Self-Adaptive Point Cloud Simplification with Feature Preservation

Huang, L.H. [1*] and Jaw, J.J. [2]

[1] Master Student, Department of Civil Engineering, National Taiwan University, Taiwan

[2] Associate Professor, Department of Civil Engineering, National Taiwan University, Taiwan

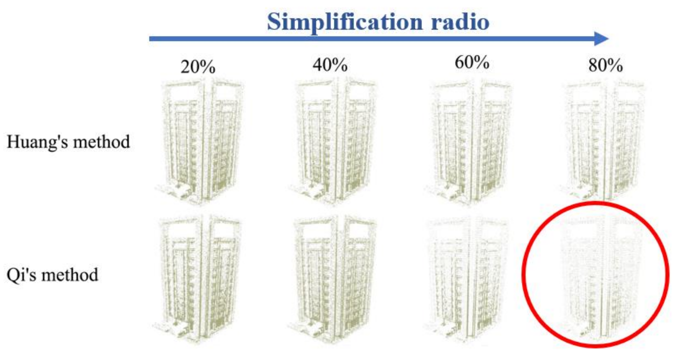
*anitahappy40127@gmail.com

Abstract **:** *With advancements in photogrammetry and computer vision, optical point clouds generated by stereo matching are widely used. However, processing large point cloud data consumes significant time and storage, necessitating data reduction while maintaining geometric accuracy. Existing simplification algorithms often rely on empirical rules and cannot adapt to regional characteristics. This study enhances a method for point cloud simplification using edge, feature, and non-feature points. The improvement is that the neighborhood size for each point is adaptively determined based on point cloud characteristics. First, the topological structure of the point cloud is established, and adaptive neighborhood size is determined using curvature features and entropy from Principal Component Analysis (PCA). The point cloud data is divided into sparse and regular areas, and different neighborhood calculation methods are applied to each area. A partitioning strategy simplifies the point cloud, with edge points extracted using normal vector angle differences and a region-growing segmentation method dividing the point cloud into feature and non-feature regions. In each feature region, points are traversed, and their importance is calculated by summing weighted differences in normal vectors, projection distances, spatial distances, and curvature differences with their neighborhoods. Each feature point's importance is compared to a threshold; if greater, the point is retained; if less, it is combined with the non-feature region as a non-feature point, and the number of non-feature points to retain is calculated by taking the ratio of local curvature to global curvature into consideration. Finally, edge, feature, and non-feature points are combined as the simplified point cloud. Preliminary experimental results indicate that this method effectively simplifies point clouds while preserving features, indeed resulting in light point clouds with quality geometric structure and content.*

Keywords: Feature preservation, Point cloud simplification, Principal component analysis, Region growing segmentation, Self-adaptive neighborhood

Introduction

With the advancement of technology, point clouds contain redundant data, which consumes time and storage space. Therefore, it has become necessary to reduce the number of points in the point cloud. However, excessive reduction can result in the loss of important features. As shown in Figure 1, the red circle illustrates that excessive point reduction leads to the inability to retain point cloud features. The main purpose of point cloud simplification is to reduce the amount of data while preserving feature points, such as edges, sharp angles, surface transitions, and high curvature areas (Chen & Yue, 2016). This study will focus on optical point clouds, and evaluate the importance of shape factors by quantifying geometric features to effectively preserve features while simplifying the number of points.



Source: (Chen et al., 2023)

Figure 1: The Results Under Different Simplification Rates.

Literature Review

In point cloud simplification, the process is mainly divided into mesh-based simplification and point-based simplification (Chen et al., 2023)

a. Mesh-Based Simplification

Mesh-based simplification methods involve reconstructing point cloud meshes, such as vertex merging and subdivision (Tang, 2007). Luebke (2001) applied mesh compression to simplify point clouds. Martin et al. (1997) uniformly divided the point cloud into grids, and replaced all points with the grid's centroid. Although this approach is simple, it results in the loss of important features. Wang et al. (2007) used an octree to retain the point closest to the centroid in leaf nodes. This method is fast but struggles to preserve features. Therefore, mesh construction is efficient for small-scale point clouds but resource-intensive for large-scale ones.

b. Point-Based Simplification

Point-based simplification methods simplify based on point features. They typically use parameters to identify key feature points such as curvature and normal angles (Chen et al., 2023). Due to the independent nature of point clouds, a topological structure must first be established to create relationships between points. This approach is more efficient as it avoids the need to reconstruct point cloud meshes.

c. Point Importance-Based Methods

The mesh-based simplification method incurs high computational cost, so point-based importance methods are more commonly used. These methods reduce point cloud data by retaining important points, with the key being determining the importance of each point. Point importance is typically evaluated based on geometric information such as normals, curvature, point spacing, and density. Dyn et al. (2008) used non-negative functions to remove unimportant points and updated the importance of neighboring points, achieving low error but slow processing speed. Weinmann et al. (2015) derived 3D features such as linearity, planarity, and curvature from local 3D neighborhoods. Ji et al. (2019) calculated point cloud importance using normal vector difference, projection distance, spatial distance, and curvature difference. Chen et al. (2023) extracted curvature and normal angle as features, while Song et al. (2009) quantified point information or redundancy based on geometric contributions and removed redundant points. Gan et al. (2023) proposed a simplification algorithm based on Principal Component Analysis (PCA), dividing the point cloud into feature and non-feature point clouds to obtain the final simplification result. Points located at sharp edges are also considered important; Huang et al. (2010) addressed the issue of boundary point loss by proposing a method to preserve boundary points. Gong et al. (2021) quantified the clustering of points within a neighborhood by calculating the summation of normal vectors of a point and its neighboring points; if the summation approaches zero, the point is considered a non-edge point. Wang et al. (2022) used region-growing segmentation to simplify point cloud partitions, while Gao et al. (2021) divided the point cloud into feature and non-feature points, simplifying non-feature points using the regional gravity center method.

d. Research gap

However, retaining only important points while discarding less important ones or regions can lead to the loss of geometric features in the point cloud. Therefore, completeness still requires further research. In summary, the methods discussed provide solutions for point cloud simplification but also have limitations. Mesh-based simplification involves high computational costs, whereas point-based importance methods preserve local information but require multiple iterations of removing unimportant points until the target number is reached. This approach increases the computational load and may lead to uneven point selection, which can result in gaps or excessive simplification.

Although there are many existing simplification algorithms, the simplification parameters for unordered point clouds often rely on empirical rules, making it difficult to adaptively adjust based on local features. This leads to the insufficient preservation of important information in regions with significant feature differences. Among current adaptive neighborhood adjustment methods, Demantké et al. (2017) proposed an entropy-based neighborhood selection method based on multi-scale analysis, while He et al. (2017) introduced a curvature-based adaptive neighborhood determination method. However, the former only uses radius search, which may fail to find points in sparse regions, and the latter adapts based on point cloud density without a fixed criterion. Therefore, combining both methods allows for adaptive neighborhood selection, and the importance threshold can be adjusted dynamically for each neighborhood. This study aims to implement an adaptive feature preservation strategy by dynamically adjusting the neighborhood size of each point, retaining geometric features while reducing computational costs.

Methodology

This study presents an improved method based on existing approaches for simplifying edge points, feature points, and non-feature points. The improvement lies in adapting the neighborhood size to the point cloud characteristics, rather than keeping it fixed, and using a partitioning strategy to enhance simplification efficiency. Unlike traditional methods, this study calculates point importance only within feature regions. It sets dynamic thresholds based on varying neighborhood sizes to preserve feature points and avoids calculations across the entire point cloud.

First, PCA is used to obtain curvature and entropy to establish adaptive neighborhoods, dividing the point cloud into scattered and regular regions, with different methods used to determine neighborhood size. Next, a partitioning strategy simplifies the point cloud by identifying edge points through normal vector distribution and segmenting the cloud into feature and non-feature regions using a region-growing method(Wang et al. ,2022).

Within each feature region, the method evaluates every point by calculating the normal vector difference, projection distance, spatial distance, and curvature difference relative to its neighborhood (Ji et al., 2019). The weighted sum of these factors determines point importance. If the importance exceeds a certain threshold, the point is retained; otherwise, it is merged with non-feature regions as a non-feature point. Non-feature points are simplified by their regional centroids. Finally, edge points, feature points, and the simplified non-feature points are combined to complete the point cloud simplification, as shown in Figure 2.

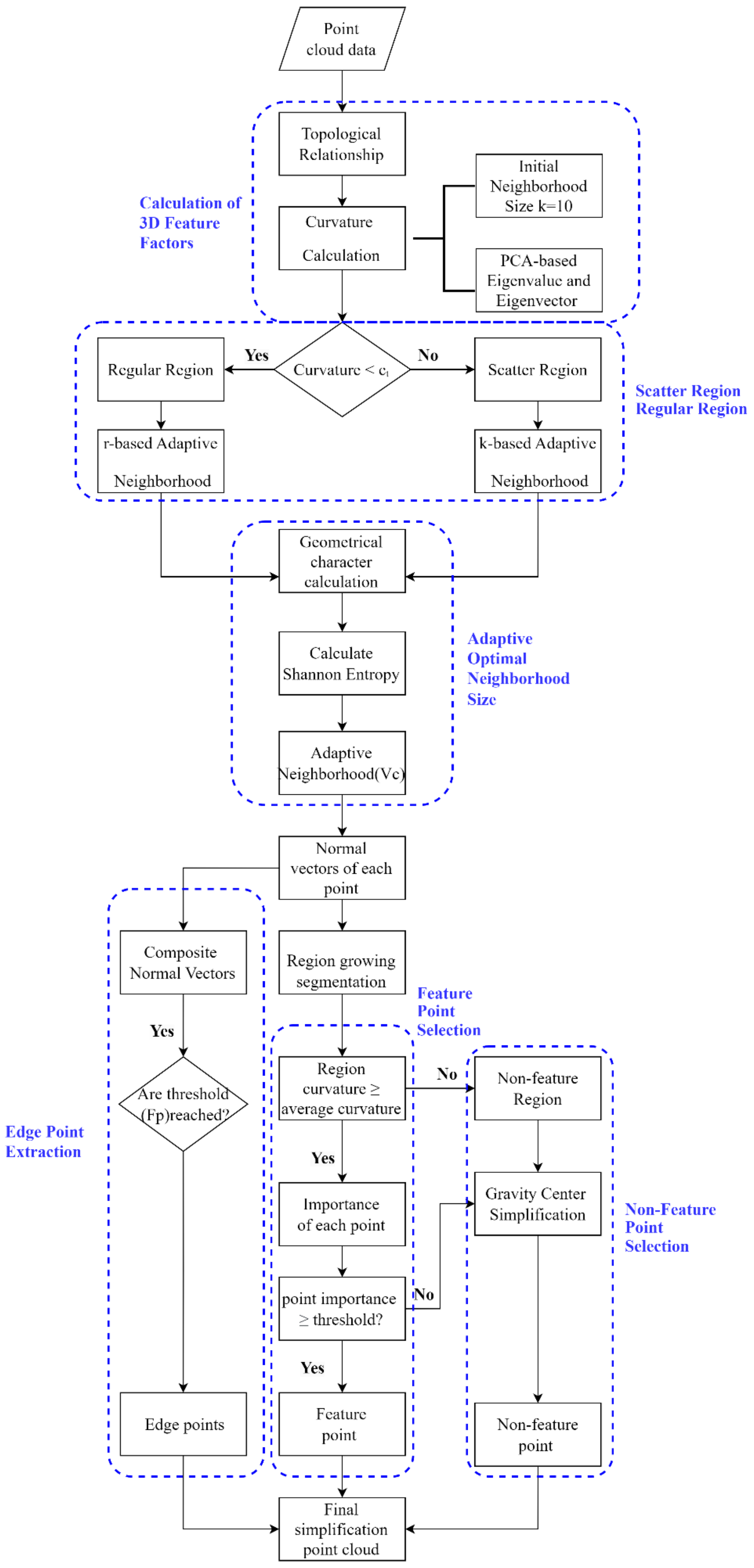

Point cloud data
Calculation of 3D Feature Factors
Topological Relationship
Curvature Calculation
Initial Neighborhood Size k=10
PCA-based Eigenvalue and Eigenvector
Regular Region
Yes
Curvature < c_t
No
Scatter Region
Scatter Region Regular Region
r-based Adaptive Neighborhood
k-based Adaptive Neighborhood
Geometrical character calculation
Calculate Shannon Entropy
Adaptive Neighborhood(Vc)
Adaptive Optimal Neighborhood Size
Normal vectors of each point
Composite Normal Vectors
Region growing segmentation
Feature Point Selection
Yes
Are threshold (Fp)reached?
Edge Point Extraction
Region curvature ≥ average curvature
No
Non-feature Region
Yes
Importance of each point
Gravity Center Simplification
Non-Feature Point Selection
point importance ≥ threshold?
No
Yes
Edge points
Feature point
Non-feature point
Final simplification point cloud

Figure 2: Flowchart of the Study.

a. Calculation of 3D Feature Factors:

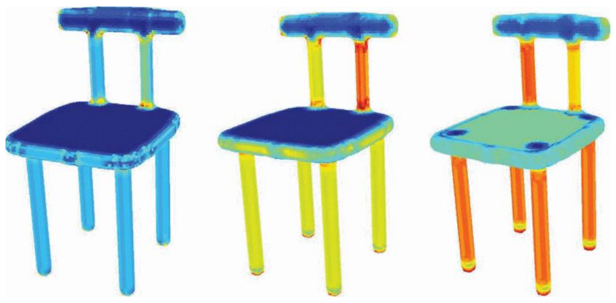
PCA calculates the eigenvalues λ_1 、λ_2 、λ_3, which are then normalized as shown in Equation (1), resulting in Table 1. Scattering, Planarity, and Linearity are used to classify corner, planar , and linear features. Curvature indicates the degree of flatness and represents the sharpness of the sample point. Higher eigenentropy signifies rapid changes and highlights the importance of the feature.

$$e_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3} \tag{1}$$

Table 1: Classification of Feature Factors. (Weinmann et al., 2015)

Feature Calculation Formula	3D Variables
$S_\lambda = \frac{e_3}{e_1}$	Scattering
$P_\lambda = \frac{(e_2 - e_3)}{e_1}$	Planarity
$L_\lambda = \frac{(e_1 - e_2)}{e_1}$	Linearity
$E_\lambda = -\sum_{i=1}^{3} e_i \ln(e_i)$	Eigenentropy
$C_\lambda = \frac{e_3}{e_1 + e_2 + e_3}$	Curvature

Curvature estimates the degree to which neighboring points deviate from the tangent plane. A ratio of 0 indicates that the neighboring points lie entirely on the plane. Pauly et al. (2002) noted that surface variations are considered curvature, but the value depends on the neighborhood size. Figure 3 shows the impact of different neighborhood sizes on the curvature estimation of a chair dataset, highlighting the importance of selecting an appropriate neighborhood size for accurate curvature estimation.



Source: (Benhabiles et al., 2013)

Figure 3: From Left to Right: Curvature Estimated With Neighborhood Sizes of 20, 50, and 100.

b. Adaptive Optimal Neighborhood Size

Due to the significant variation in point cloud structure and density, a fixed neighborhood size is insufficient, and dynamic adjustment of the number of neighbors for each point is required (Weinmann et al., 2015). He et al. (2017) proposed a curvature-based adaptive neighborhood method that divides the point cloud into scattered and regular regions, applying k-nearest neighbors and r-nearest neighbors, respectively. In low-density or irregularly distributed areas, k-nearest neighbors ensure sufficient neighbors are found, while in high-density areas, r-nearest neighbors are more efficient, avoiding the linear behavior issues of the k-nearest method.

First, the curvature of each point is calculated, with the neighborhood size initially set to k=10. Then, the k-means algorithm is applied to divide the point cloud into two clusters, with the initial cluster centers being the minimum and maximum curvature values in the point cloud. Assuming the centers of the two clusters are cur_1 and cur_2 $(cur_2 > cur_1)$, the curvature threshold c_t can be expressed as shown in Equation (2).

$$c_t = cur_1 + (cur_2 - cur_1) \times \frac{numCluster_2}{numPoints} \tag{2}$$

Here, cur_1 and cur_2 represent the curvature values of the two cluster centers, $numCluster_2$ is the number of points in the higher curvature cluster, and $numPoints$ is the total number of points.

For scattered regions, the k-nearest neighborhood (v_k) is used, while the r-nearest neighbors neighborhood (v_r) is applied for regular regions. After setting the ranges for v_k and v_r, Shannon entropy is used to determine the optimal neighborhood. Based on the multi-scale analysis method of Demantké et al. (2017), the entropy within a radius is calculated, and the radius with the lowest entropy is chosen as the optimal neighborhood. In the range $[r_{min}, r_{max}]$, L_λ 、 P_λ and S_λ are calculated for each point, and Shannon entropy is compared to determine the optimal radius for the point, as shown in Equation (3).

$$E_f\left(V_r{}^P\right) = -L_\lambda \ln(L_\lambda) - P_\lambda\, ln(P_\lambda) - S_\lambda\, ln(S_\lambda) \tag{3}$$

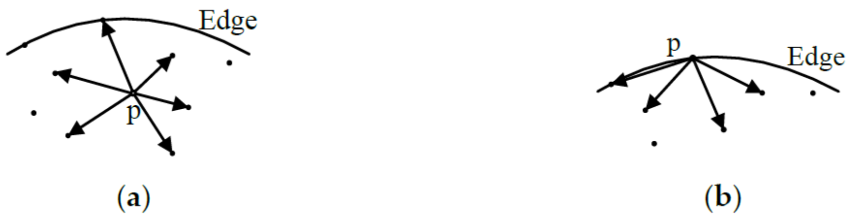
Equation (4) calculates the entropyE_f of a point P within a neighborhood of radius r. The optimal radius $r^*{}_{E_f}$ is the radius that minimizes $E_f(V_r{}^P)$, as shown in Equation (4):

$$r^*{}_{E_f} = \arg \min_{r \in [r_{min}, r_{max}]} E_f\left(V_r{}^P\right) \tag{4}$$

Similarly, this study suggests that Shannon entropy can be used to select the optimal k value. By applying the method separately to scattered and regular regions, the k value and radius with the lowest entropy are chosen and then combined to form the adaptive neighborhood (v_c).

c. Edge Points

Based on preliminary experiments, this study chose to use the method proposed by Gong et al. (2021) , non-edge points have a uniformly distributed neighborhood, while edge points have their neighborhood points clustered in a specific direction. The edge coefficient F p measures the degree of distribution clustering within the neighborhood of point p by summing the normal vectors of the point and its neighbors. If the resultant vector is close to zero, the point is classified as a non-edge point (see Figure 4(a)); if the resultant vector significantly deviates from zero, the point is classified as an edge point (see Figure 4(b)).



Source: (Gao et al., 2021)

Figure 4: Distribution of Edge and Non-Edge Points

$$F_P = \frac{1}{D}\left|\sum_{i=1}^{D} \frac{\overrightarrow{pp_i}}{|\overrightarrow{pp_i}|}\right| \quad (5)$$

Where：

F_P is the edge coefficient, measuring the clustering degree of points around p.

D is the number of neighboring points.

$\overrightarrow{pp_i}$ is the vector from point p to its neighbor p_i.

Equation (5) calculates the average magnitude of the unit vectors from point p to each neighboring point. A larger F_P value indicates uneven distribution of neighbors, suggesting that point p may be an edge point.

d. Region growing segmentation

Even with the region-growing segmentation method by Wang et al. (2022), the point cloud can be adaptively divided into feature and non-feature regions based on curvature, improving accuracy with a smaller computation range. If the average regional curvature is greater than or equal to the global average, it is classified as a feature region, retaining feature points P_f with importance exceeding a threshold; otherwise, it is classified as a non-feature region. Here, C_{th} and C_{th} are two parameters determining regional adaptiveness in the region-growing segmentation: C_{th} is the curvature threshold, which decides the number of seed points, while θ_{th} is the angle threshold, which determines the number of points within the region. These parameters together determine the region size. When C_{th} and θ_{th} are set too low, the region segmentation becomes overly detailed, resulting in more points being defined as feature points, which increases computational load. Conversely, when C_{th} and θ_{th} are set too high, the region segmentation becomes too coarse, leading to the omission of many feature points. The settings of these parameters will be analyzed in subsequent experiments.

e. Calculation of Point Cloud Importance

Linsen (2001) defined point importance as the sum of neighborhood distance, non-planarity, and normal vector variation. Alexa et al. (2001) used surface projection distance as a measure, while Chen et al. (2023) introduced parameters such as curvature, normal vector angle, and distance. Song et al. (2009) argued that points cannot substitute each other when the neighborhood distance is too large. Therefore, this study adopts normal vector difference, projection distance, spatial distance, and curvature difference to describe point importance (Ji et al., 2019).

$$\sum_{j=1}^{k}\left[\alpha \times \left(1 - {n_p}^T n_j\right) + \beta \times \left|{n_p}^T\left(p - p_j\right)\right| + \gamma \times \left\|p - p_j\right\| + \delta \times \left|cu_p - c_j\right|\right] \tag{6}$$

Where $\alpha, \beta, \gamma, \delta$ ($\alpha > 0, \beta > 0, \gamma > 0, \delta > 0$ and $\alpha + \beta + \gamma + \delta = 1$) are weighting factors. p is the sample point , $p_j (1 \leq j \leq k)$ are the neighbors of p. n_p is the normal vector of p, $n_j (1 \leq j \leq k)$ are the normal vectors of the neighboring points p_j, cu_p is the curvature of p, and $c_j (1 \leq j \leq k)$ are the curvatures of p_j.

$\left(1 - {n_p}^T n_j\right)$: A larger normal vector difference indicates the surface around the point is more convex.

$\left|{n_p}^T\left(p - p_j\right)\right|$: The projection distance reflects the point's concavity or convexity.

$\left\|p - p_j\right\|$: Euclidean distance, with larger values suggesting the point is in a sharp or sparse region, and should be retained to maintain completeness.

$\left|cu_p - c_j\right|$: Curvature difference, which indicates whether the point is in a sharp region.

f. Threshold Determination for Feature Point Selection

When the importance of a point is greater than or equal to the threshold, the point is retained as a feature point. According to the definition by Yang et al. (2023), it is set as:

$$\frac{\varepsilon \times \sum_{j=1}^{N}\left[\left(1 - {n_p}^T n_j\right) + \left|{n_p}^T\left(p - p_j\right)\right| + \left\|p - p_j\right\| + \left|cu_p - c_j\right|\right]}{T} \tag{7}$$

Where ε represents the control coefficient for the number of feature points, and T denotes the number of neighboring points in the point cloud.

g. Non-Feature Point Simplification

After extracting feature points, flat areas may develop holes, so non-feature points need to be simplified. This study employs the region centroid-based simplification method (Gao et al., 2021), which divides non-feature points into small cubes, calculates the distance between the centroid and the internal points, and finally merges the edge points, feature points, and non-feature points to complete point cloud simplification.

$$l_x = x_{max} - x_{min}\ ;\ l_y = y_{max} - y_{min}\ ;\ l_z = z_{max} - z_{min} \tag{8}$$

$$d_x = \frac{l_x}{u}\ ;\ d_y = \frac{l_y}{u}\ ;\ d_z = \frac{l_z}{u} \tag{9}$$

Based on the coordinates of the point cloud data, calculate the maximum and minimum values for the X, Y, and Z axes (x_{max}, y_{max}, z_{max}, x_{min}, y_{min}, z_{min}).Use formula (8) to calculate the edge lengths of the cubes(l_x, l_y, l_z). Set the edge length of each cube to u and use formula (9) to calculate the size of the small cubes. The choice of u depends on the object's size. If u is too large, details may be lost; if u is too small, the simplification effect may be inadequate (Wang et al., 2022). In this study, u is used to control the simplification rate.

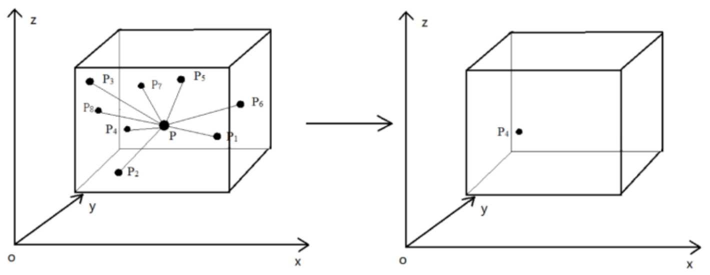
Assuming a cube contains N points p_1 、p_2 、$p_3 \dots . p_N$, with coordinates (p_{xi}, p_{yi}, p_{zi}) for $i = 1,2, \dots N$, the average point p is defined as:

$$p_x = \frac{\sum_{i=1}^{N} p_{xi}}{N}\ p_y = \frac{\sum_{i=1}^{N} p_{yi}}{N}\ p_z = \frac{\sum_{i=1}^{N} p_{zi}}{N} \tag{10}$$

Then, calculate the distance d_i between each point $p_i(p_{xi}, p_{yi}, p_{zi})$ in the small cube and the average point p. The distance d_i is given by:

$$d_i = \sqrt{(p_{xi} - p_x)^2 + (p_{yi} - p_y)^2 + (p_{zi} - p_z)^2} \tag{11}$$

Retain the point closest to the centroid within the small cube, while deleting the other points. The simplification process is illustrated in Figure 6.



Source: (Gao et al., 2021)

Figure 5: Illustration of Region Centroid-Based Simplification (Gao et al., 2021).

h. Quantitative Standards for Point Cloud Simplification

According to Cignoni et al. (1998), the error of simplified point clouds can be measured using maximum geometric error, average geometric error, and RMSE. Let S represent the original point cloud surface, and S' represent the simplified surface. The geometric error $d(p,S')$ is defined as the Euclidean distance between a point p and its projection on S'. The maximum and average geometric errors, as well as RMSE, are defined as follows:

$$\Delta_{max}(S,S') = \underbrace{max}_{q\in S} |d(p,S')| \tag{12}$$

$$\Delta_{ave}(S,S') = \frac{1}{\|S\|}\sum\nolimits_{q\in S} d(p,S') \tag{13}$$

$$\sigma = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(X_i)^2} \tag{14}$$

X_i is the i-th error value and n is the total number of error values.

The formulas for the simplification rate and the change ratio of volume are given as follows:

$$Simplification\ Rate = \frac{Original\ Points - Simplified\ Points}{Original\ Points} \times 100\% \tag{15}$$

$$\boldsymbol{Change\ Ratio\ \ of\ Volume = \frac{Volume\ before - Volume\ after}{Volume\ before} \times 100\%} \tag{16}$$

Results and Discussion

This study aims to achieve effective feature preservation within an adaptive neighborhood range while reducing the number of points in the point cloud. To validate the proposed method, simulation experiments were conducted using datasets to assess its performance. Additionally, actual point cloud data were used to evaluate the method's effectiveness in real-world scenarios.

a. Simulation experiment

To validate the accuracy of the proposed method, the Bunny point cloud dataset from the Stanford 3D Point Cloud Database was used to test its feasibility. The original point cloud consists of 35,947 points, as shown in Figure 6. All experiments were conducted on a system running Windows 11 Pro (version 23H2), equipped with an Intel(R) Core(TM) i7-12700F processor (2.10 GHz) and 32 GB of memory. The code was developed in the MATLAB 2024a environment.

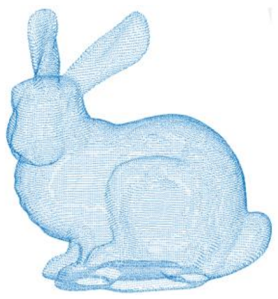


Figure 6: Stanford bunny Original Point Cloud.

b. Scatter Region and Regular Region

The point cloud data is divided into scatter regions (curvature greater than or equal to c_t and regular regions (curvature less than c_t). In Figure 7, which shows the Stanford Bunny dataset, the scatter regions, marked in red, are predominantly located where curvature changes are more pronounced. These areas, such as the ears, neck, and legs, exhibit dense and highly variable point distributions. Conversely, the regular regions, marked in blue, are mostly found on the body, with a relatively flat and uniform point distribution. This indicates that curvature provides a consistent criterion for evaluating the geometric properties of points in the point cloud, serving as the basis for adaptive neighborhood selection.

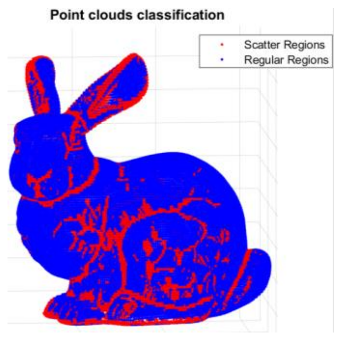


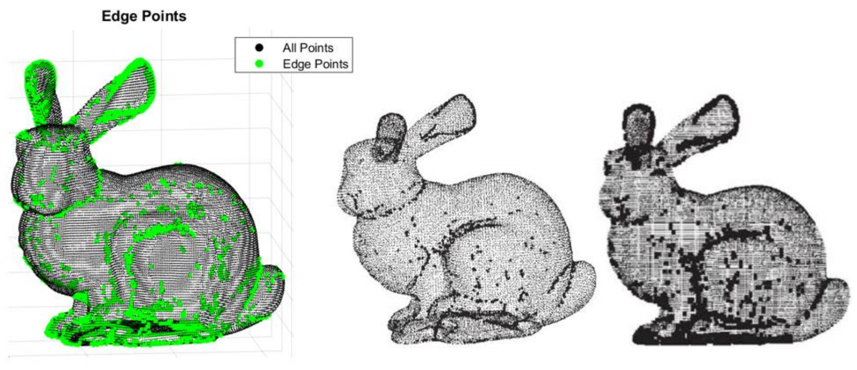
Figure 7: Scatter Region and Regular Region on Bunny.

c. Setting Parameters for the Experiments—v_k and v_r

The parameters to be set in this study include v_k andv_r, the parameters C_{th} and θ_{th} for the region growing segmentation, the four importance parameters $\alpha, \beta, \gamma, \delta$, and the feature point threshold ε. The range for the adaptive neighborhood v_k varies between $k_{min} = 10$ and $k_{min} = 30$, with an interval of 1. The neighborhood v_r, based on point cloud density and average spacing, is chosen to be within the range [0.001, 0.004] after several trials, with a spacing of 16 according to Demantké et al. (2017). After assigning points to scatter or regular regions, the Shannon entropy is used to select the k value or radius with the smallest entropy as the optimal neighborhood.

d. Edge Point Extraction

Figure 9(a) displays the edge point extraction results from this study. Compared to Figure 8 (b) from Benhabiles et al. (2013) and Figure 9(c) from Song & Feng (2009), all methods detect sharp regions such as the ear boundaries and tail contours.



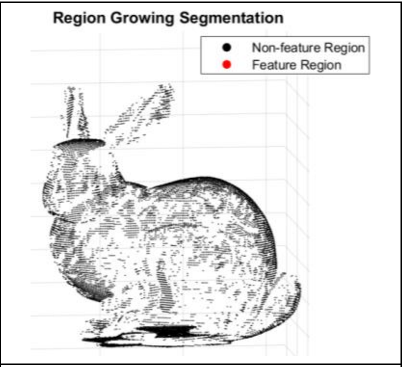
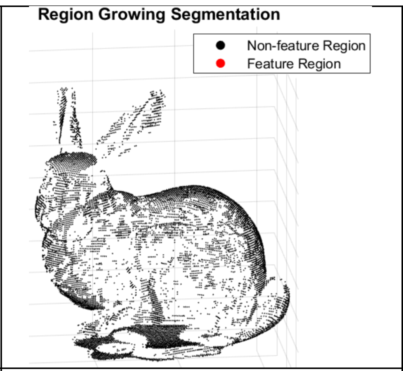
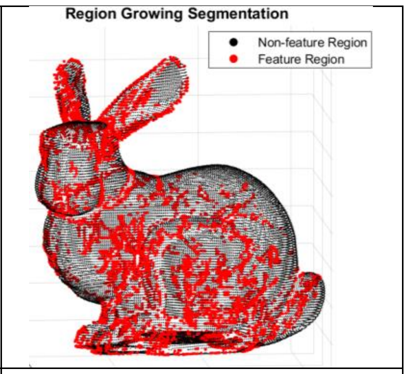
(a) (b) (c)

Source: (Benhabiles et al., 2013); (Song & Feng, 2009)

Figure 8: Edge Point. (a) Edge point extraction results using the method from this study. (b) Results from Benhabiles et al. (2013). (c) Results from Song & Feng (2009).

e. Setting Parameters for the Experiments—C_{th} 、 θ_{th} 、 $\alpha, \beta, \gamma, \delta$ 、 ε

C_{th} and θ_{th} are two key parameters in the region-growing segmentation process. C_{th} represents the curvature threshold, while θ_{th} is the angle threshold. Setting C_{th} and θ_{th} too small results in overly detailed segmentation, increasing computational load, while setting them too large leads to coarse segmentation, potentially missing feature points. This study experimentally verifies the effects of settingC_{th} and θ_{th}, as shown in Figure 9.

 Region Growing Segmentation ● Non-feature Region ● Feature Region	 Region Growing Segmentation ● Non-feature Region ● Feature Region	 Region Growing Segmentation ● Non-feature Region ● Feature Region
(a)	(b)	(c)

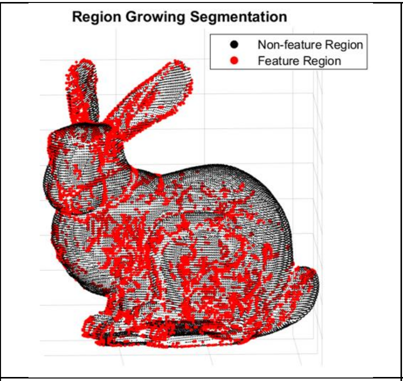
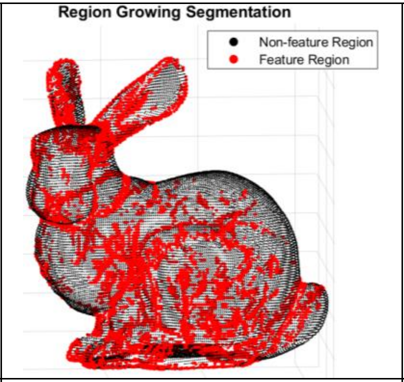
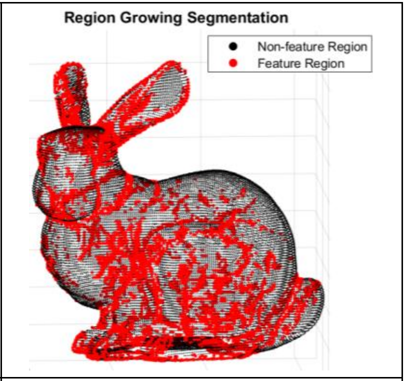
		
(d)	(e)	(f)

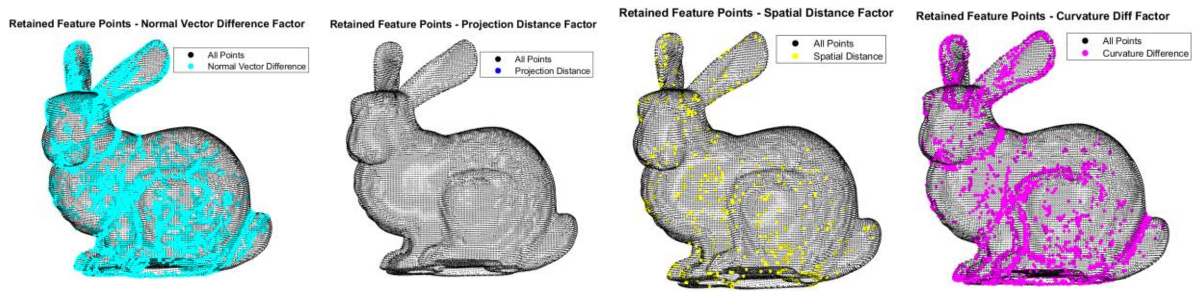
Figure 9: The feature points of differentC_{th} and θ_{th} on Bunny simplified dataset. (a)$C_{th} = 0.001, \theta_{th} = 5$; (b) $C_{th} = 0.001, \theta_{th} = 30$; (c) $C_{th} = 0.01, \theta_{th} = 5$; (d) $C_{th} = 0.01, \theta_{th} = 30$; (e) $C_{th} = 0.1, \theta_{th} = 5$; (f). $C_{th} = 0.1, \theta_{th} = 30$.

Additionally, the number of retained feature points is analyzed, as shown in Table 2.

Table 2: The different results of different C_{th} and θ_{th}.

Parameters	**Feature Points**
$C_{th} = 0.001, \theta_{th} = 5$	0
$C_{th} = 0.001, \theta_{th} = 30$	0
$C_{th} = 0.01, \theta_{th} = 5$	5207
$C_{th} = 0.01, \theta_{th} = 30$	5199
$C_{th} = 0.1, \theta_{th} = 5$	6849
$C_{th} = 0.1, \theta_{th} = 30$	6841

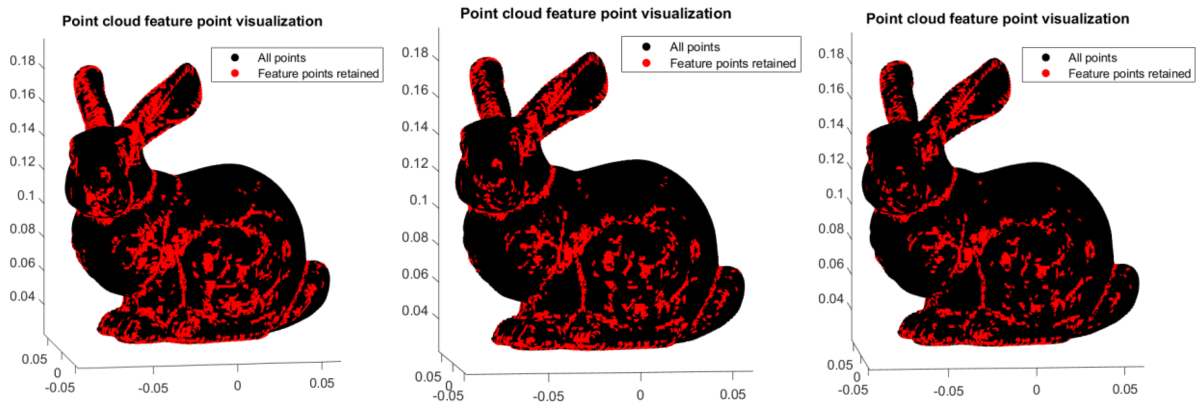
The experimental results indicate that when the curvature threshold C_{th} is set too low (e.g., $C_{th} = 0.001$), no feature points are detected, regardless of the angle threshold θ_{th}. As C_{th} increases, the number of feature points grows. While a larger θ_{th} slightly reduces the number of feature points, its impact is less significant than that of the curvature threshold. Therefore, setting $C_{th} = 0.1$ and $\theta_{th} = 5$ effectively preserves key features such as the neck, feet, and ears. To evaluate the influence of the weight parameters for feature factors, the initial setting is $\varepsilon = 1$. The analysis of each parameter set to 1 is detailed below.


Retained Feature Points - Normal Vector Difference Factor
● All Points
● Normal Vector Difference
Retained Feature Points - Projection Distance Factor
● All Points
● Projection Distance
Retained Feature Points - Spatial Distance Factor
● All Points
● Spatial Distance
Retained Feature Points - Curvature Diff Factor
● All Points
● Curvature Difference

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

Figure 10: Analysis of Importance Weighting Factors. (a) $\alpha = 1\ \beta = 0\ \gamma = 0\ \delta = 0$; (b) $\alpha = 0\ \beta = 1\ \gamma = 0\ \delta = 0$; (c) $\alpha = 0\ \beta = 0\ \gamma = 1\ \delta = 0$; (d) $\alpha = 0\ \beta = 0\ \gamma = 0\ \delta = 1$.

As shown in Figure 10, the normal vector difference concentrates feature points in areas with significant geometric changes, indicating its ability to effectively capture edges. The projection distance did not retain feature points and performed poorly, suggesting that an extremely small ε value might be required for feature points to appear. Although spatial distance retained fewer feature points, it was still able to identify points in edge regions of the dataset. Curvature difference preserved a large number of feature points, especially in areas with geometric changes (e.g., ears, feet, nose), demonstrating strong performance in capturing details. Therefore, curvature and normal vector difference should be prioritized, with spatial distance as a complement. The final weights were set to $\alpha = 0.4$ $\beta = 0.1$ $\gamma = 0.2$ $\delta = 0.3$, and the ε value was determined accordingly.



(a) (b) (c)

Figure 11: Comparison of the Feature Point Control Coefficient ε. (a) $\varepsilon = 1$ (b) $\varepsilon = 2$ (c) $\varepsilon = 3$.

Table 3: Volume Variation Due to Changes in ε Value.

The Value of ε	**Original Feature Region Volume**	**Feature Region Volume After Removing Unimportant Points**	**Change Ratio of Volume(%)**
1	0.0012065	0.0012043	0.18
2	0.0012065	0.0012	0.54
3	0.0012065	0.0011927	1.14

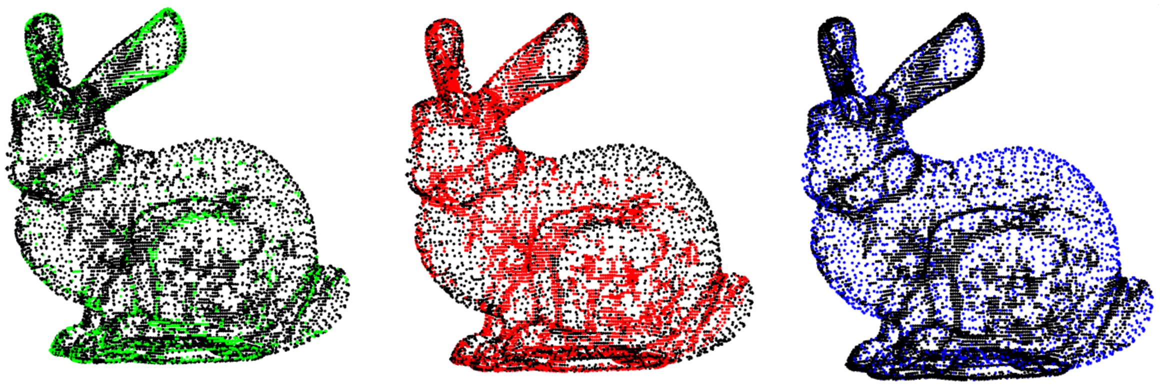
Based on Figure 11 and Tables 3, when $\varepsilon = 1$, most feature points are preserved, with minimal change in the volume of feature regions. At $\varepsilon = 2$, the number of feature points decreases, and the volume slightly reduces, achieving a balance between feature point preservation and simplification. At $\varepsilon = 3$, the volume significantly decreases, retaining only the most prominent feature points while removing more details.

Next, we analyze the effect of different ε values on feature preservation. As previously mentioned, u controls the simplification rate. To determine the impact of ε, u is set uniformly to 0.005. Figures 12 (a), (b), and (c) show edge points, feature points, and non-feature points for $\varepsilon = 1$, with a simplification rate of 67.92%. Figures 12 (d), (e), and (f) correspond to $\varepsilon = 2$, with a simplification rate of 72.15%. Figures 12 (g), (h), and (i) display the results for $\varepsilon = 3$, with a simplification rate of 74.60%. Figure 13 (a) shows the final point cloud simplification results from different perspectives for $\varepsilon = 1$. Figure 13 (b) corresponds to $\varepsilon = 2$, and Figure 13 (c) corresponds to $\varepsilon = 3$.

At $\varepsilon = 1$, features such as the ear edges, depressions, and thigh undulations are well-preserved. However, too many points are retained, including those in non-feature areas and even in smoother regions like the thigh undulations and around the tail. This results in weaker simplification. Thus, this ε value may not achieve effective point cloud simplification. Although feature preservation is good, the simplification is insufficient, and feature and non-feature points are not effectively distinguished.

At $\varepsilon = 2$, the details of the ear edges are better preserved. Depressions such as the rabbit's neck, feet, and thigh transitions maintain a relatively complete shape, and the details around the tail are also moderately preserved without excessive simplification. This indicates that $\varepsilon = 2$ provides a good balance between preserving important geometric features and achieving effective point cloud simplification.

At $\varepsilon = 3$, too many points are removed, resulting in the loss of important geometric features. The ear edges begin to blur, and details in depressions such as the neck transitions, feet, and tail are also lost. Although the thigh undulations are still present, the details are not as clear as with $\varepsilon = 2$.



(a) (b) (c)

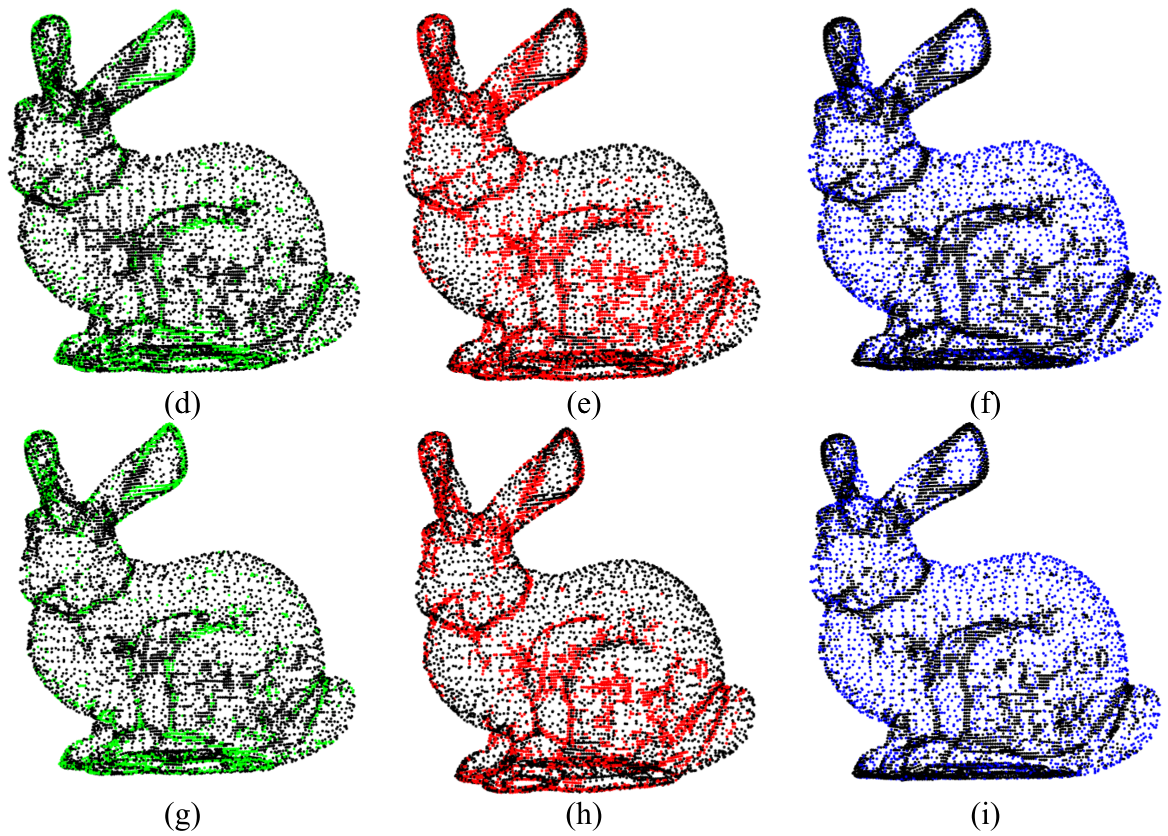
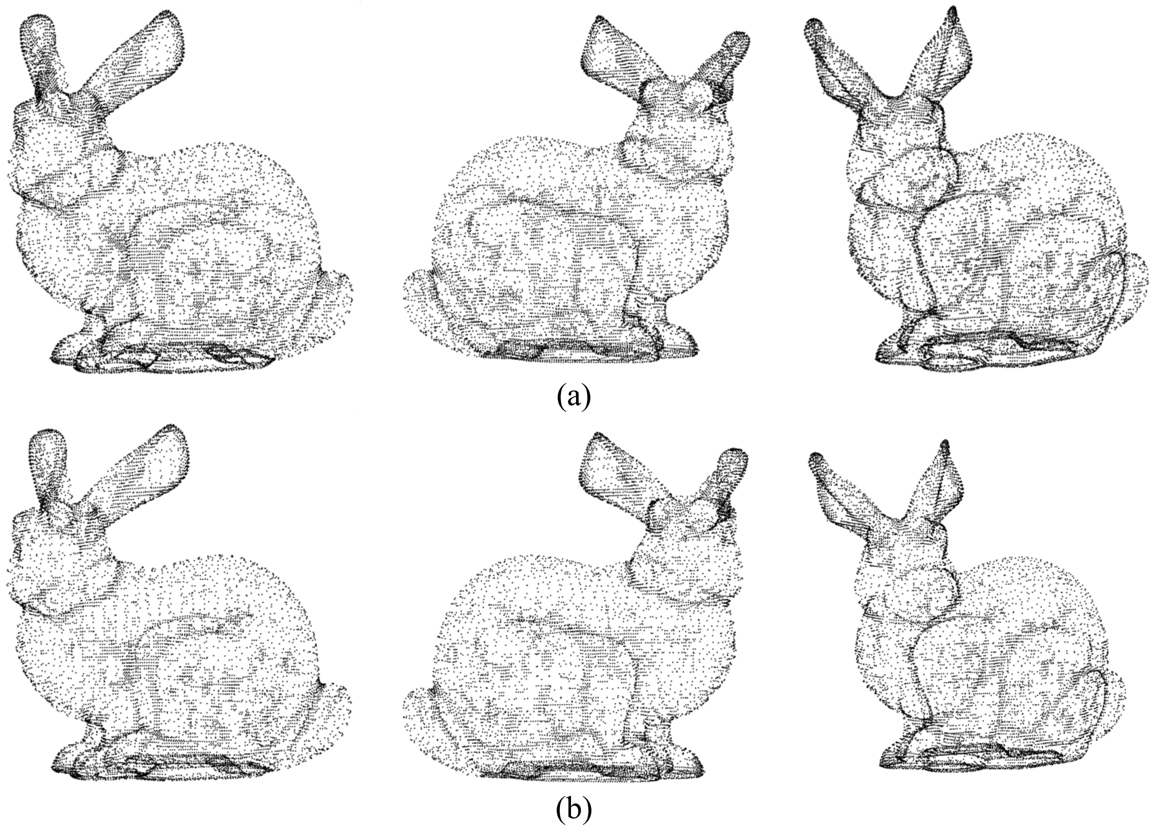


Figure 12: Simplified Point Cloud with Edge, Feature, and Non-Feature Points.



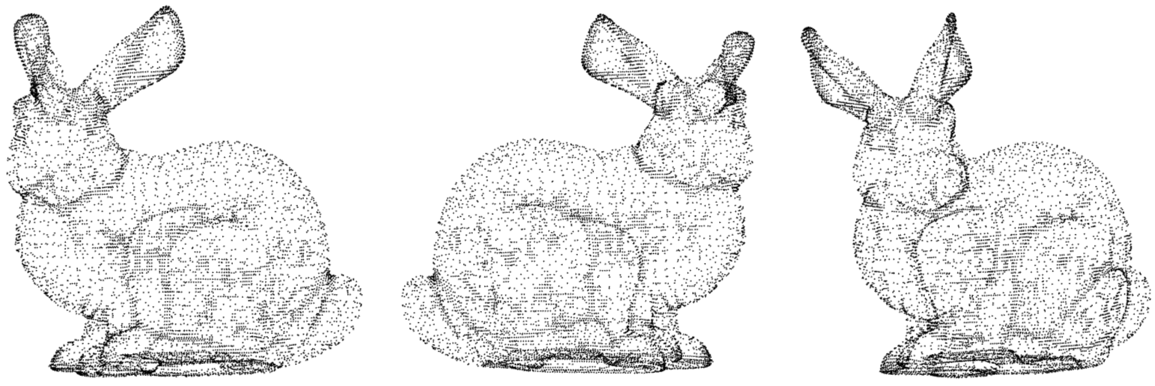

(c)

Figure 13: (a) Final Result for $\varepsilon = 1$, (b) Final Result for $\varepsilon = 2$, (c) Final Result for $\varepsilon = 3$.

Table 4: Error Comparison of Different ε Values in Simplified Point Cloud.

The Value of ε	Number of Points	Simplification Rate (%)	Maximum Geometric Error	Average Geometric Error	RMSE	Change Ratio of Volume(%)
1	11531	67.92	0.003847	0.001105	0.001418	0.18
2	10013	72.15	0.003847	0.001192	0.001483	0.54
3	9131	74.60	0.003847	0.001245	0.001523	1.14

Table 4 shows that as the ε value increases, the number of points in the point cloud decreases, while the maximum geometric error remains stable. However, the average geometric error and RMSE slightly increase, indicating a decrease in accuracy. Thus, a larger ε value results in better simplification but more detail loss, while a smaller ε value preserves more details but has limited effect on reducing error. Therefore, considering the balance between simplification accuracy and feature preservation, $\varepsilon = 2$ is the most suitable.

In Figure 13(b), the selection of parameters such as normal vector difference, projection distance, Euclidean distance, and curvature difference proves to be crucial for feature preservation. Normal vector difference effectively captures the geometric details of the rabbit's ears and feet. Projection distance helps maintain the undulations and transitions of the thighs and ears. Euclidean distance effectively preserves the contours at edge points like the tips of the ears and tail, while curvature difference protects sharp transitions. These four parameters work together, and with an appropriate ε setting, they achieve both feature detail preservation and point cloud simplification.

f. Comparison of Simplification Effects with Other Methods

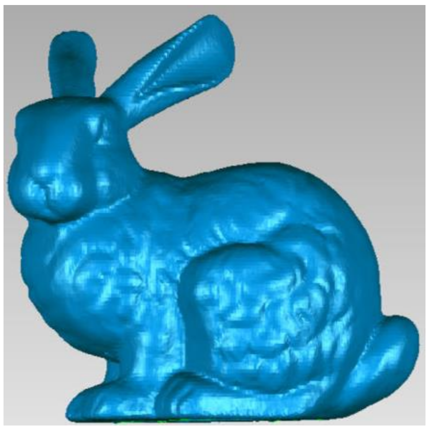
This study compares the simplification effects on the Bunny dataset of several recently developed point cloud simplification methods: Delaunay neighborhood-based method (Gong et al., 2021), Partition-based method (Wang et al., 2022), Grid-based method (Zhou et al., 2021), and Boundary-Preserving method (Chen et al., 2023), aiming for similar simplification rates. The original point cloud, shown in Figure 14 comes from Ji et al. (2019) using Geomagic Studio. The point cloud simplification results are analyzed against this standard, as detailed in Tables 5 and 6.

The Delaunay neighborhood-based method (Gong et al., 2021) maintains the overall shape well but suffers from significant loss of detailed features. Details such as the edges of the rabbit's ears, the subtle depressions on the thighs, and the tail's details are not effectively preserved, showing poor feature representation. In contrast, this study, at a comparable simplification rate, not only maintains the overall shape effectively but also preserves details such as the ears, thighs, and tail more comprehensively. Although the maximum error is higher than in this study, the average error and RMSE are lower.

The Grid-based method (Zhou et al., 2021) produces somewhat blurred point cloud results, especially with almost indiscernible contours of the ears and details of the feet. This study preserves better features at a similar simplification rate, with clear contours of the ears and more pronounced undulations of the tail and thighs, demonstrating superior feature preservation. This method lacks values for maximum and average errors, but its RMSE is significantly higher than that of this study.

Compared to the Partition-based method (Wang et al., 2022), this study shows higher errors, indicating room for improvement in global error control. However, in terms of feature preservation, this study is superior in maintaining edge details of the ears and feet, particularly the depressions inside the ears. For the neck and thighs, this study better preserves undulations and features, making the overall shape retention of the rabbit more precise.

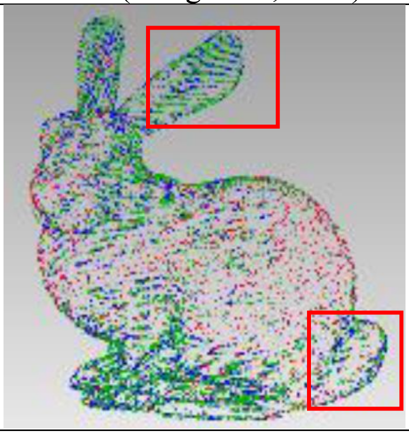
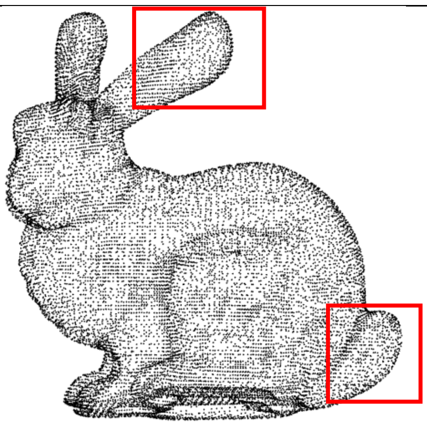
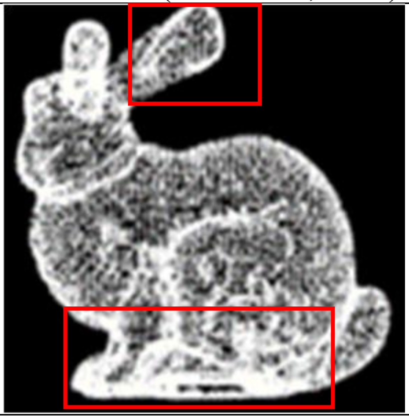
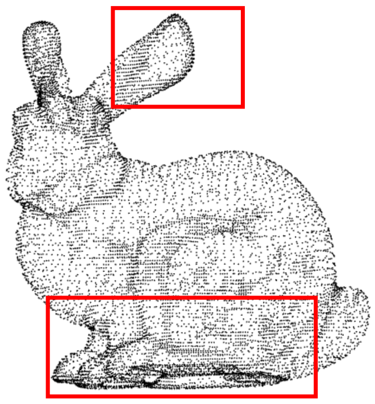
The Boundary-Preserving method (Chen et al., 2023) demonstrates strong feature preservation capabilities at an 80% simplification rate, especially in preserving details such as the edges of the rabbit's ears and the tail. In comparison, this study at the same simplification rate retains fewer non-feature points, resulting in less representation of the rabbit's back. Consequently, this study has higher maximum and average errors compared to this method (which lacks RMSE values for comparison). Nonetheless, this study maintains detailed features in the ear depressions and accurately preserves the shapes of the thighs and feet.



Source: (Ji et al., 2019)

Figure 14: Stanford Bunny dataset.

Table 5: Comparison of Features with Other Methods.

Simplification Rate (%) : Other Method/The Proposed	**Method**	
	Delaunay neighborhood-based(Gong et al., 2021)	The Proposed
53.00/53.02		
	Grid-based (Zhou et al., 2021)	The Proposed
66.5/66.84		
	Partition-based (Wang et al., 2022)	The Proposed

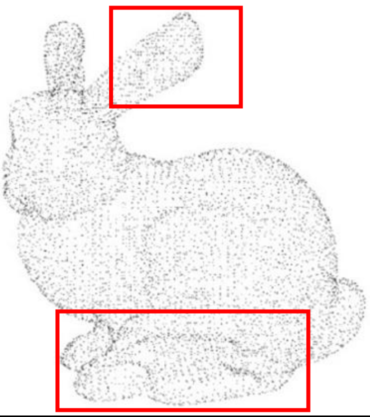
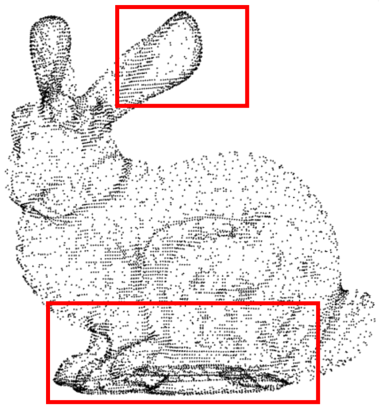
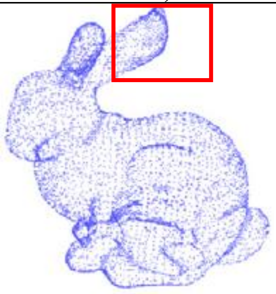
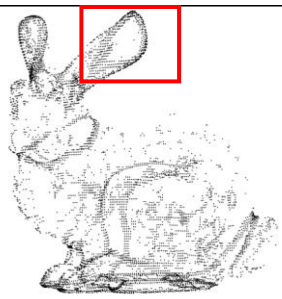
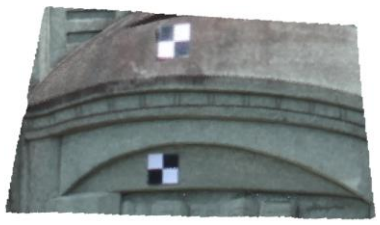
75/75.04		
	Boundary-Preserving(Chen et al., 2023)	The Proposed
80/80		

Table 6: Error Comparison with Other Methods.

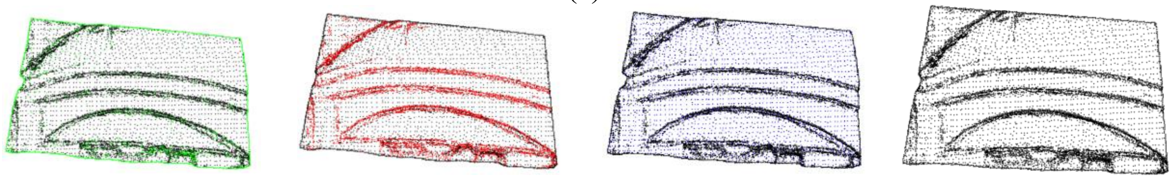
Method	**Simplification Rate (%)**	**Maximum Error**	**Average Error**	**RMSE**	**Change Ratio of Volume for the Proposed Method(%)**
Delaunay neighborhood-based(Gong et al., 2021)/The Proposed	53.00/53.02	0.021136 /0.002358	0.000204 /0.000622	0.000596 /0.000877	0.07
Grid-based Reduction(Zhou et al., 2021) /The Proposed	66.5/66.84			0.0256 /0.001203	0.15
Method(Wang et al., 2022) /The Proposed	75/75.04	0.002868 /0.004855	0.000066 /0.001426	0.000136 /0.001769	0.27
Four-Feature Boundary-Preserving(Chen et al., 2023)	80/80	0.0042 /0.018812	0.0011 /0.003181		1.16

g. Real Point Cloud Test

To further validate the robustness of the method, a 49,393-point sculpture point cloud (Yang & Jaw, 2023) from the Department of Civil Engineering, National Taiwan University, was selected for testing. This sculpture features an arched structure and geometric markings with rich details. Table 9 shows the extracted edge points and the regions classified into feature and non-feature areas through region growing segmentation in the real point cloud. The range for v_k is set between [10, 30], v_r is chosen based on the point cloud density and average spacing, between [0.001m, 0.004m], and ε is set to 2, with other parameters remaining unchanged. Results and errors for different simplification rates are compared. As shown in Figure 15, Figure 16 and Tables 7.

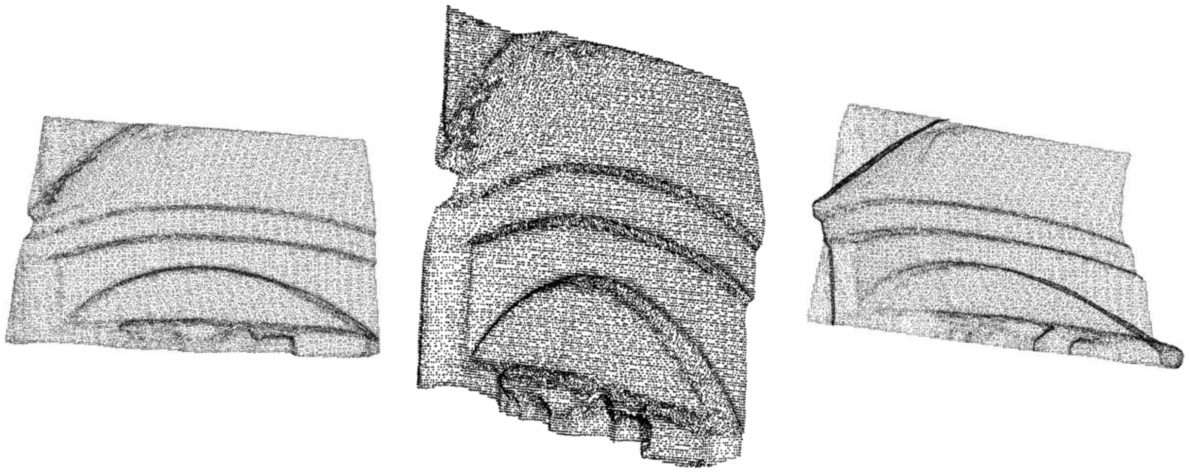


(a)

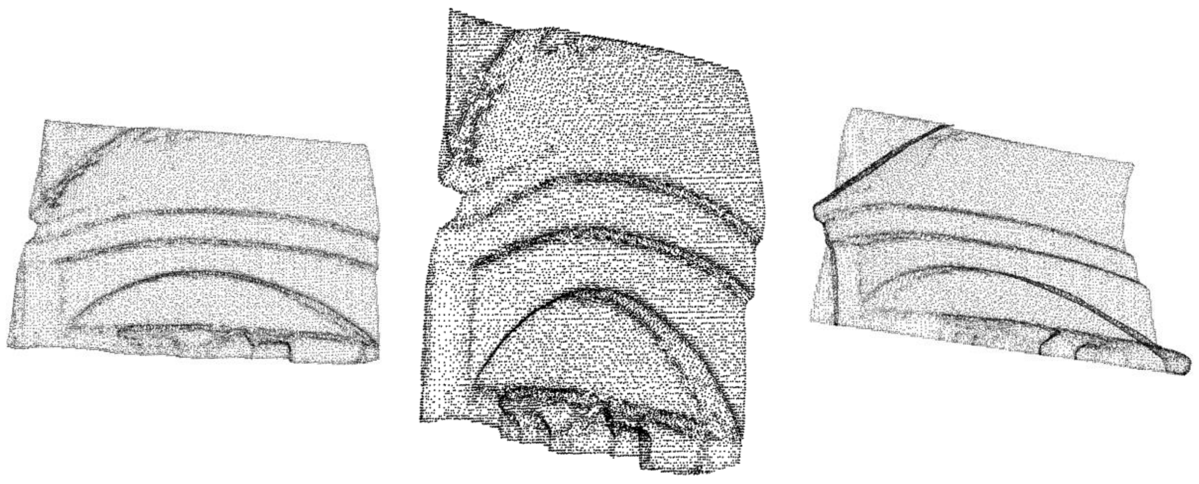


(b) (c) (d) (e)

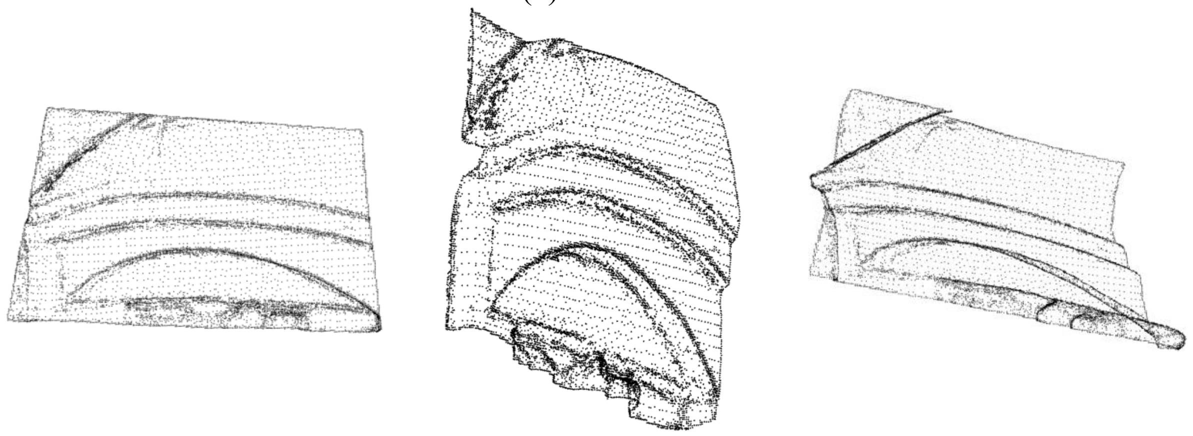
Figure 15: Results with a simplification rate of 78.55% under the real point cloud. (a) original point cloud (b) edge points (green) (c) feature points (red) (d) non-feature points (blue) (e) all points (black)



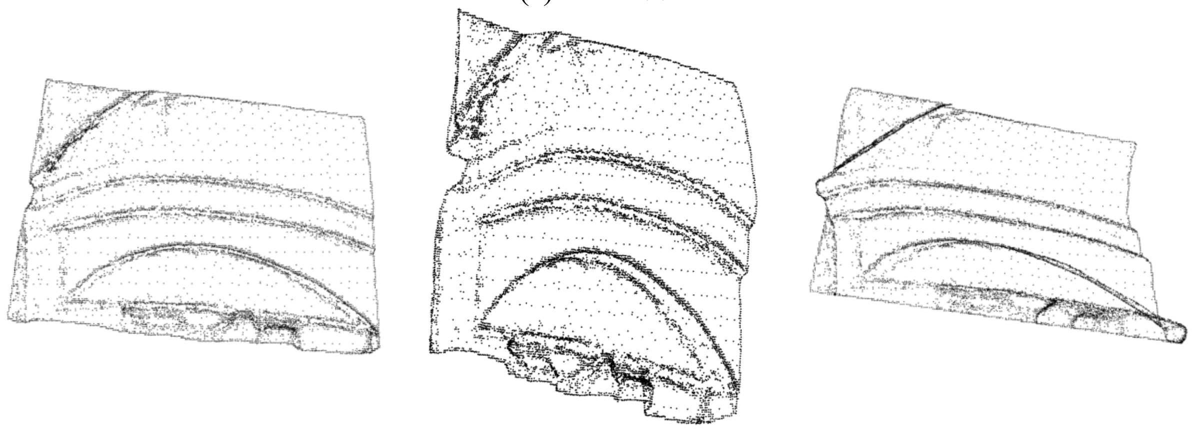
(a) 44.04%



(b) 58.63%



(c) 78.55%



(d) 81.71%

Figure 16: Results of Different Simplification Rates from Various Perspectives for the Real Point Cloud. (a) 44.04% (b) 58.63% (c) 78.55% (d) 81.71%

Table 7:   Error Comparison for Real Point Clouds at Different Simplification Rates.

Simplification Rate (%)	**Maximum Error(m)**	**Average Error (m)**	**RMSE(m)**	**Change Ratio of Volume(%)**

44.04	0.001399	0.000356	0.00055	0.05
58.63	0.001693	0.000522	0.000701	0.08
78.55	0.003785	0.001135	0.00139	0.22
81.71	0.006453	0.001687	0.002135	0.27

As shown in Figure 15(c), at a simplification rate of 78.55%, the point cloud still exhibits the uneven features at the bottom, with these details being well-preserved. However, smaller details in the upper-left corner and the central arc-shaped depressions are not fully retained, indicating feature loss in these areas during simplification. This may be because these regions have smoother or less prominent features compared to the more pronounced undulations, leading them to be deemed less important and removed during the simplification process. In other words, the feature selection and preservation mechanism performs better with large-scale features but is somewhat inadequate for small-scale or complex geometric details.

From Figure 16 and Tables 7, it is evident that at a 44.04% simplification rate, the geometric error is relatively small, and the point cloud details are well-preserved, showing high similarity to the original point cloud. As the simplification rate increases, geometric error rises significantly, and the point cloud becomes progressively sparser. At simplification rates of 78.55% and 81.71%, details are significantly reduced, especially at 81.71%, where the geometric shape becomes coarse, which aligns with the high geometric errors in the data.

Conclusion and Recommendation

This study demonstrates that by dynamically adjusting the neighborhood size to more accurately reflect local geometric features, the proposed method outperforms traditional fixed-neighborhood approaches. The method adjusts the neighborhood size based on regional characteristics, effectively capturing details in the point cloud and avoiding issues of information loss and feature confusion caused by either too small or too large neighborhoods. Additionally, the study uses a partitioning strategy to divide the point cloud into feature and non-feature regions, balancing feature preservation and simplification rate effectively.

In feature regions, point importance is calculated using a combination of normal vector differences, projection distances, spatial distances, and curvature differences, with varying retention thresholds set to ensure the preservation of geometrically significant points. For non-feature regions, a regional centroid-based simplification method is employed, reducing the number of points by mapping them to centroid points while preserving overall shape and edge features. The final simplification integrates edge points, feature points, and non-feature points, and measures errors using maximum geometric error, average geometric error, and root mean square error.

The experimental results indicate that the point cloud simplification method proposed in this study shows improved performance in preserving important feature details, such as those of the ears, thighs, and tail, compared to traditional methods, though there are still some areas where it could be further enhanced. However, there are limitations when it comes to smoother or less prominent geometric details in real point clouds. Future improvements should focus on enhancing the ability to identify and preserve such fine features to better control errors.

References

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. T. (2001). Point set surfaces. In Proceedings Visualization, 2001. VIS'01. (pp. 21-29). IEEE.

Benhabiles, H., Aubreton, O., Barki, H., & Tabia, H. (2013). Fast simplification with sharp feature preserving for 3D point clouds. In 2013 11th international symposium on programming and systems (ISPS) (pp. 47-52). IEEE.

Cignoni, P., Rocchini, C., & Scopigno, R. (1998). Metro: measuring error on simplified surfaces. In Computer graphics forum (Vol. 17, No. 2, pp. 167-174). Oxford, UK and Boston, USA: Blackwell Publishers.

Chen, Y., & Yue, L. (2016). A method for dynamic simplification of massive point cloud. In *2016 IEEE International Conference on Industrial Technology (ICIT)* (pp. 1690-1693). IEEE.

Chen, H., Cui, W., Bo, C., & Yang, N. (2023). Point cloud simplification for the boundary preservation based on extracted four features. *Displays*, *78*, 102414.

Dyn, N., Iske, A., & Wendland, H. (2008). Meshfree thinning of 3D point clouds. Foundations of computational Mathematics, 8, 409-425.

Demantké, J., Mallet, C., David, N., & Vallet, B. (2011). Dimensionality based scale selection in 3D lidar point clouds. In *Laserscanning*.

Gao, Y., Ping, C., Wang, L., & Wang, B. (2021). A simplification method for point cloud of t-profile steel plate for shipbuilding. Algorithms, 14(7), 202.

Gong, M., Zhang, Z., & Zeng, D. (2021). A new simplification algorithm for scattered point clouds with feature preservation. Symmetry, 13(3), 399.

Gan, Z., Ma, B., & Ling, Z. (2023). PCA-based fast point feature histogram simplification algorithm for point clouds. *Engineering Reports*, *6*(7), e12800.

Huang, w. M., xiao, z. X., yun, p. Z., & wu, x. J. (2010). Point cloud simplification with boundary points reservation. *Journal of Computer Applications*, *30*(2), 348.

Hackel, T., Wegner, J. D., & Schindler, K. (2016). Fast semantic segmentation of 3D point clouds with strongly varying density. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, *3*, 177-184.

He, E., Chen, Q., Wang, H., & Liu, X. (2017). A curvature based adaptive neighborhood for individual point cloud classification. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *42*, 219-225.

Ji, C., Li, Y., Fan, J., & Lan, S. (2019). A novel simplification method for 3D geometric point cloud based on the importance of point. *Ieee Access*, *7*, 129029-129042.

Linsen, L., Prautzsch, H., (2001). Local versus global triangulations. In: Proceedings of Eurographics, Manchester, UK, 5–7 September, pp. 257–263.

Luebke, D. P. (2001). A developer's survey of polygonal simplification algorithms, IEEE Comput. Graphics Appl., Vol. 21, no. 3, pp. 24-35.

Li, Z., Ding, G., Li, R., & Qin, S. (2014). A new extracting algorithm of k nearest neighbors searching for point clouds. *Pattern recognition letters*, *49*, 162-170.

Martin, R., Stroud, I., & Marshall, A. (1997). Data reduction for reverse engineering. RECCAD, Deliverable Document, 1, 111.

Pauly, M., Gross, M., & Kobbelt, L. P. (2002). Efficient simplification of point-sampled surfaces. In IEEE Visualization, 2002. VIS 2002. (pp. 163-170). IEEE.

Song, H., & Feng, H. Y. (2009). A progressive point cloud simplification algorithm with preserved sharp edge data. The International Journal of Advanced Manufacturing Technology, 45, 583-592.

Tang, H., Shu, H. Z., Dillenseger, J. L., Bao, X. D., & Luo, L. M. (2007). Moment-based metrics for mesh simplification. Computers & Graphics, 31(5), 710-718.

Wang, H. T., Zhang, L. Y., Du, J., Li, Z. W., & Zhou, R. R. (2007). Simplification and error analysis based on implicit surface for measuring points-sets. Journal of image and graphics, 12(N11), 2114-2118.

Wang, S., Hu, Q., Xiao, D., He, L., Liu, R., Xiang, B., & Kong, Q. (2022). A new point cloud simplification method with feature and integrity preservation by partition strategy. Measurement, 197, 111173.

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS Journal of Photogrammetry and Remote Sensing, 105, 286-304.

Yuan, S., Zhu, S., Li, D. S., Luo, W., Yu, Z. Y., & Yuan, L. W. (2018). Feature preserving multiresolution subdivision and simplification of point clouds: A conformal geometric algebra approach. *Mathematical Methods in the Applied Sciences*, *41*(11), 4074-4087.

Yang, Y.C., & Jaw, J.J., (2023). A comparative analysis of photogrammetric point cloud and mesh models for 3D object representation. *44th Asian Remote Sensing Conference, October 30–November 3 2023, Taipei, Taiwan.*