Simplification of Fan-Meshes Models for Fast Rendering of Large 3D Point-Sampled Scenes

Xiaotian Yan, Fang Meng, and Hongbin Zha

National Laboratory on Machine Perception
Peking University, Beijing, P.R. China
{yanxt, mengfang, zha}@cis.pku.edu.cn

Abstract. Fan-Meshes (FM) are a kind of geometrical primitives for generating 3D model or scene descriptions that are able to preserve both local geometrical details and topological structures. In this paper, we propose an efficient simplification algorithm for the FM models to achieve fast post-processing and rendering of large models or scenes. Given a global error tolerance for the surface approximation, the algorithm can find an approximately minimal set of FMs that covers the whole model surfaces. As compared with splat-based methods, the FM description has a large simplification rate under the same surface fitting error measurement.

1 Introduction

3D models and scene representations are of great potentiality to be integrated in Augmented Reality (AR) applications to enhance both realism in the environment rendering and interactive effects. To make them really useful in practical tasks, however, the geometrical descriptions of the models and representations should meet the following requirements: (1) The 3D datasets have to be simplified enough to allow for real-time rendering operations on general computing platforms. (2) Local editing or retargeting can be performed easily on the model surfaces. (3) The data structures of the descriptions are suitable for interactive and progressive transmission.

With advancements in 3D scanning technologies, we can obtain high-quality sampling points for large scenes much more conveniently than before [1]. Since the size of current available scanning data increases rapidly and the topological information becomes very complex, it is a hard work to perform post-processing on these huge sampling point datasets. In general, there are two approaches for the 3D data processing on original scanning data: (1) mesh-based data processing, where completed mesh models should be generated before the post-processing; (2) point-based processing that can be carried out directly on the sampling points. Traditional 3D data processing on simplification and editing operations are designed based on mesh models [2]. However, it is unpractical in most cases due to the expensive computational costs and storage requirements. In recent years, point-based geometry processing has gained increasing attention as an alternative surface representation, both for efficient rendering and for flexible geometry processing [3]. Since we need not to store and maintain globally consistent topological information during the data processing, it is easy to handle huge point-sampled geometry by data partition and integration operations [4]. Until now, some techniques have been developed for the simplification [5], rendering [6], transmission [7] of the point-sampled models. Compared with mesh models, however, the point-based approach has difficulty in realizing local editing on the point sets because there has no any connectivity information kept among the points. Moreover, point-based rendering are mostly implemented based on planar primitives, and hence local geometrical details have to be preserved with great care [8].

To solve the problem, we have presented a new kind of local mesh descriptions, called Fan-Meshes (FMs), which can not only represent local geometrical details but also allow for flexible operations on the models. FMs have advantages for the post-processing of huge sampling point datasets, such as building multi-resolution data structure or interactive transmission. In order to combine FMs into AR applications, in this paper, we propose an efficient simplification and rendering algorithm for models represented by FMs. We can decrease the number of FMs greatly by a re-sampling operation. For a given simplification rate, more accurate geometrical representation will be acquired by FMs than by using splats [10], the most popular primitives in point-based rendering. Experimental results illustrate that our simplification method can improve the usage of FMs in AR applications, and a good tradeoff can be achieved between the running speed and visual effects.

2 Generation of FM

At first, the definition of FM is given in this section. Then we explain in detail the FM's generation process that consists of two steps: organizing and selecting neighbors for a center point, and determining the vertices and region of FM. At last, we present a unified error-based criterion to compare FMs with splats.

2.1 FM's Definition

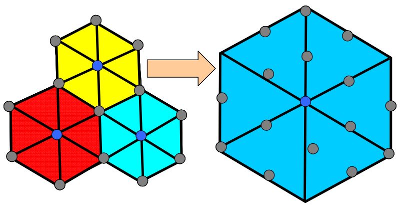
Fig. 1. Left: Original FM; Right: FM we use in this paper

FM is a geometrical primitive we proposed in [9] to reconstruct 3D models and scenes represented by dense scanning point clouds. An original FM consists of three components: a center point, its neighbor points and the connection relationship among these points. As illustrated in the left of Fig. 1, the blue points are center points, the gray ones are their neighbors and six triangles with the same color form a FM. The model surface can be visualized without holes after rendering a large number of FMs with enough overlaps.

The definition of FMs can be modified further to make them adaptive to the shape changes on the model surface. The number of triangles that a FM contains depends on local geometrical properties and its shape depends on the neighbors' distribution. Since original point clouds are very dense and many adjacent FMs may locate nearly on the same planes in large scenes, the FMs will be redundant if we draw them all.

We can use a big FM shown in the right of Fig. 1 to cover all the co-planar points. That is the FM we use in this paper with new properties: First, while all vertices of the original FM are selected form the point cloud, most of the FM's vertices here are not certainly ones on the initial model. Second, in order to get better visual effects, we assume that every FM consists of six triangles and the FM's projection on the least squares plane is a regular hexagon.

2.2 Organizing and Selecting Neighbors

Given an error tolerance *E*, we must select the largest neighbor which makes the error of FM stay below *E* for a center point. This step is similar to the splat generation used in [10]. We will discuss the differences between FM and splats in detail in Section 2.4.

The first task is to find the largest neighbor for a center point P_0 under the error toleration *E*. We estimate the local normal direction $\vec{n}$ by fitting a least squares plane *F* to P_0 and its *k* nearest neighbors. For the *i*th nearest neighbor P_i of P_0, we compute its sign distance h_i and projection distance d_i in the best fitting plane *F* as

$$h_i = \vec{n} \bullet \left(\vec{P_i} - \vec{P_0} \right), \quad (1)$$

$$d_i = \left\| \left(\vec{P_i} - \vec{P_0} \right) - h_i \vec{n} \right\| . \quad (2)$$

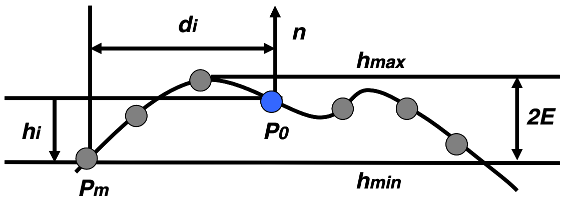

Fig. 2. Find neighbors in a FM

We rank the neighbors $P_1, P_2, \ldots, P_K$ by their projection distance (Eq. 2). Then we grow the FM by adding other neighbor points and update the interval of their sign distances $[h_{min}, h_{max}]$. The growing stops as soon as the interval between the h_{max} and h_{min} becomes larger than 2*E*. As illustrated in Fig. 2, the added neighbors $P_1, P_2, \ldots, P_m$ are the points we need to generate a FM.

2.3 Determining the Vertices and Region of FM

$P_1, P_2, \ldots, P_m$ are the selected neighbors of P_0, and they are ranked by their projection distances. We denote their projections on the best fitting plane *F* as $Q_1, Q_2, \ldots, Q_m$ respectively. The vertices of FM are $V_1, V_2, \ldots, V_6$ and their projections on *F* are $R_1, R_2, \ldots, R_6$. We compute the FM that obeys the following three rules: First, the center of FM is P_0 and R_1 is coincident with Q_m. Second, $R_1, R_2, \ldots, R_6$ form a regular hexagon on *F*. Third, the sum of distances from $P_1, P_2, \ldots, P_m$ to FM is minimal.

It takes too long a time to solve the problem by a least squares method. Here we present a fast algorithm to generate FM by an approximation method. Since the locations of $R_1, R_2, \ldots, R_6$ can be computed by the first and second rules, the vertex V_j locates in the line that is perpendicular to *F* and through R_j ($j=1, 2, \ldots, 6$). An example of computing the vertex V_3 is shown in Fig. 3. First, we form a coordinate system that centers at P_0, and we assume that the coordinates of R_3 are $(1, 0, 0)$. Then we find a point set $\{S_1\}$ of the neighbors whose projections fall into triangle $P_0R_3R_4$. The center of this point set is denoted as P_a (x, y, z) and its projection on *F* is Q_a $(x, y, 0)$. Lines P_0P_a and P_0Q_a have points of intersection with plane $V_3R_3R_4$ respectively. The red line segment in Fig. 3 connects the two intersection points together, and its length is

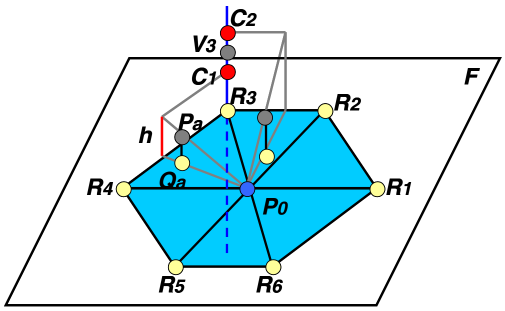

Fig. 3. Determining the vertices

$$h = \sqrt{3}z/(\sqrt{3}x + y) \ . \quad (3)$$

We mark a red point C_1 at line R_3V_3 so that the length of R_3C_1 is equal to h. Then the sum of distances from neighbors in set $\{S_1\}$ to plane $P_0P_aC_1$ is minimal. With a new point set $\{S_2\}$ of the neighbors whose projections fall into triangle $P_0R_2R_3$, we can compute another red point C_2 at line R_3V_3. The vertex V_3 locates in the middle of line segment C_1C_2. We calculate the other five vertices similarly and form the final FM. By this method, the generated FM makes the sum of distances from all the neighbors to it approximately minimal.

In order to perform the simplification process, we have to define an coverage measurement for each FM, which depends on two factors: 1) the area S_a it covers on the surface; 2) the number of points it overlaps.

The area of each sample point is computed first. We define that the area of P_0 is the area of the FM's projection hexagon on F. Since the projection hexagon is regular and its edge length is equal to the longest projection distance d_m, the area of P_0 is

$$A(P_0) = 3\sqrt{3}d_m^2/2 \ . \quad (4)$$

Then we sum the areas of the points that overlapped by the FM and consider the sum as the cover region of FM. The cover region of FM is not equal to the area S_a, but it can be used as the coverage measurement which is given by

$$A(FM) = \sum_{k=0}^{k \le m} A(P_k) \ . \quad (5)$$

The value of $A(P_0)$ depends on the local geometrical properties around P_0, and it is close to the area S_a. As the local geometrical properties around P_k and P_0 are similar, the area value $A(P_k)$ is approximate to $A(P_0)$. As show by the following formulae, the cover region $A(FM)$ is in direct proportion both to the area that the FM covers and to the number of points the FM overlaps.

$$A(P_k) \approx A(P_0) \approx S_a \Rightarrow A(FM) \approx m \cdot S_a \ . \quad \mathbf{(6)}$$

2.4 Fitting Accuracy Comparison of FM and Splats

When a plane patch is used as a surface primitive for point-sampled surfaces, we can define the sum of distances from all points to the plane as the standard error measurement. Here, we will give a simple comparison of the splat-based method and our FM method in the surface fitting accuracy by using the error measurement.

Assume the same neighbor selection method is used for our method and the splat simplification method proposed in [10]. Then the standard error of FM is different from that of splats even if they cover the same number of sample points. As illustrated in Fig. 4, the blue line L is a circle splat of P_0, and the two green lines are two triangles of FM we generated. The standard error from the left four points to L is (P_0P'+

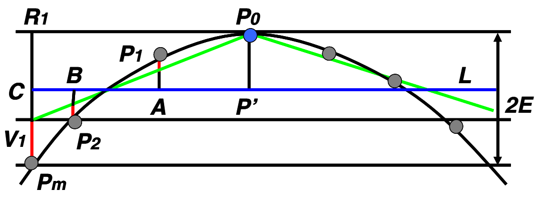

Fig. 4. Standard errors of splats and FM

P_1A+ P_2B+ P_mC), and the error of FM equals to the sum of lengths of the red line segments in Fig. 4 multiplied by $cos(\angle R_1P_0V_1)$, Obviously, the standard error of FM is smaller than that of splats, and the detailed comparison results are presented in Section 4.

3 Post-Processing of FMs and Rendering

Since there are many crossovers among FMs, we can select a subset of the FMs to get a further simplified description. We present two FM simplification algorithms with different purposes here. One aims to have a simple and fast rendering process with consideration on reducing the influence of noisy points at the data boundaries. The other is to get an approximately minimal subset of FMs by using a greedy searching method. The performances of the both methods are shown in the experimental results.

Fast rendering method. Given a sampled point cloud, we divide the points into two sets: points $\{P_c\}$ treated as centers of FMs and points $\{P_d\}$ that are covered by the FMs. For the input point cloud, we set indices for the points in the order of their *XYZ* coordinates. After that, we select the point P_m in the middle of the index queue and put it into $\{P_c\}$. An FM is generated for P_m and we put all points covered by this FM into $\{P_d\}$. The original point set is then divided into two small sets at the location of P_m with P_m and the covered points taken out. The above process is applied to the two sets iteratively until all the points are put into $\{P_c\}$ or $\{P_d\}$.

For a 3D scene with boundaries, the points in the middle zone of the index queue probably distribute in the center region of the scene. By handling the points from center to outside, the boundary points, which usually contain more noises, have little possibility of being selected as the center points of FMs. By doing so, effects of outliers at the boundaries are reduced.

Greedy searching method. The best strategy for the FM simplification is to get a minimal subset of FMs to cover all the sample points. However, it is a NP hard problem and we have to design an algorithm to find an approximation solution with much less costs. In this paper, we use a method based on the greedy searching strategy.

At first, we generate FM_i for every sampled point P_i. Then the areas of the center point $A(P_i)$ and the cover region $A(FM_i)$ are calculated by Eq.(**4**) and (**5**). The inter-coverage relationship is recorded as: 1) the points overlapped by FM_i; 2) the FMs that cover P_i. When the process begins, we select the FM with the maximum cover region and mark the points $\{P_k\}$ it covers as being used. The next step is to update the inter-coverage relationships of remaining FMs. For every point P_j in $\{P_k\}$ we find all the FMs covering it and cut $A(P_j)$ away from the covered regions. Then we repeat the selecting and updating steps until all points are marked as being used.

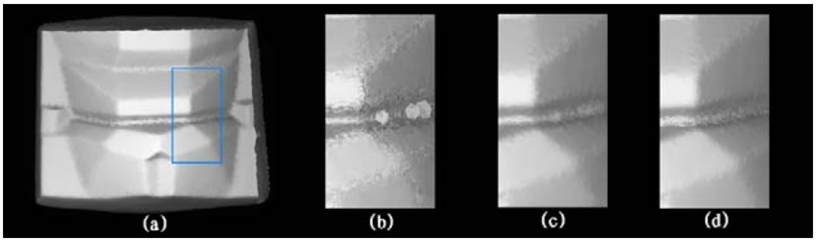

Fig. 5. (a): the simplified mouth model with 5,446 FMs and the region to be zoomed in; (b): fragmentations on the region; (c): the region rendered by the average method; (d): the region rendered by the weighted average method

We need some special techniques to visualize FM models or scenes as there are fragmentations among the FMs. At first, we select the FMs that are at the forefront surface with respect to the viewpoint. These FMs may yet be overlapped by other neighboring FMs, and hence in the corresponding pixel there may be more than one brightness value. Our idea is to use the average of the brightness values to render the pixel. To improve the visual effects, we can set a weight coefficient to every brightness value according to their depths to get a weighted average of the brightness values. The results of our rendering methods are shown in Fig.5.

4 Experimental Results

We apply our algorithm to some 3D models and scenes obtained by using laser scanner. Some performance data that are measured on a Pentium4 2.6GHz CPU with 512MB memory system are given in Table 1. The third column is the Error Tolerance (ET) with respect to the model's bounding box diagonal. The forth and fifth columns are the computing times of the rendering method (Time-R) and the numbers of FMs rendered (FM-R). The sixth and seventh columns are times used in the greedy searching method (Time-S) and the numbers of FMs after simplification (FM-S). The last two columns are the Average Standard Error (ASE) of models obtained by the original splat-based method and our FM method, respectively.

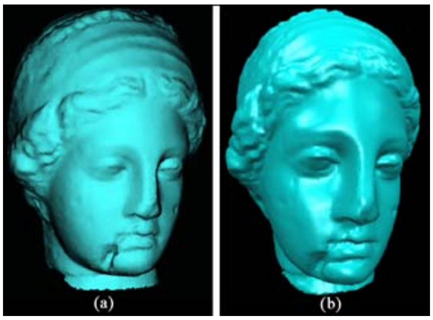

Fig. 6. Simplified Igea models with the same ASE. (a): 28,199 FMs in our method; (b): 29,847 splats in splatting method [10]

We notice that ASE of the splat-based method is bigger than ASE of the FM based method. Therefore, with the same ASE, we can use less FMs than splats to represent the model but get similar visual effects. The comparison is shown in Fig.6.

Table 1. The experimental results of our algorithm

Model	Points	ET (%)	Time-R	FM-R	Time-S	FM-S	ASE of Splats	ASE of FMs
Mouth	23,649	0.03	1.656	5,446	9.14	3,306	0.5325	0.1468
		0.08	1.047	3,021	10.30	1,760	2.3345	0.6791
		0.14	0.857	2,395	11.65	1,337	4.7422	1.3572
		0.56	0.813	1,375	15.38	562	32.930	9.0826
Horse	48,485	0.04	3.969	12,251	18.68	7,252	0.5094	0.1517
		0.12	2.484	6,923	20.45	4,394	2.5961	0.7807
		0.20	2.282	5,447	24.38	3,104	5.5823	1.6191
		0.80	2.000	3,187	32.49	1,394	36.286	9.6502
Desk	103,645	0.01	12.187	24,674	57.69	15,549	0.6276	0.1640
		0.05	7.516	11,213	73.45	5,832	6.2245	2.0355
Building	220,454	0.01	23.922	39,490	143.45	26,085	0.0233	0.0061
		0.02	17.578	28,117	157.81	16,501	0.0548	0.0162

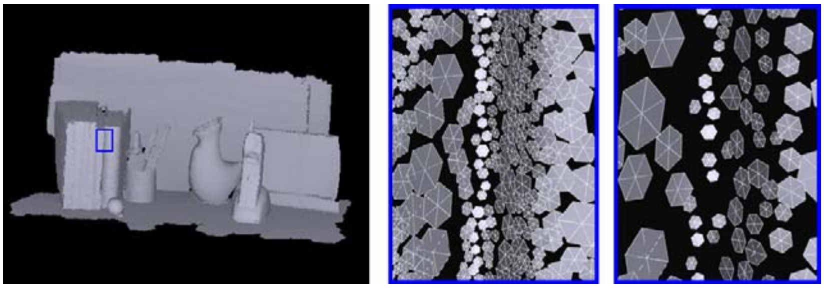
Fig. 7. Distribution of FMs

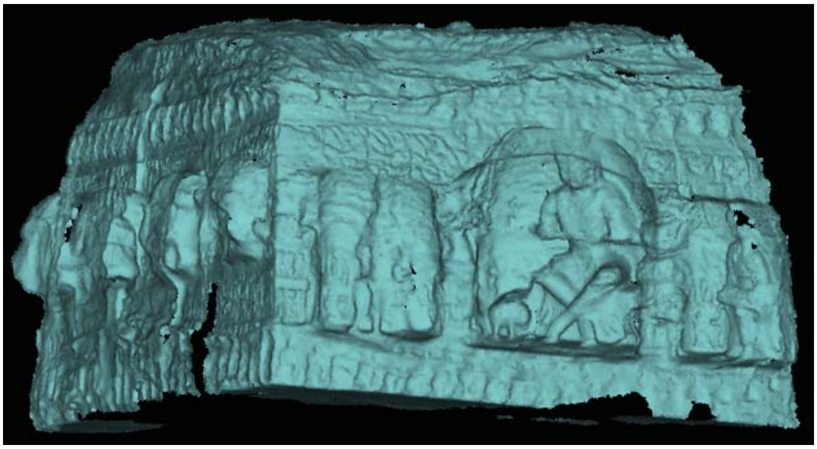
Fig. 8. The YunGang Grotto model with 302,864 FMs. The ASE of this model is 0.032%

Fig. 7 illustrates the distribution of FMs in the results. The left image shows a desk scene with 103645 points, and the blue box indicates the region to be zoomed in. The middle is the distribution of FMs resulting from the fast rendering method, and the right is the results of the greedy searching method. We find the used FMs are much more sparse but with more uniform distribution.

In order to testify the efficiency of our methods for huge models, we use a dataset of scanning points of the YunGang Grottos in China. It contains 2,981,168 points after registration and it is too large to be held in main memories. Therefore, we cut it into several blocks and perform the simplification on each block. Then we align them together without any special processing for the piece stitching. The whole model is illustrated in Fig. 8, and the overall running time is less than half an hour.

7 Conclusions

In this paper, we proposed an efficient FM-based simplification and rendering algorithm for geometrical description of 3D models or scenes. With our re-sampling operations on FMs, we can decrease the number of the needed FMs greatly, which will improve interactive rendering and transmission performance in augmented reality applications. Since our method is designed based on local geometrical information, it is convenient to perform partial editing and processing on models. Moreover, data structures defined here are easy to integrate into level-of-detail techniques.

Acknowledgement

The work was supported in part by the NSFC Grant (No. 6033010) and NKBRPC (No. 2004CB318000).

References

1. Hu, J., You, S. and Neumann, U., “Approaches to Large-Scale Urban Modeling”, *IEEE Computer Graphics and Applications,* Volume 23, Issue 6, 2003, pp. 62-69.
2. Cignoni, P., Montani, C., Rocchini, C., etc., “External Memory Management and Simplification of Huge Meshes”, *Visualization and Computer Graphics,* Volume 9, Issue 4, 2003, pp. 525-537.
3. Kobbelt, L. and Botsch, M., “A Survey of Point-Based Techniques in Computer Graphics”, *Computers & Graphics,* Volume 28, Issue 6, 2004, pp. 801-814.
4. Meng, F. and Zha, H., “An Easy Viewer for Out-of-core Visualization of Huge Point-Sampled Models”, *Proc. 2nd Intel. Sym. on 3DPVT,* 2004, pp. 207-214.
5. Pauly, M., Gross, M. and Kobbelt, L., “Efficient Simplification of Point-Sampled Surfaces”, *Proc. IEEE Visualization,* 2002, pp.163-170.
6. Pajarola, R., “Efficient Level-of-Detail for Point Based Rendering”, *Proc. IASTED Computer Graphics and Imaging,* 2003.
7. Meng, F. and Zha, H., “Streaming Transmission of Point-Sampled Geometry Based on View-Dependent Level-of-Detail”, *Proc. Fourth Intl. Conf. on 3DIM,* 2003, pp.466-473.
8. Kalaiah, A., and Varshney, A., “Modeling and Rendering of Points with Local Geometry”, *IEEE Trans. on Visualization and Computer Graphics*, 2002, pp.100-128.
9. Yan, X., Meng, F. and Zha, H., “Fan-Meshes: A Geometric Primitive for Point-based Description of 3D Models and Scenes”, *Proc. 2nd Intel. Sym. on 3DPVT*, 2004, pp. 207-214.
10. Wu, J. and Kobbelt, L., “Optimized Sub-Sampling of Point Sets for Surface Splatting”, *Computer Graphics Forum (Eurographics 2004),* Volume 23, Issue 3, 2004, pp. 643-652.