

# Error-Controllable Simplification of Point Cloud

Yichen Li<sup>1</sup>, Mingqiang Wei<sup>2</sup>, Jianhuang Wu<sup>3</sup>, and Mingyong Pang<sup>1</sup>

<sup>1</sup> Nanjing Normal University, Nanjing, P.R. China

<sup>2</sup> The Chinese University of Hong Kong, Hong Kong

<sup>3</sup> Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences  
yichen.lee.njnu@gmail.com, mqwei@cse.cuhk.edu.hk,  
jh.wu@siat.ac.cn, panion@netease.com

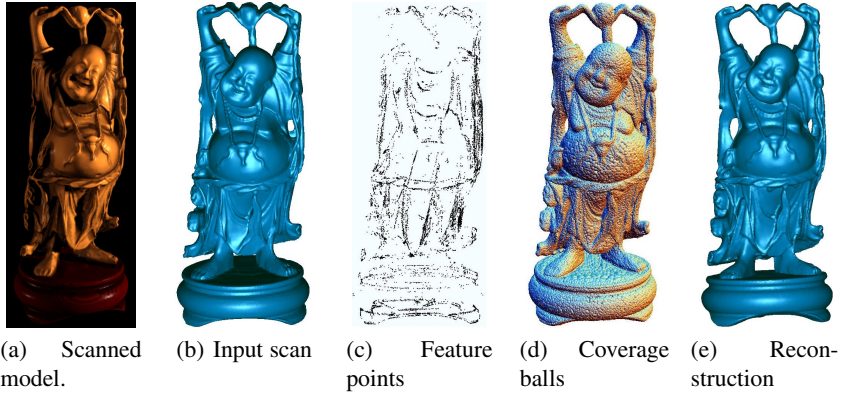
**Abstract.** Point cloud simplification has become a vital step in any point-based surface processing pipeline. This paper describes a fast and effective algorithm for point cloud simplification with feature preservation. First, feature points are extracted by thresholding curvatures; Second, for non-feature points, they are covered by distinct balls, the points in each ball are substituted by an optimized point. Thus, the simplified point cloud consists of extracted feature points and optimized points. This algorithm is able to produce coarse-to-fine models by controlling a general error level. But the error level of each ball may be adaptively adjusted according to the local curvature and density that can avoid holes generation during the simplification process. Finally, the simplified points are triangulated by Cocone algorithm for surface reconstruction. This algorithm has been applied to a set of large scanned models. Experimental results demonstrate that it can generate high-quality surface approximation with feature preservation.

**Keywords:** Point Cloud Simplification, Surface Reconstruction, Quadric Error Metrics.

## 1 Introduction

Due to recent advances in point cloud acquisition techniques, 3D object boundary surfaces are now commonly acquired with sub-millimeter accuracy. The initial output of acquisition devices such as laser range scanners therefore generally consists of point clouds of considerable redundancy. Unfortunately, the huge numbers of point clouds will bring us a great deal of trouble in the downstream processing and the data storage, transmission and rendering. By simplifying the point set first, the surface reconstruction from simplified point cloud is accelerated significantly and the mesh simplification step is avoided altogether. Furthermore, with the increasing availability of powerful point-based modelling[1] and visualisation[2] techniques, the simplification of dense point clouds for subsequent point-based rather than polygonal mesh-based processing plays a rather important role by itself. In either case, point cloud simplification represents a vital step in point-based surface processing pipeline.

Obviously the efficiency of point cloud simplification algorithm is essential for the large scale input data. Meanwhile, to guarantee the quality of simplification result for following geometric processing, we must take geometric features into account during simplification process. Generally, clustering method may achieve high efficiency while



**Fig. 1.** Simplification and reconstruction of a Buddha model(543652 points). From a raw scan with significant point-sampled data, our algorithm extracts the feature points, and calculates the optimized points for the non-feature points. Those two kinds of points are then blended to produce a simplified version for surface reconstruction.

losing a lot original features, contrarily, particle simulation method could obtain satisfied result while losing some efficiency[12]. So we have to hold the tradeoff carefully between quality and efficiency of simplification.

Our algorithm is an improved version of the algorithm proposed by Othake [20]. In our implementation, we sufficiently consider the feature preservation and can deal with the non-uniform point cloud. We first extract the feature points by thresholding curvatures, and then simplify the non-feature points by solving a collection of error functions. The feature points and optimized points are assembled to reconstruct mesh surface using Cocone [21] algorithm. Fig. 1 illustrates the pipeline of our approach. Experimental results demonstrate that our method could reserve enough original feature without loss of efficiency. In the rest of this paper, it is organized as follows. Section 2 gives a brief summary of related work and section 3 and 4 detail the preprocessing and the algorithm. Experimental results and analysis are given in Section 5, followed by concluding remarks in Section 6.

## 2 Related Work

In recent years, many researchers have focused on how to simplify redundant point clouds efficiently and effectively. The re-sampling method calculates a set of new samples from the original point cloud based on certain rules. Dey et al. [3] present a point cloud simplification approach, and adopt local curvature to detect the redundancy in the input point cloud and to ensure relevant point densities; This is accomplished by exploiting a 3D Voronoi diagram. Alexa et al. [4] uniformly reduce point cloud redundancy by estimating a point's contribution to the moving least squares (MLS) representation of the underlying surface. Those points contributing the least are subsequently removed. This method does not guarantee the absence of insufficiently dense output

point sets. Scheidegger et al. [5] extended the MLS approach to simplify point set surfaces and constructing a high-quality triangulation. Kalaiah and Varshney [6] get rid of redundancy through measuring the redundancy of each individual point based on the local geometric properties derived from the surrounding points. Lee et al. [7] presented a simplification method that reduces the number of points using part geometry information. In their work, the points are removed based on their normal vector values using 3D grids. Miao et al. [8] proposed a curvature-aware adaptive sampling method. An adaptive mean-shift clustering scheme is designed to generate non-uniformly distributed sampling points. Bossonnat et al. [9] describe a coarse-to-fine point simplification algorithm that randomly calculates a point subset and constructs a 3D Delaunay triangulation. Moening et al. [10] propose an intrinsic coarse-to-fine point simplification algorithm that guarantees uniform or feature-sensitive distribution. However, their method requires many computations and a large memory. Wu et al. [11] present a new sub-sampling technique for dense point clouds which is specially adjusted to the particular geometric properties of circular or elliptical surface splats. A global optimization scheme computes an approximately minimal set of splats that covers the entire surface while staying below a globally prescribed maximum error tolerance.

Generally, re-sampling methods can produce high quality of output models while the computation complexity is high. Clustering methods split the point cloud into a number of sub-sets, each of which is replaced by one representative sample. Pauly [12] proposes two types of clustering simplification algorithms. One type of the algorithms are based on incremental region-growing starts from a random seed point and a cluster is built by successively adding nearest neighbors. Such incremental region-growing is terminated when the size of the cluster reaches a maximum bound. The other type of the algorithms are hierarchical clusterings which compute the set of clusters recursively splits the point cloud using a binary space partition. Yu et al. [13] present an Adaptive Simplification Method (ASM) which is an efficient technique for simplifying point-based complex 3D models based upon hierarchical cluster tree structure. Shi et al. [14] also present an ASM by employing the k-means clustering algorithm to gather similar points together in the spatial domain and uses the maximum normal vector deviation as a measure of cluster scatter to partition the gathered point sets into a series of sub-clusters in the feature field. However, the proposed method may generate uniformly distributed sparse sampling points in the flat areas and necessary higher density in the high curvature regions.

The clustering methods usually are simple and fast, however, the simplified models are not satisfied without optimization processing. Another important simplification strategy for point-based surfaces is iterative reducing the number of points using an atomic decimation operator. Decimation operations are usually arranged in a priority queue according to an error metric that quantifies the error caused by the decimation [12]. This kind of method is similar to mesh-based simplification methods mentioned in [16]. Quadric error metrics is applied in [16] to measure the error caused by the mesh contraction. Iterative method can obtain both satisfied quality and speed except a larger memory consumption. In [17], a geometry-images-based simplification algorithm for point-sampled surfaces is proposed and the point set surfaces are simplified according to the curvature and simplified density fastly. Lee et al. [18] adopt discrete shape

operator to find the weight of the features of the 3D model and extract the relevant points for a dense input point set. Jong et al. [19] present a novel rapid and effective point simplification algorithm using local coplanar analysis on the basis of an octree data structure. By using the octree data structure, it proposes some hierarchical simplifications and renderings for the base model to suit user demand.

### 3 Preprocessing

Assuming a piecewise smooth surface  $\Phi$  that is approximated by a set of sampled points  $P = \{p_1, \dots, p_N\}$ , our goal is to fast and effectively simplify it and create a high-quality surface approximation. Typically, raw point cloud data are obtained by encoding a group of overlapped range images and the local density is higher at the regions that correspond to the overlapped regions of the range images. Thus, appropriate weights are assigned to the points for compensating density irregularities. We assign to each point  $p_i \in P$  a weight  $a_i$  defined by

$$a_i = \frac{1}{K} \sum_{j=1}^K \|p_i - p_j\|^2 \quad (1)$$

where  $\{p_j\}_{j=1}^K \in P$  are the  $K$  nearest neighbors of  $p_i$ . This weight scheme is sufficient to compensate the density irregularities.

To extract the feature points, calculate the optimized points and compute the 2D convex hull, we need to estimate the unit normals  $\aleph = \{n_1, \dots, n_N\}$  at the points of  $P$ . Generally, unit normals can be directly acquired via photometric stereo [22][23]. If the point cloud data provide no normal information, we employ a method relied on covariance analysis [24] to estimate them. The  $3 \times 3$  covariance matrix  $C$  for a sample  $p_i$  is given by

$$C = \begin{bmatrix} p_{i,1} - \bar{p} \\ \dots \\ p_{i,K} - \bar{p} \end{bmatrix}^T \cdot \begin{bmatrix} p_{i,1} - \bar{p} \\ \dots \\ p_{i,K} - \bar{p} \end{bmatrix}, \bar{p} = \sum_{j=1}^K p_{i,j} \quad (2)$$

where  $\{p_{i,1}, \dots, p_{i,K}\}$  is the  $K$  nearest neighbors of the point  $p_i$  in  $P$ . Since  $C$  is symmetric and positive semi-definite, all eigenvalues are real-values and all eigenvectors form an orthogonal frame [25]. The eigenvector corresponding to the smallest eigenvalue is taken as the normal vector of the point  $p_i$ .

### 4 Point Simplification Algorithm

To simplify the point cloud, we first extract the feature points  $P^f = \{p_1, \dots, p_m\}$  ( $0 \leq m \leq N$ ), by thresholding point curvature, and then generate a collection of balls centered at  $\{c_1, \dots, c_n\}$  with adaptive radius  $\{r_1, \dots, r_n\}$  covering the non-feature points. The ball generation used in this paper is an improved version of the approach proposed by Ohtake [20] for creating an approximately minimal set of spheres to cover the whole point-sampled surface. The algorithm is described as follows:

1. Extract the feature points according to curvature; The feature points are protected and do not participate the following processing.
2. Set all non-feature points as uncovered.
3. Select point  $c$  from the set of uncovered points without regard to the order of the selection and label this point as covered.
4. For the selected point  $c$ , evaluate  $r$ , the radius of ball centered at  $c$ , and an optimized point  $v$ .
5. Calculate local curvature  $cur$  and density  $dens$  at point  $c$ . According to the local properties, adjust the error threshold, and repeat step 4.
6. Project the non-feature set  $\{\|p - c\| < r\}$  onto the plane tangent to  $c$ . Compute the 2D convex hull of the projections of  $P_c^r$ . Label the points of  $P_c^r$  which are projected strictly inside the convex hull as covered.
7. Terminate the process if there are no more uncovered points. Otherwise return to step 3.

#### 4.1 Feature Points Extraction

Feature points describe the basic shape of the object and have a significant influence on the quality of surface reconstruction. Thus, in order to avoid geometric information lost, before simplifying the point cloud, we extract the feature points by thresholding curvature. In further simplification process, those feature points will not be dealt with.

There are many kinds of feature points, we mainly focus on silhouettes and corners, and extract them, because they are important visual cues for shape perception [26] and are very effective at conveying shapes [27][28]. More precisely, silhouettes are those points positioned at convex or concave boundaries, and corners are located at sharp regions. If point  $p$  is a feature point, the curvature at this point is higher, i.e., the sum of the distances from the  $k$  nearest neighbors of  $p$  to the tangent plane at  $p$  is higher. Otherwise if point  $p$  is a non-feature point, the curvature is lower, i.e., the sum of the distances from the  $k$  nearest neighbors of  $p$  to the tangent plane at  $p$  is lower. As seen in Fig. 2, hollow points refer to the  $k$  nearest neighbors of point  $p$ , the distance sum is higher which means the curvature at point  $p$  is higher, thus point  $p$  in the left image is a feature point. In contrast, point  $p$  in the right image is a non-feature point.

Based on the analysis described above, we construct the measurement function as follows:

$$measure(p) = \frac{1}{K} \sum_{j=1}^K |(p - p_j) \cdot n_p| \quad (3)$$

where  $\{p_j\}$  are the  $k$  nearest neighbors of point  $p$ ,  $n_p$  is the unit normal at  $p$ ,  $(p - p_j) \cdot n_p$  refers to the distance from the  $j^{th}$  nearest neighbor to the tangent plane at  $p$ . If  $measure(p)$  is higher than a threshold  $\delta$ , this means point  $p$  is considered as a feature point; Otherwise it is a non-feature point. To distinguish feature points and non-feature points, we set threshold  $\delta$  as follows:

$$\delta = \frac{\alpha}{N} \sum_{i=1}^N measure(p_i) \quad (4)$$

where  $N$  is the number of points,  $\alpha$  is the adjustment factor. If  $measure(p)$  is higher than  $\delta$ , point  $p$  is set as a feature point. Otherwise it is a non-feature point. Fig. 3 and Fig. 4 demonstrate the extracted quality of setting different  $\alpha$  for the same models and same  $\alpha$  for different models.

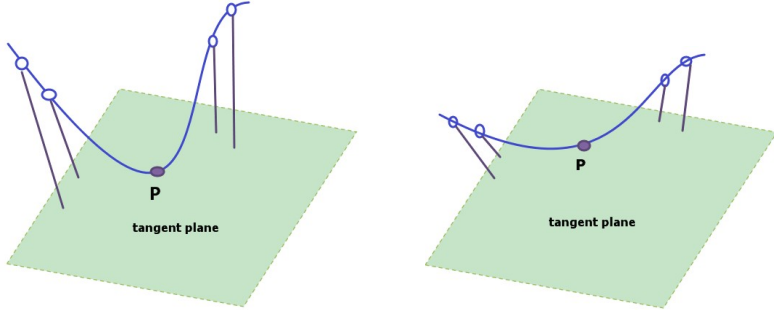


Fig. 2. Measuring feature and non-feature points by curvature

### 4.2 Point Cloud Density

The extraction methods of point cloud characteristics can be roughly classified into two categories: based on the point-to-point distance method and based on the clustering method. We adopt the method based on point-to-point distance to analyze the density distribution of the point cloud. For non-feature point set  $P' = \{p'_1, \dots, p'_M\}$  ( $0 < M < N$ ), we denote the distance between point  $p'_i$  and  $p'_j$  by  $dis(p'_i, p'_j)$ , local density at point  $p'_i$  can be defined by

$$dens_i = \min(dis(p'_i, p'_j)), (1 \leq j \leq M, i \neq j) \tag{5}$$

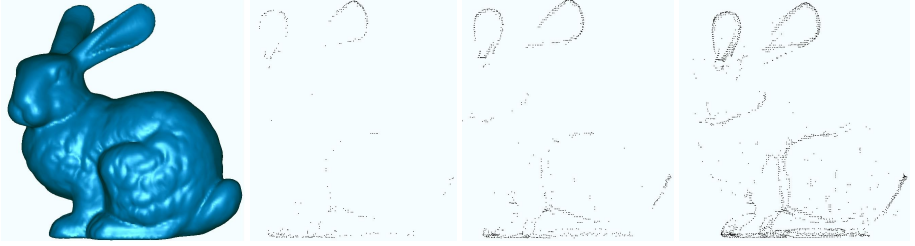
The average density of non-feature point set  $P'$  is given by

$$\overline{dens}_i = \frac{1}{M} \sum_{i=1}^M dens_i \tag{6}$$

where  $M$  is the number of non-feature points. Thus, when we decrease  $dens_i$ , the algorithm selects more points locally. From the magnified fragment of the left image of Fig. 5, the density is changing in this region, thus, in the process of simplification, we will sufficiently consider such region.

### 4.3 Curvature Estimation

In order to preserve more details, when simplifying the point cloud, we will sufficiently consider the curvatures of the point cloud. There are many approaches to estimate the curvatures on discrete points, such as Paraboloid Fitting approach, Circular Fitting approach, Gauss-Bonnet approach and so on. The paraboloid fitting approach is used



(a) Bunny model (35292points) with  $\alpha$  from left to right assigned to 4.5 (278points), 3.5 (702points) and 2.5 (1086points), respectively.



(b) Hand model (39325points) with  $\alpha$  from left to right assigned to 4.5 (372points), 3.5 (880points) and 2.5 (2228points), respectively.

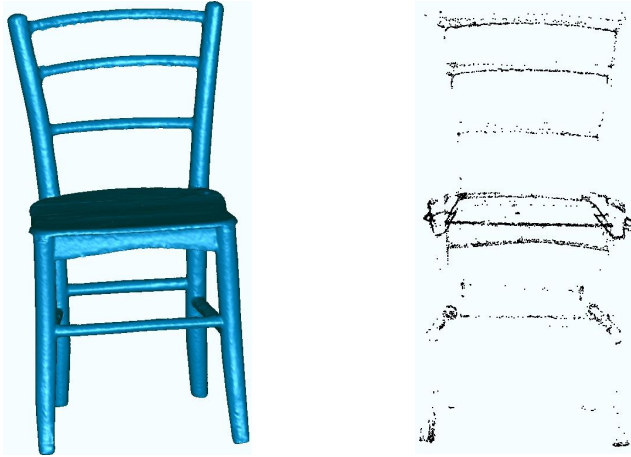
**Fig. 3.** Different parameter values for the same models

in this paper to estimate the mean curvature, because it is more robust when using paraboloid in the local neighborhood of a point to estimate the curvature, and can result in most optimized results. Assuming paraboloid equation is

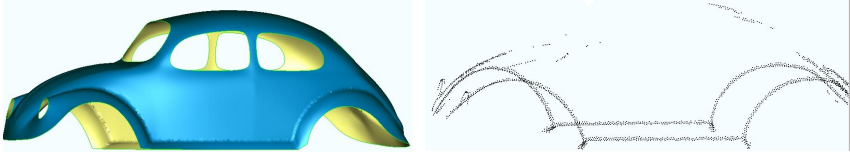
$$z = ax^2 + bxy + cy^2 \quad (7)$$

For each non-feature point  $p$ , we fit a paraboloid by least square method using point  $p$  and its  $k$  nearest neighbors. The coefficients  $a$ ,  $b$  and  $c$  can be computed by solving a linear equation, i.e.,  $Ax = b$ , where matrix

$$A = \begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ x_{K+1}^2 & x_{K+1}y_{K+1} & y_{K+1}^2 \end{bmatrix}, \quad (8)$$



(a) Chair model (212634points) and its extracted feature points (4329points).



(b) Car model (30024) and its extracted feature points (1855).

**Fig. 4.** Same parameter value ( $\alpha = 3.0$ ) for different models

$$x = \{a, b, c\}^T, \tag{9}$$

$$b = \{z_1, z_2, \dots, z_{K+1}\}^T. \tag{10}$$

This overdetermined equation can be solved by householder transformation to obtain coefficients. Thus, the mean curvature at  $p$  is

$$H_i = a + c, \tag{11}$$

and the mean curvature of the non-feature point set is

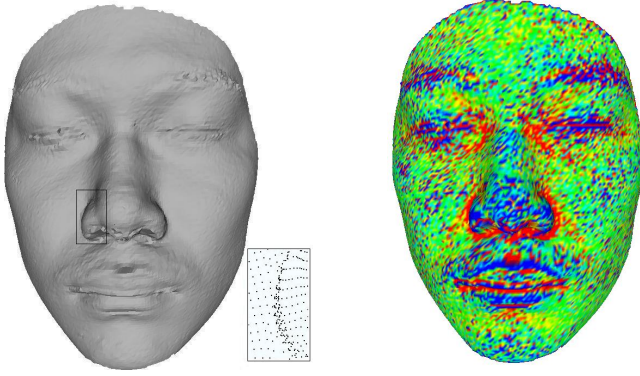
$$\bar{H} = \left(\sum_{i=1}^M H_i\right)/M. \tag{12}$$

The right image of Fig. 5 shows the mean curvature visualization.

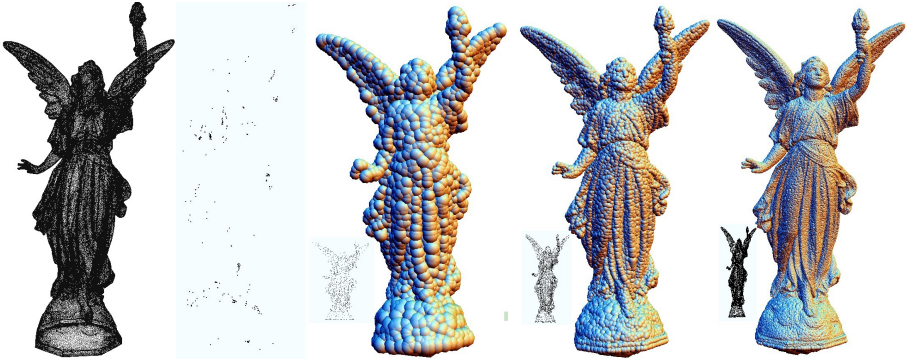
#### 4.4 Ball Radius and Optimized Point

The core of the algorithm is computing a set of coverage balls with adaptive radius  $\{r_1, \dots, r_m\}$  and optimized points  $\{v_1, \dots, v_m\}$  in each ball. The optimized points





**Fig. 5.** Face model with varying local density (magnified fragment) and mean curvature visualization using different colors



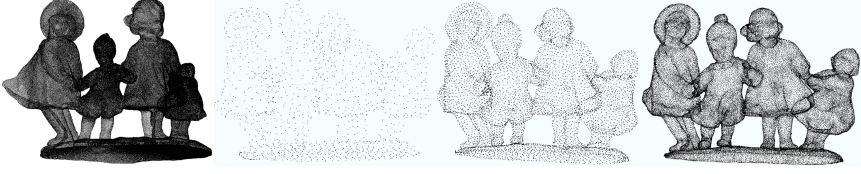
**Fig. 6.** Lucy model (262909 points) with different initial  $T_{err}$ . From left to right, original model, extracted feature points (264 points with  $\alpha = 3.5$ ), coverage balls with  $T_{err} = 10^{-4}$  (4646points),  $10^{-5}$  (21347points),  $10^{-6}$  (91019points), respectively.

will substitute the non-feature points in each ball for simplification. The approach is based on local quadric error minimization strategy, and is an improved version of the approach described by Ohtake[20] which set a fixed error threshold. In contrast, our approach will adjust the error threshold according to local curvature and density.

For each selected ball centered at  $c$ , we calculate the adaptive radius  $r$  and an optimized point  $v$  in it. According to [20][16][11], we define a quadric error function for each point  $c$  as follows:

$$Q(c, r, x) = \sum_j w_j G_R(\|p_j - c\|)(n_j \cdot (x - p_j))^2 \quad (13)$$

where weights  $\{w_j\}$  are defined by (1),  $p_j$  is a non-feature point within a bounding sphere centered at  $c$  with radius  $R$ ,  $n_j$  is the unit normal at point  $p_j$ ,  $x$  is the potential



**Fig. 7.** Children model (724742 points) and simplified model with different error values  $T_{err}$ . From left to right, original model, models with  $T_{err} = 10^{-4}$ ,  $10^{-5}$  and  $10^{-6}$ , respectively.

optimized point (we need further judge whether  $x$  is an optimized point),  $n_j(x - p_j)$  denotes the distance from point  $x$  to the tangent plane at point  $p_j$ ,  $r$  is the radius of covered ball,  $G_R(\|\cdot\|)$  is a Gaussian-like function as defined by

$$G\sigma(\rho) = \begin{cases} \exp(-8(\rho - \sigma)^2) & \text{if } |\rho| \leq \sigma/2, \\ 16/e^2(1 - |\rho|/\sigma)^4 & \text{if } \sigma/2 < |\rho| \leq \sigma, \\ 0 & \text{if } \sigma < |\rho|. \end{cases}$$

In practice, we set  $R = 2r$ . Function  $Q(c, r, x)$  computes a weighted sum of the squared distances from point  $x$  to the tangent planes at  $\{p_j\}$  within spherical region  $\|p_j - c\| \leq R$ . If  $r$  in (13) is fixed, point  $x_{min} = x_{min}(r)$ , the minimizer of  $Q(c, r, x)$  is easily found by solving a system of linear equations. Thus we set error function

$$E(r) = \frac{1}{L} \sqrt{Q(c, r, x_{min})} \quad (14)$$

where  $L$  is the length of a main diagonal of the bounding box of the point cloud. As proved in [29], the error function  $E(r)$  weighs the curved degree of the reconstructed surface inside sphere  $\|x - c\| < R$ . We get  $r$  by solving the following equation






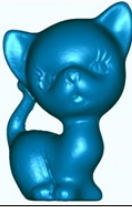
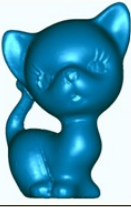
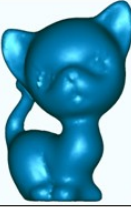




$$E(r) = T_{err} \quad (15)$$

where  $T_{err}$  is a user-controlled accuracy.  $E(r)$  is monotonically decreasing as  $r \rightarrow 0$ , thus, we can use bisection method to solve (14).

Once  $r$  is fixed, we check whether  $x_{min}$  lies inside or outside of region  $\|x - c\| < r$ . If  $x_{min}$  is within this region, we use it as the optimized point  $v$  associated with the ball centered at  $c$ . Otherwise, we set  $v = c$ , because the surface curvature in this region is large and  $c$  there must be a sharp feature.

As seen, for each center  $c$ , the above-mentioned error threshold  $T_{err}$  is unaltered. Actually, adjusting  $T_{err}$  for different center  $c$  will be more desirable according to the local curvature and density. Thus, we compute the local curvature and density at point  $c$  for regulating the accuracy  $T_{err}$  as follows:

1. If mean curvature  $H_i$  at  $c_i$  is smaller than  $\overline{H}$  and  $dens_i$  is larger than  $\overline{dens}$ , this fact means that this region is relatively flatter, we can increasingly augment  $T_{err}$  for extending  $r$  to simplify more points.
2. If mean curvature  $H_i$  at  $c_i$  is larger than  $\overline{H}$  and  $dens_i$  is smaller than  $\overline{dens}$ , which indicates that this region contains more features, we decrease  $T_{err}$  for shrinking  $r$  to save more points.

Model	Original model	Simplified model		
		$10^{-7}$	$10^{-6}$	$10^{-5}$
Head				
points	146773	102425	42807	5578
Simplified rate		30%	70%	96%
Kitten				
points	137098	97204	32773	4215
Simplified rate		29%	76%	96%
Armadillo				
points	165954	132018	77762	8701
Simplified rate		20%	53%	94%

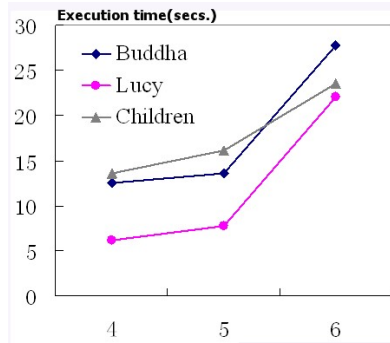
**Fig. 8.** The reconstruction results produced by Cocone algorithm using the simplified points compared with the original models are demonstrated. As seen, controlling different  $T_{err}$  can generate different levels of detail.

As shown in Fig. 6, the error  $T_{err}$  controls the approximation accuracy, large  $T_{err}$  results in large radius of coverage balls and simplify more points. Otherwise, small  $T_{err}$  preserves more points. Fig. 7 demonstrates how different errors  $T_{err}$  control the final simplified results.

## 5 Results and Analysis

### 5.1 Parameter Settings

To use our method, one must set two parameters beforehand, one is adjustment factor  $\alpha$  which determines the number of feature points. If  $\alpha$  is larger, the threshold becomes larger simultaneously, the number of feature points will naturally decrease. Otherwise if  $\alpha$  becomes smaller, there will be more points considered as feature points. Thus, in the implementation, user can select different factors for different models. The other



**Fig. 9.** The execution time of simplification with different error values  $T_{err}$  for different models. The digital 4, 5, 6 on the abscissa refer to  $T_{err} = 10^{-4}, 10^{-5}$  and  $10^{-6}$ , respectively.

Model	Preprocessing(s)	Simplification(s)			Reconstruction(s)		
		$10^{-7}$	$10^{-6}$	$10^{-5}$	$10^{-7}$	$10^{-6}$	$10^{-5}$
Head	2.25	26.76	12.01	3.80	77.77	28.33	2.86
Kitten	2.23	23.90	9.7	3.35	72.29	22.06	2.18
Armadillo	2.79	33.34	19.53	5.23	103.78	56.78	4.95

**Fig. 10.** Time measurements of three main stages of the surface reconstruction process for the models considered in Fig. 8

parameter is initial error  $T_{err}$  which controls the approximation accuracy. It determines the tolerance of the optimized point and the size of radius of covered ball. See Fig. 8 for more details.

### 5.2 Timing and Computational Complexity

The execution time of experimental results is measured on a Pentium 4 (2.99GHz) PC with 2GB of main memory. In the statistical data shown in Fig. 9 and Fig. 10, our simplification is fast in obtaining the preferable results. Since we use *kd-tree* to find nearest neighbors, the computational complexity is  $O(N \log N)$ , where  $N$  is the number of points.

### 5.3 Simplification Quality

The algorithm first extracts the feature points in the original model, and then generates adaptive balls covered the non-feature points. We use Cocone algorithm to produce

meshes, which takes the simplification results as input. Our error-controllable simplification algorithm can produce levels-of-detail results with feature preservation, and the simplification results can be used to generate high-quality meshes using Cocone algorithm.

## 5.4 Limitations

The proposed algorithm is lack of strict theoretic proof and is basically an application issue. In addition, this algorithm may not be to handle the sparse-point-set case.

## 6 Conclusions

We present a fast and robust algorithm for point cloud simplification. The algorithm is an improved version of Ohtake's approach. Our algorithm can reconstruct surfaces from uniform and non-uniform point clouds. In the simplification process, we enable the local properties to adjust the error threshold. The simplification results can ensure the surface reconstruction results by using Cocone algorithm.

**Acknowledgment.** The authors would like to thank the anonymous reviewers for their constructive comments. The authors would like to thank Shengzhou Luo, Seng Wang both from Shenzhen Institutes of Advanced Technology for their valuable suggestions. The authors would like to thank Luming Liang from Colorado School of Mines for his wording modification for this paper. The models used in this paper are courtesy of Stanford 3D Scanning Repository and AIM@SHAPE Repository. This work was supported by the National Natural Science Foundation of China (Grant No. 60803108).

## References

1. Linsen, L.: Point cloud representation. CS Technical Report 2001-3. Universitat Karlsruhe, Germany (2001)
2. Alexa, M., Behr, J., Cohen-Or, D., et al.: Point Set Surfaces. In: Proc. 12th IEEE Visualization Conf., San Diego, USA, pp. 21–28 (2001)
3. Dey, T.K., Giesen, J., Hudson, J.: Decimating samples for mesh simplification. In: Proc. of 13th Canadian Conference on Computational Geometry, pp. 85–88 (2001)
4. Alexa, M., Behr, J., Cohen-Or, D., et al.: Computer and rendering point set surface. IEEE Transaction on Visaulization and Computer Graphics 9(1), 3–15 (2003)
5. Scheidegger, C., Fleishman, S., Silva, C.: Triangulating point set surfaces with bounded error. In: Eurographics Symposium on Geometry Processing, pp. 63–72 (2005)
6. Kalaiah, A., Varshney, A.: Modeling and rendering of points with local geometry. IEEE Trans. Vis. Comput Graphics 9(1), 30–42 (2003)
7. Lee, K., Woo, H., Suk, T.: Point data reduction using 3D grids. The International Journal of Advanced Manufacturing Technology 18(3), 201–210 (2001)
8. Miao, Y., Pajarola, R., Feng, J.: Curvature-aware adaptive re-sampling for point- sampled geometry. Computer-Aided Design 41(6), 395–403 (2009)
9. Boissonnat, J.D., Cazals, F.: Coarse-to-fine surface simplification with geometric guarantees. In: Proc. of EUROGRAPHICS, pp. 490–499 (2001)

10. Moenning, C., Dodgson, N.A.: Intrinsic point cloud simplification. In: Proc. of 14th GrahIcon (2004)
11. Wu, J., Kobbelt, L.P.: Optimized sub-sampling of point sets for surface splatting. In: Proc. of EUROGRAPHICS, pp. 643–652 (2000)
12. Pauly, M., Gross, M., Kobbelt, L.P.: Efficient simplification of point-sampled surfaces. In: Proc. of IEEE Visualization, pp. 163–170 (2002)
13. Yu, Z., Wong, H., Peng, H., et al.: An adaptive simplification method for 3D point-based models. *Computer-Aided Design* 42(7), 598–612 (2010)
14. Shi, B., Liang, J., Liu, Q.: Adaptive simplification of point cloud using k-means clustering. *Computer-Aided Design* 43(8), 910–922 (2011)
15. Song, H., Feng, H.: A global clustering approach to point cloud simplification with a specified data reduction ratio. *Computer-Aided Design* 40(3), 281–292 (2008)
16. Garland, M., Heckbert, P.: Surface simplification using quadric error metrics. In: Proc. of SIGGRAPH, pp. 209–216 (1997)
17. Wang, R., Hang, S., Ye, X.: A novel simplification algorithm for point-sampled surfaces. In: International Conference on Multimedia and Ubiquitous Engineering, pp. 573–578 (2007)
18. Lee, P., Jong, B.: Point-based Simplification Algorithm. *WSEAS Transactions on Computer Research* 3(1), 61–66 (2008)
19. Jong, B., Lee, P.: Hierarchical Point Simplification Using Coplanar Criterion. In: Tseng 2006 IEEE Region 10 Conference, pp. 1–4 (2006)
20. Ohtake, Y., Belyaev, A., Seidel, H.-P.: An integration approach to meshing scattered point data. In: ACM Symposium on Solid and Physical Modeling, pp. 61–69 (2005)
21. Dey, T.K., Goswami, S.: Tight Cocone: a water tight surface reconstruction reconstructor. In: Proc. of 8th ACM Sympos. Solid Modeling Appl., pp. 127–134 (2003)
22. Tagliasacchi, A., Zhang, H., Cohen-Or, D.: Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics* 28(3), 1–9 (2009)
23. Nehab, D., Rusinkiewicz, S., Davis, J., et al.: Efficiently combining positions and normals for precise 3D geometry. *ACM Transactions on Graphics* 25(3), 560–568 (2005)
24. Hoppe, H., DeRose, T., Duchamp, T., et al.: Surface reconstruction from unorganized points. In: Proc. of SIGGRAPH, pp. 71–79 (1992)
25. Wu, J., Wei, M., Li, Y., et al.: Scale-adaptive surface modeling of vascular structures. *BioMedical Engineering Online* 9, 75 (2010)
26. Koenderink, J.: What does the occluding contour tell us about solid shape? *Perception* 13, 321–330 (1984)
27. Gooch, B., Sloan, P.J., Gooch, A., et al.: Interactive technical illustration. In: Proc. ACM Symposium on 3D Interactive Graphics, pp. 31–38 (1999)
28. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: Proc. of SIGGRAPH, pp. 517–526 (2000)
29. Heckbert, P.S., Garland, M.: Optimal triangulation and quadric-based surface simplification. *Computational Geometry: Theory and Application* 14(1-3), 49–65 (1999)