

where $|P|$ is the point number of an original point cloud. The square brackets mean taking the integral multiple of 10.

### B. Evaluation of AIVS-based Isotropic Simplification

To further illustrate the performance of our simplification method, we compare the simplification results from different methods based on geometric maximum & average and density error analysis. The density error analysis is used to estimate the isotropic property of different simplification results. We organize the points of the simplified point cloud into a kd-tree structure and find the $k$-nearest neighbour point set $K(p_i)$ for each point $p_i$ ($k$ is set to 8 in our implementation). Let $\overline{d_i}$ be the average distance between $p_i$ and the points in $K(p_i)$, then the density error $\Delta den$ can be computed as

$$\Delta \text{den} = max\{\overline{d_i}\} - \min\{\overline{d_i}\}, \tag{15}$$

and

$$\overline{d_i} = \sum_{p_j \in K(p_i)} d(p_i, p_j)/k. \tag{16}$$

We compare the proposed method with several classical peer ones in the literature, including MLS error optimization [8], Hierarchical simplification [13], Wlop [14], and Laplace graph [3]. The Hierarchical simplification and Wlop were implemented by Computational Geometry Algorithms Library (CGAL). The Hierarchical simplification can not always output a simplified point cloud with user-specified point number. For fair comparison, we manually adjust its parameters to control the point number in simplification result. The MLS error optimization and Laplace graph were implemented in our soft platform. The quadratic optimization task of the Laplace graph is solved by MOSEK, which is an efficient optimization library. In Figure 15, we compare the simplification results of different methods. As can be observed from the figure, the simplified point clouds obtained with our method are more evenly distributed compared to peer ones.

To further prove the effectiveness of the proposed framework, we show the quantitative comparison of different sim-

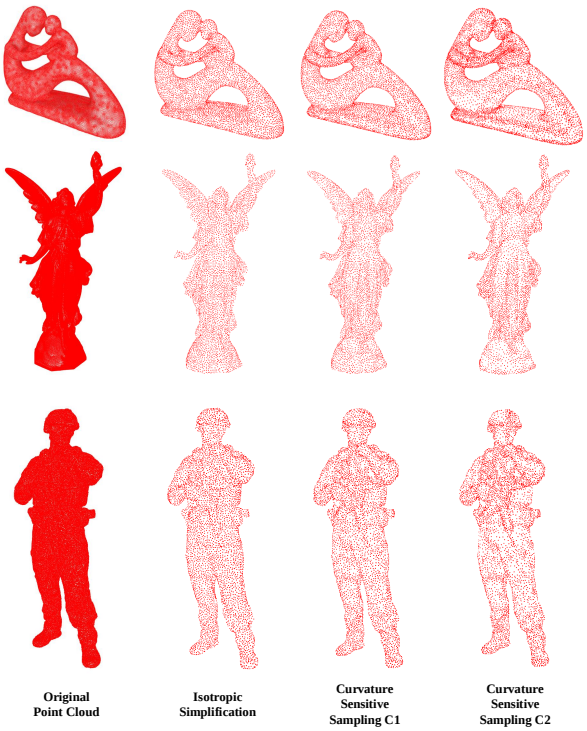



Fig. 16. Curvature-sensitive sampling based on AIVS. From left to right: original point cloud models; isotropic simplification results with 10,000 points; curvature-sensitive sampling with 10,000 points by C1 configuration {2, 3, 4, 5, 6}; curvature-sensitive sampling with 10,000 points by C2 configuration {1, 2, 3, 4, 5}.

plification methods in Tables II - V. Some experimental results with excessive time cost (more than two hours) are removed. The density error results of different methods are shown in Tables II and III. From these two tables, we can see that our simplification method achieves better performance in terms of density optimization. In Tables IV and V, we show the geometric maximum & average errors of different methods. The results verify that our method has better MLS surface fitting performance than peer methods.

### C. Evaluation of AIVS-based Flexible Simplification

In Sec. IV-C, we have introduced that the different kinds of flexible simplifications can be realized by AIVS-based flexible simplification scheme. In this part, we evaluate the efficiency of different flexible simplification schemes, including curvature-sensitive sampling, sharp-feature keeping simplification, and axis-based flexible simplification.

For the curvature-sensitive sampling, the category number $c$ in Equation 13 has a great impact on point cloud simplification. This is because if $c$ is too large, the points in each class are insufficient for local simplification in a voxel box; else if $c$ is too small, the density changes of the adjacent point sets with different curvature values are not continuous and smooth. In our implementation, we set $c$ to 5, and thus have 5 curvature ranges: $c_1 \in [0, \pi/10)$, $c_2 \in [\pi/10, \pi/5)$, $c_3 \in [\pi/5, 3\pi/10)$, $c_4 \in [3\pi/10, \pi/3)$, $c_5 \in [\pi/3, \pi/2]$. Then, the points of the input point cloud are classified into different categories according to the ranges. A configuration $C$ is defined by a set of ratios $\{a_i\}$ for $\{c_i\}$. In Figure 16, we show the curvature-sensitive sampling results with different configurations. It shows that the AIVS can achieve different kinds of curvature-sensitive sampling results with user-specified configurations.

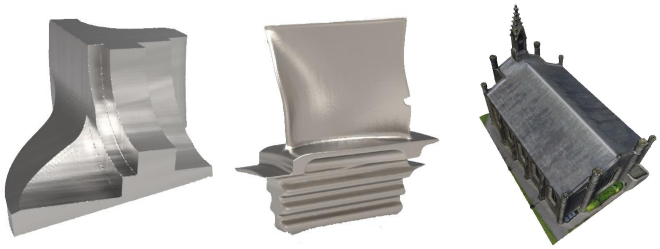


Fig. 17. 3D objects (Fandisk, Blade and Church) with sharp-features.

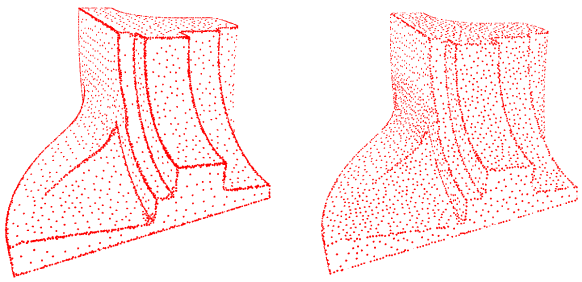


Fig. 18. Sharp-feature keeping simplification (5,000 points) with different configurations based on AIVS. Configuration: left {1, 9} and right {3, 7}.

For simplification with sharp-feature keeping, the goal is to keep the sharp edges of a point cloud for visualization and geometric feature-based analysis. In Figure 17, we show three 3D objects with sharp features. The implementation of the simplification based on AIVS is similar to curvature-sensitive sampling. The difference is that the point classification is based on the sharp feature. The sharp-feature keeping simplification results obtained with different configurations are shown in Figure 18. The sharp edges are more clear by the configuration with a higher ratio for edge points.

To evaluate the efficiency of the proposed sharp-feature keeping simplification scheme, we compare it with several peer simplification methods. In [14] and [23], the authors discussed that the Wlop simplification is a balanced scheme for uniform sampling and sharp-feature keeping. The Laplace graph-based simplification [3] increases the weights of the points in the sharp edge by reducing the balance parameter $\lambda$. We set $\lambda$ to 0.2 to increase the weight of sharp-feature sensitivity. To control the parameters (max cluster size and surface variation) of the Hierarchical simplification, a sharp-feature sensitive simplification result is achieved. In Figure 19, we compare the sharp-feature keeping simplification results obtained with different methods. The configuration for AIVS is {1, 9} ($a_1$ of ordinary point set is set to 1 and $a_2$ of edge point set is set to 9). The results show that the AIVS-based simplification achieves a better balance between locally uniform sampling and sharp-feature keeping. Even the simplification point number is small, the sharp-feature also can

TABLE IV
GEOMETRIC MAXIMUM & AVERAGE ERRORS OF DIFFERENT METHODS ON THE STANFORD MODELS (10,000 SIMPLIFICATION POINTS).

| **Methods** | **MLS error** | | **Hierarchy** | | **Wlop** | | **Laplace graph** | | **AIVS** | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** |
| Bunny | 0.02541 | 0.00329 | 0.06283 | 0.00319 | 0.02761 | 0.00298 | 0.02483 | 0.00297 | **0.02082** | **0.00258** |
| Horse | 0.03356 | 0.00221 | 0.03207 | 0.00175 | 0.03513 | 0.00179 | 0.01704 | 0.00184 | **0.01780** | **0.00164** |
| Armadillo | 0.09539 | 0.00508 | 0.04834 | 0.00505 | 0.02702 | 0.00337 | **0.02072** | 0.00330 | 0.02401 | **0.00329** |
| Angel | 0.05130 | 0.00474 | 0.05674 | 0.00448 | 0.04446 | 0.00252 | 0.02345 | **0.00243** | **0.01605** | 0.00244 |
| Dragon | 0.07126 | 0.00539 | 0.07059 | 0.00848 | 0.03178 | 0.00361 | 0.02502 | 0.00346 | **0.02499** | **0.00344** |
| Buddha | 0.09334 | 0.00725 | 0.00828 | 0.00740 | 0.07465 | 0.00415 | 0.02439 | 0.00391 | **0.02198** | **0.00381** |
| Asian Dragon | —— | —— | **0.02083** | 0.00326 | 0.03270 | 0.00302 | —— | —— | 0.02597 | **0.00289** |

TABLE V
GEOMETRIC MAXIMUM & AVERAGE ERRORS OF DIFFERENT METHODS ON THE STANFORD MODELS (0.1 SIMPLIFICATION RATE).

| **Methods** | **MLS error** | | **Hierarchy** | | **Wlop** | | **Laplace graph** | | **AIVS** | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** | **ΔMax** | **ΔAvg** |
| Bunny | 0.09732 | 0.00815 | 0.07272 | 0.008858 | 0.04204 | 0.00646 | 0.03703 | 0.00611 | **0.03665** | **0.00603** |
| Horse | 0.07611 | 0.00480 | 0.03582 | 0.00497 | 0.03223 | 0.00302 | 0.02559 | 0.00313 | **0.02242** | **0.00291** |
| Armadillo | 0.07491 | 0.00355 | 0.03502 | 0.0032 | 0.02234 | 0.00245 | 0.01607 | 0.00231 | **0.01439** | **0.00229** |
| Angel | 0.05621 | 0.00233 | 0.05135 | 0.002401 | 0.04329 | 0.00134 | 0.02602 | 0.00136 | **0.01330** | **0.00129** |
| Dragon | 0.02759 | 0.00173 | 0.03492 | 0.00181 | 0.02455 | 0.00123 | 0.01779 | 0.00134 | **0.00981** | **0.00114** |
| Buddha | 0.04376 | 0.00184 | 0.06175 | 0.00272 | 0.03371 | 0.00124 | 0.02631 | 0.00129 | **0.01357** | **0.00121** |
| Asian Dragon | —— | —— | 0.01221 | 0.00065 | 0.00725 | **0.00026** | —— | —— | **0.00564** | 0.00027 |

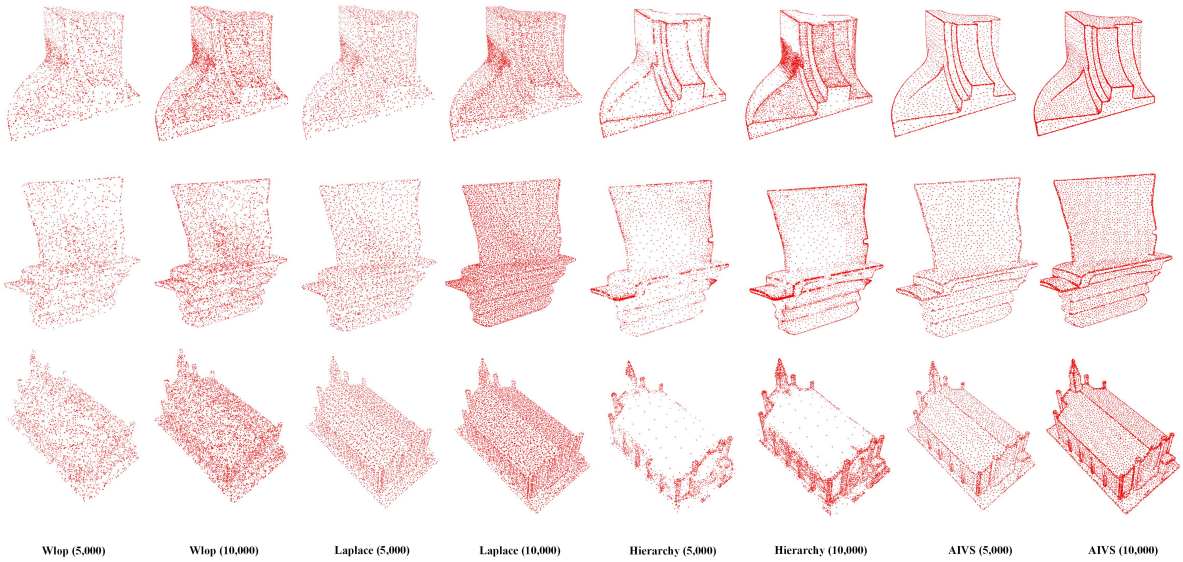



Fig. 19. Comparisons of sharp-feature keeping simplification for Fandisk, Blade and Church with different simplification points (5,000 and 10,000).

be maintained by our method.

In fact, AIVS provides a high degree of freedom for the flexible simplification design to meet users' demands. For instance, a point cloud-based application requires a flexible simplification which should satisfy the perspective principle i.e., points that are near to the eyes appear more dense than those that are far away. Using the proposed AIVS-based flexible simplification, such requirements can be fulfilled. We provide an axis-based flexible simplification to demonstrate the function. Given a point cloud, we classify its points into different classes according to their $Y$-coordinates. The category number $c$ is also set to 5, as did in curvature-sensitive sampling. Let $Y_{max}$ and $Y_{min}$ be the maximum and minimum value of the $Y$-coordinates of all points, respectively, and $Y_d(Y_d = Y_{max} - Y_{min})$ be the range of $Y$-coordinates, we define the ranges of the five classes as: $c_1 \in [Y_{min}, Y_{min} + 0.2Y_d)$, $c_2 \in [Y_{min} + 0.2Y_d, Y_{min} + 0.4Y_d)$, $c_3 \in [Y_{min} + 0.4Y_d, Y_{min} + 0.6Y_d)$, $c_4 \in [Y_{min} + 0.6Y_d, Y_{min} + 0.8Y_d)$, $c_5 \in [Y_{min} + 0.8Y_d, Y_{max}]$. Using a configuration $C$ with a set of ratio $a_i$, an axis-based flexible simplification result is achieved based on AIVS. In Figure 20, we show the instances of such simplification results with different configurations. The results simulate a perspective effect just like a realistic illumination in a point cloud.

### *D. Evaluation of the robustness*

In this part, we evaluate the robustness of different methods on noisy and multi-objects point clouds. As aforementioned, the pre-processing of our method improves the quality of input point clouds. We collect a test set from SHREC models

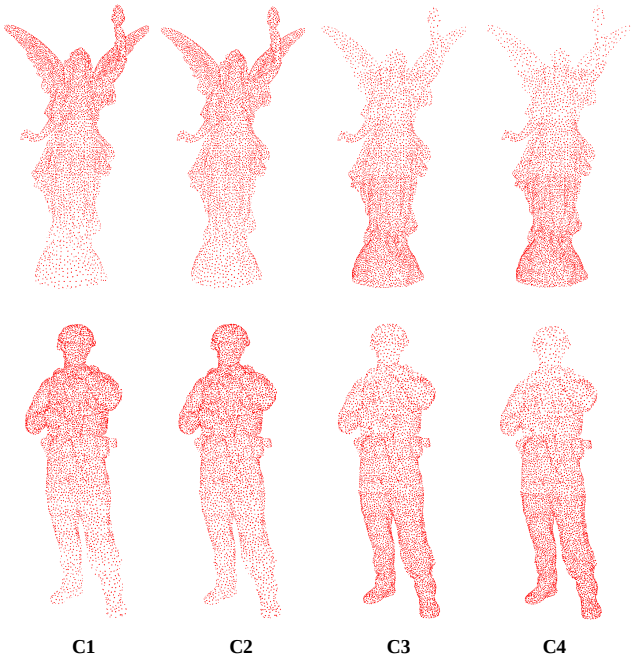



Fig. 20. Instances of axis-based flexible simplification. The configurations from left to right: C1{1, 2, 3, 4, 5}; C2{2, 3, 4, 5, 6}; C3{6, 5, 4, 3, 2}; C4{5, 4, 3, 2, 1}.

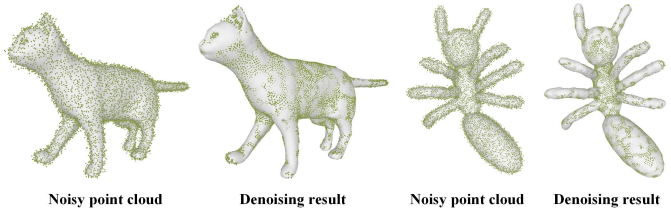



Fig. 21. Two denoising examples obtained with the proposed pre-processing scheme.

with Gaussian noise to conduct the experiment. In Figure 21, we compare the point cloud models with noise and the corresponding denoising results. It is clear that the pre-processing recovers the MLS surfaces from noisy point clouds. To quantify such improvement, we compare the proposed method with Wlop [14] and SampleNet [36] in terms of geometric average error ($\Delta$Avg). The results are illustrated in Figure 22, from which we can see that the geometric average error of the proposed method is significantly lower than that of peer methods, and this proves that our method is more robust to noisy point clouds and keeps better geometric consistency.

In previous experiments, the test point clouds are single-object models. To further demonstrate the effectiveness of our method, we collect another test dataset from RGB-D scene models. In this dataset, point clouds are mainly multi-objects models like indoor scenes. We show some instances of simplification results and the geometric average errors of different methods in Figures 23 and 24, respectively. The SampleNet samples a critical point set as the simplification result. However, the SampleNet breaks the MLS surface of the original point cloud. The Wlop achieves the simplification

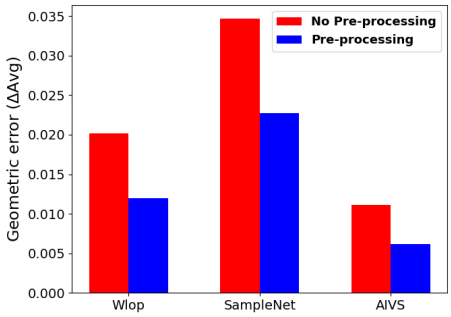



Fig. 22. Geometric average error ($\Delta$Avg) comparison of different simplification methods with/without the pre-processing. It is clear that the pre-processing improves the quality of simplification for different methods. With the pre-processing, our method achieves better geometric consistency.

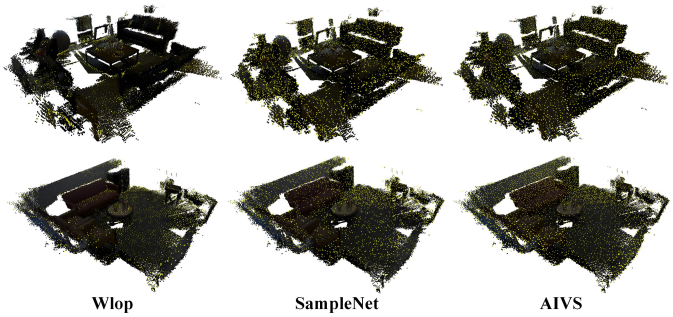



Fig. 23. Instances of multi-objects point clouds simplification results obtained with different methods. The simplified points are highlighted in yellow.

result with non-uniform density. Compared to the above methods, our simplification method achieves better balance between uniform density and geometric consistency.

### *E. Further Analysis of AIVS-based Simplification*

The time complexity of AIVS without parallel computation is $O(nmlog(m))$, where $n$ is the voxel box number, and $m$ is the average point number in each voxel box. For each voxel box, the simplification time complexity can be regarded as the local FPS, which is equal to sort a set of points based on point distances. Using the parallel computation, the time complexity of AIVS can be reduced to $O(8mlog(m))$. The neighborhood size is 8 for each voxel, which means the number of parallel computation steps is 8. In our framework, we use the OpenMP to implement the parallel computation. In Table VI and VII, we compare the computation time of different methods on different point clouds. The running time comparison of the methods is shown in Figure 25.

Benefited from the intrinsic control of point distance, our simplification achieves better neighbor structure for a point cloud, especially for the region with sharp curvature changes. Using the accurate neighbor structure, our simplification keeps better geometric consistency between simplification result and original point cloud. To evaluate the effectiveness of intrinsic control of point distance, we use a reconstruction method [50] to rebuild the mesh based on CVT [27] and our simplification result. The comparison is shown in Figure 26. It is clear

TABLE VI
TIME COST OF DIFFERENT METHODS IN SECONDS (10,000 SIMPLIFICATION POINTS).

| Methods | MLS | Hierarchy | Wlop | Laplace | AIVS |
|---|---|---|---|---|---|
| Bunny | 192.42 | 6.06 | 207.51 | 217.91 | **2.86** |
| Horse | 301.69 | 7.65 | 285.56 | 316.51 | **4.71** |
| Armadillo | 2225.32 | 18.23 | 206.19 | 1152.36 | **14.45** |
| Angel | 3699.47 | 25.06 | 342.70 | 1882.06 | **21.20** |
| Dragon | 11112.92 | 42.88 | 325.82 | 2712.82 | **28.69** |
| Buddha | 16458.72 | 53.81 | 367.94 | 4321.31 | **31.26** |
| Asian-Dragon | —— | 359.01 | 4271.37 | —— | **60.23** |

TABLE VII
TIME COST OF DIFFERENT METHODS IN SECONDS (0.1 SIMPLIFICATION RATE).

| Methods | MLS | Hierarchy | Wlop | Laplace | AIVS |
|---|---|---|---|---|---|
| Bunny | 231.81 | 3.91 | 62.20 | 206.48 | **2.67** |
| Horse | 334.21 | 5.51 | 113.59 | 314.94 | **4.08** |
| Armadillo | 2150.22 | 22.61 | 349.19 | 1174.19 | **14.66** |
| Angel | 3521.07 | 30.29 | 460.70 | 1895.78 | **22.21** |
| Dragon | 10192.72 | 60.35 | 1384.23 | 2742.64 | **28.51** |
| Buddha | 15394.71 | 75.72 | 1875.78 | 4441.88 | **35.64** |
| Asian-Dragon | —— | 548.39 | 15878.43 | —— | **61.23** |

that our simplification improves the accuracy of the reconstructed mesh in the region with sharp curvature changes. As our method achieves more accurate neighbor structure, some incorrect connection between points can be avoided. The inner borders can be reconstructed by such property. In Figure 27, we show the inner border reconstruction results. Our method reconstructs better inner borders.

In summary, our simplification method is proved to have better performance both in accuracy, flexibility and time consumption. Based on the local structure of AIVS, the simplification task can be processed without global analysis. The intrinsic and isotropic properties can be maintained during the local FPS sampling in each voxel box. The proposed method is capable of simplifying a given point cloud as specified by a user. In addition, it provides a flexible scheme to meet the diverse needs of simplification such as curvature-sensitive sampling, sharp feature keeping and axis-based flexible simplification. The time cost can be reduced by the parallel structure of AIVS.

## VI. CONCLUSIONS

We have proposed a point cloud simplification method based on Approximate Intrinsic Voxel Structure (AIVS). The AIVS divides the simplification task with voxel boxes. It provides the intrinsic control of point distance in an efficient way, which improves the accuracy of geometric consistency. Benefited from the local FPS and the intrinsic control of point distance, the AIVS-based isotropic simplification achieves global uniform simplification without complex optimization. According to different point categories with different simplification rates, a flexible simplification scheme has also been designed based on specifications whenever necessary. Different flexible simplification sub-tasks can be processed, including curvature-sensitive sampling, sharp-feature keeping,

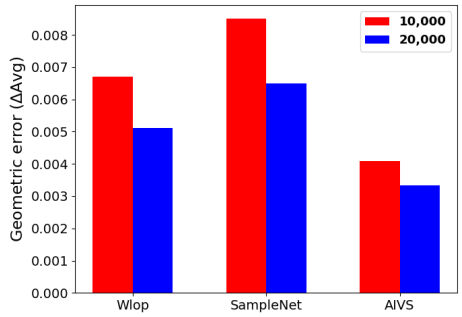



Fig. 24. Geometric average error (ΔAvg) comparison of different methods with two simplification point numbers: 10,000 (red) and 20,000 (blue).

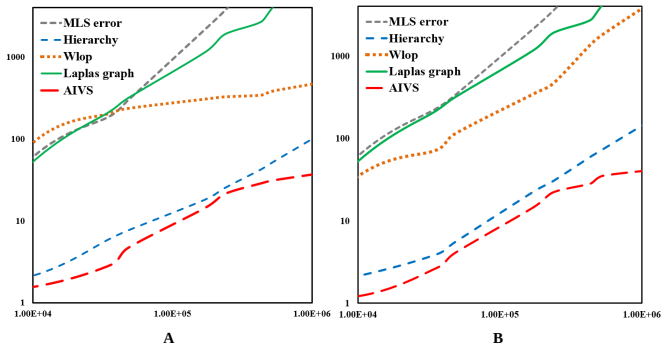



Fig. 25. Running time comparison of different simplification methods. A and B show the time cost in the setting of 10,000 simplification points and a simplification rate of 0.1, respectively. The x-axis of A and B represents the number of points of an original point cloud, and the y-axis is the time cost (in seconds).

and axis-based flexible simplification. At the same time, the flexible simplification scheme keeps the balance between such task requirements (curvature sensitive, sharp-feature keeping, etc) and locally uniform point density. The proposed solution also allows parallel computation to be used to improve the simplification efficiency.

## REFERENCES

[1] J. Yang, S. Quan, P. Wang, and Y. Zhang, "Evaluating local geometric feature representations for 3d rigid data matching," *IEEE Transactions on Image Processing*, vol. 29, pp. 2522–2535, 2020.
[2] G. Elbaz, T. Avraham, and A. Fischer, "3d point cloud registration for localization using a deep neural network auto-encoder," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4631–4640.
[3] J. Qi, W. Hu, and Z. Guo, "Feature preserving and uniformity-controllable point cloud simplification on graph," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 284–289.
[4] R. Schnabel and R. Klein, "Octree-based point-cloud compression." in *Eurographics Symposium on Point-Based Graphics*, vol. 6, 2006, pp. 111–120.
[5] S. M. Prakhya, B. Liu, W. Lin, V. Jakhetiya, and S. C. Guntuku, "B-shot: A binary 3d feature descriptor for fast keypoint matching on 3d point clouds," *Autonomous Robots.*, vol. 41, no. 7, pp. 1501 – 1520, 2017.
[6] W. Cheng, W. Lin, X. Zhang, M. Goesele, and M.-T. Sun, "A data-driven point cloud simplification framework for city-scale image-based localization," *IEEE Transactions on Image Processing.*, vol. 26, no. 1, pp. 262 – 275, 2017.
[7] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kovačević, "Fast resampling of three-dimensional point clouds via graphs," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666–681, 2017.

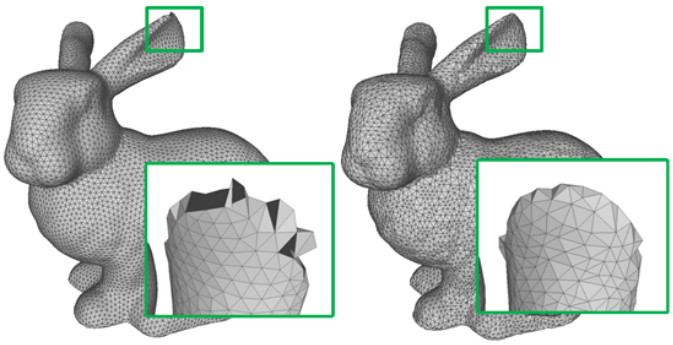


Fig. 26. Instances of reconstructed meshes. Left: reconstructed mesh from CVT resampling result; Right: reconstructed mesh from our simplification result.

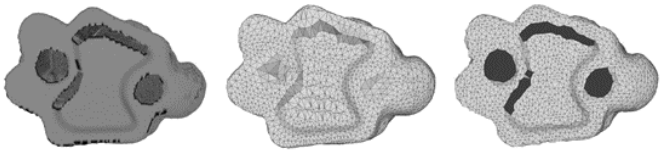


Fig. 27. Instances of reconstructed meshes with inner borders. From left to right: Ground truth; CVT reconstruction result; Our reconstruction result. Benefited from intrinsic control, the inner borders can be reconstructed from our simplification result, which improves the accuracy of reconstruction and keeps better geometric consistency.


[8] M. Alexa, J. Behr, and D. Cohen, "Point set surfaces," in *Visualization*, 2001, pp. 21 – 28.

[9] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[10] H. Song and H.-Y. Feng, "A global clustering approach to point cloud simplification with a specified data reduction ratio," *Computer-Aided Design*, vol. 40, no. 3, pp. 281–292, 2008.

[11] C. Moening and N. Dodgson, "A new point cloud simplification algorithm," in *International conference on visualization, imaging and image processing*, 2003, pp. 1027 – 1033.

[12] ——, "Intrinsic point cloud simplification," in *International conference on computer graphics and vision*, 2004, pp. 1147 – 1154.

[13] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 163–170.

[14] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 176.1 – 176.7, 2009.

[15] Y. Miao, R. Pajarola, and J. Feng, "Curvature-aware adaptive re-sampling for point-sampled geometry," *Computer-Aided Design.*, vol. 41, no. 6, pp. 395 – 403, 2009.

[16] B. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using k-means clustering," *Computer-Aided Design.*, vol. 43, no. 8, pp. 910 – 922, 2011.

[17] C. Luo, X. Ge, and Y. Wang, "Uniformization and density adaptation for point cloud data via graph laplacian," *Computer Graphic Fourm*, vol. 37, no. 1, pp. 325 – 337, 2018.

[18] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model," *IEEE Transactions on Image Processing*, vol. 29, p. 3474 – 3489, 2018.

[19] K. Lee, H. Woo, and T. Suk, "Point data reduction using 3d grids," *The International Journal of Advanced Manufacturing Technology*, vol. 18, no. 3, pp. 201 – 210, 2001.

[20] Z. Xiao and W. Huang, "Kd-tree based nonuniform simplification of 3d point cloud," in *International Conference on Genetic and Evolutionary Computing*, 2009, pp. 339 – 342.

[21] H. Han, X. Han, F. Sun, and C. Huang, "Point cloud simplification with preserved edge based on normal vector," *Optik*, vol. 126, p. 2157–2162, 2015.

[22] P. Lee and C. Huang, "The dso feature based point cloud simplification," in *International Conference Computer Graphics, Imaging and Visualization*, 2011, pp. 1–6.

[23] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Transactions on Graphics*, vol. 32, no. 1, pp. 1– 12, 2013.

[24] H. Benhabiles, O. Aubreton, H. Barki, and H. Tabia, "Fast simplification with sharp feature preserving for 3d point clouds," in *International Symposium on Programming and Systems*, 2013, pp. 47 – 52.

[25] K. Liu, J. Chen, S. Xing, and H. Han, "Simplification of point cloud data based on gaussian curvature," in *IET International Conference on Smart and Sustainable City*, 2013, pp. 84 – 87.

[26] X. Ding, W. Lin, Z. Chen, and X. Zhang, "Point cloud saliency detection by local and global feature fusion," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5379 – 5393, 2019.

[27] Z. Chen, T. Zhang, and J. C. et al, "Point cloud resampling using centroidal voronoi tessellation methods," *Computer-Aided Design*, vol. 102, p. 12 – 21, 2018.

[28] D. Boltcheva and B. Lévy, "Surface reconstruction by computing restricted voronoi cells in parallel," *Computer-Aided Design*, vol. 90, pp. 123–134, 2017.

[29] N. Ray, D. Sokolov, S. Lefebvre, and B. Lévy, "Meshless voronoi on the gpu," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–12, 2018.

[30] Z. Zhong, X. Guo, W. Wang, B. Lévy, F. Sun, Y. Liu, W. Mao *et al.*, "Particle-based anisotropic surface meshing," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 99–1, 2013.

[31] S. Ni, Z. Zhong, J. Huang, W. Wang, and X. Guo, "Field-aligned and lattice-guided tetrahedral meshing," in *Computer Graphics Forum*, vol. 37, no. 5. Wiley Online Library, 2018, pp. 161–172.

[32] S. Zhong, Z. Zhong, and J. Hua, "Surface reconstruction by parallel and unified particle-based resampling from point clouds," *Computer Aided Geometric Design*, vol. 71, pp. 43–62, 2019.

[33] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

[34] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4548–4557.

[35] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2760–2769.

[36] I. Lang, A. Manor, and S. Avidan, "Samplenet: differentiable point cloud sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7578–7588.

[37] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, "Modeling point clouds with self-attention and gumbel subset sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3323–3332.

[38] E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, and J. Luo, "Adaptive hierarchical down-sampling for point cloud classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 956–12 964.

[39] Y. Qian, J. Hou, Y. Zeng, Q. Zhang, S. Kwong, and Y. He, "Mops-net: A matrix optimization-driven network fortask-oriented 3d point cloud downsampling," *arXiv preprint arXiv:2005.00383*, 2020.

[40] X. Wang, Y. Xu, K. Xu, A. Tagliasacchi, B. Zhou, A. Mahdavi-Amiri, and H. Zhang, "Pie-net: Parametric inference of point cloud edges," *arXiv preprint arXiv:2007.04883*, 2020.

[41] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5589–5598.

[42] W. Wu, Y. Zhang, D. Wang, and Y. Lei, "Sk-net: Deep learning on point cloud via end-to-end discovery of spatial keypoints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6422–6429.

[43] D. Levin, "Mesh-independent surface interpolation," in *Geometric modeling for scientific visualization*. Springer, 2004, pp. 37–49.

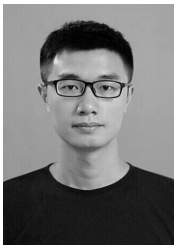
[44] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015.

[45] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, 2010.

[46] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg *et al.*, "The digital michelangelo project: 3d scanning of large statues," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 131–144.
[47] A. Bronstein, M. Bronstein, U. Castellani, B. Falcidieno, A. Fusiello, A. Godil, L. Guibas, I. Kokkinos, Z. Lian, M. Ovsjanikov *et al.*, "Shrec 2010: robust large-scale shape retrieval benchmark," *Proc. 3DOR*, vol. 5, no. 4, 2010.
[48] K. Lai, L. Bo, X. Ren, and D. Fox, "Rgb-d object recognition: Features, algorithms, and a large scale benchmark," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 167–192.
[49] M. Alexa and A. Adamson., "On normals and projection operators for surfaces defined by point sets," in *Eurographics Symposium on Point-Based Graphics*, 2004, pp. 149 – 155.
[50] D. Cohen-Steiner and F. Da, "A greedy delaunay-based surface reconstruction algorithm," *The visual computer*, vol. 20, no. 1, pp. 4–16, 2004.

**Chenlei Lv** received PhD degree in College of Information Science and Technology, Beijing Normal University (BNU). He is currently a Postdoctor in School of Computer Science and Engineering, Nanyang Technological University (NTU). His research interests include Computer Vision, 3D Biometrics, Computer Graphics, Discrete Differential Geometry and Conformal Geometric. He has published several papers in IEEE Transactions on Multimedia, Pattern Recognition, etc. He has also served as reviewer for Pattern Recognition and NEUROCOMPUTING. He is now mainly working on a point cloud processing and simplification project. His personal page can be reached via: https://aliexken.github.io/.

**Weisi Lin** (M'92–SM'98–F'16) received the Ph.D. degree from King's College London, U.K. He is currently a Professor with the School of Computer Science and Engineering, Nanyang Technological University. His research interests include image processing, perceptual signal modeling, video compression, and multimedia communication, in which he has published over 200 journal papers, over 230 conference papers, filed seven patents, and authored two books. He is a fellow of the IET and an Honorary Fellow of the Singapore Institute of Engineering Technologists. He was the Technical Program Chair of the IEEE ICME 2013, PCM 2012, QoMEX 2014, and the IEEE VCIP 2017. He has been an Invited/Panelist/Keynote/Tutorial Speaker at over 20 international conferences and was a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2016 to 2017 and the Asia–Pacific Signal and Information Processing Association (APSIPA) from 2012 to 2013. He has been an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE SIGNAL PROCESSING LETTERS.



**Baoquan Zhao** received his Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2017. He is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has served as a reviewer and technical program committee member of several journals and international conferences. He is a recipient of the Outstanding Reviewer Award of 2020 IEEE ICME. His research interests include point cloud processing and compression, visual information analysis, multimedia systems and applications, etc.