Journal of Southeast University (English Edition) Vol. 20 No. 3 Sept. 2004 ISSN 1003—7985

Multiresolution analysis over triangle meshes: method and data structure

Tang Jie Zhang Fuyan

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

Abstract: A robust and efficient algorithm is presented to build mulitiresolution models (MRMs) of arbitrary meshes without requirement of subdivision connectivity. To overcome the sampling difficulty of arbitrary meshes, edge contraction and vertex expansion are used as downsampling and upsampling methods. Our MRMs of a mesh are composed of a base mesh and a series of edge split operations, which are organized as a directed graph. Each split operation encodes two parts of information. One is the modification to the mesh, and the other is the dependency relation among splits. Such organization ensures the efficiency and robustness of our MRM algorithm. Examples demonstrate the functionality of our method.

Key words: mesh simplification; CAD/CAM; multiresolution model; geometric modeling

Triangle meshes and piecewise parametric patches (e.g. trimmed non-uniform rational B-spline (NURBS) surfaces) are two of the most popular choices for surface representations. Compared with NURBS patches, triangle meshes have advantages in representing objects with arbitrary shapes and detail levels, as well as being computationally efficient and robust. Because of these advantages, triangle meshes are widely used in many applications such as computer animation, visualization, robotics, tool path generation, reverse engineering, and rapid prototyping manufacture.

However, with the improving ability of scanning instruments, meshes generated from a digitized set of points usually have a large number of vertices and faces. It is inefficient and unnecessary to use these unprocessed models directly in all cases and sometimes it is even impossible to utilize such meshes directly, for example browsing a 3-D mesh through the Internet with limited bandwidth. Thus, operations such as mesh simplification[1-5] and progressive mesh[6] appeared, which partly solved the problem.

Multiresolution analysis[7,8] (MRA) is a powerful tool of signal process. In mesh processing, MRA means to create representation of a mesh at different levels of detail. A multiresolution model (MRM) of mesh is represented by a base mesh plus a collection of mesh modifications which possess some relations between each of them. MRMs of meshes are convenient for a number of applications, including compression/simplification, progressive display and transmission, level-of-detail control, multiresolution editing, etc.

Sampling is a base operation for creating MRMs of meshes. But unfortunately, unlike regularly sampled data, such as waves or images, of which the downsampling and upsampling method are straightforward, triangle meshes with arbitrary connectivity form an inherently irregular sampling setting. Therefore, new algorithms need to be developed.

Data structure is the other pivotal factor for MRMs to be successfully employed. It should have a low overhead with respect to storing just the mesh at the full resolution and support efficiently algorithms for selective refinement, as well as algorithms for performing spatial selection, such as a point location, a window, or a range query.

In this paper, we present a robust and efficient algorithm to build MRMs of arbitrary meshes without requirement of subdivision connectivity. A new mesh simplification method is developed to overcome the sampling difficulty of arbitrary meshes. Edge contraction and vertex expansion are used as downsampling and upsampling methods, respectively. Our MRMs of a mesh are composed of a base mesh and a series of edge split operations. With such organization of MRMs, refining, coarsening and selective refining of a mesh are allowed to be executed more easily and interactively.

1 Sampling

Sampling of arbitrary mesh is difficult because meshes are usually defined on a 2-mainfold. The connectivity of a mesh is much more complicated than that of a wave or an image. In order to simplify both

Received 2004-01-13.

Biographies: Tang Jie (1971—), male, doctor, jietang@graphics.nju.edu.cn; Zhang Fuyan (1939—), male, professor, fyzhang@graphics.nju.edu.cn.

geometry and connectivity through downsampling, mesh simplification is the best choice. The reverse operation of mesh simplification, mesh expansion, is therefore used as upsampling.

However mesh simplification should be executed carefully so that smooth sampling results can be reached. In this paper, we present a new mesh simplification algorithm based on consecutive edge contraction (see Fig.1). To avoid the accumulation of each contraction step's error, the simplification algorithm should compute the contraction cost using the information of original mesh. To do this, we select some sample points from the original model, and distribute them into each triangle. Throughout the simplification, we keep these points and use them to calculate the cost of a contraction. If a triangle is deleted, the sample points attached to it will be redistributed to other relative triangles. Thus the original information is kept. We use L_∞ error to evaluate the approximation of the simplification. But our method is a little different from Garland's[4]. While computing the approximation, Garland uses the distance from a point to the plane in which the triangle lies, no matter whether the projection of the point to this plane falls into the triangle or not. This causes the disturbance of the error evaluation and affects the simplification quality. To overcome this, we use the distance from a point to the triangle instead of the plane to calculate the approximation, which will be explained later in detail.

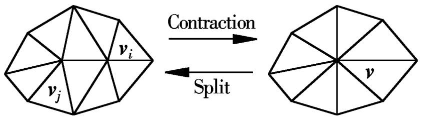


Fig.1 Edge contraction and vertex split

The outline of the algorithm is as follows:

1) Select a set of candidate vertex pairs.

2) Select a set of sample points and distribute them to each triangle.

3) Calculate a cost of contraction to each candidate pair.

4) Consistency checking.

5) Place all candidates in a queue keyed to the cost with the minimum cost pair at the top.

6) Repeat until the desired approximation is reached:

① Remove the pair ($\boldsymbol{v}_i$, $\boldsymbol{v}_j$) of the least cost from the heap;

② Contract this pair;

③ Redistribute the sample points of deleted triangles;

④ Update the costs of all candidate pairs involving $\boldsymbol{v}$.

1.1 Candidate pairs

There are two ways to select candidate pairs. The first one simply selects all edges of M_0 as candidate pairs. This is the most natural and efficient way. The second one selects not only all edges of M_0 but also some non-edge pairs of which the length is less than a threshold. This selection is beneficial if the original model has more than one connected part, because contraction of a non-edge pair can bring unconnected parts into one part, thus producing better approximation. However, contraction of non-edge pairs can also produce a non-manifold surface. Moreover, identifying all suitable non-edge pairs is a time-consuming job though many methods have been proposed. For our algorithm, we use the first choice to prepare candidate pairs.

1.2 Sample points

Sample points are used to calculate the contraction cost, which means the approximation of a simplified mesh to the original model. Therefore they should catch as many details of the original model as they can. More sample points usually bring out better simplifying quality, but the tradeoff is the decreasing of the computing speed. With these considerations, we use a self-adaptive way to select sample points. That is, for an original mesh $M_0 = (K, V)$, firstly, all vertices from V are selected as sample points; secondly, in those areas with sharp features, more sample points should be selected. If the dihedral angle of an edge is greater than a user-specified value, the edge is defined as a feature edge. Another case of feature edge is boundary edge, which only has one neighboring triangle. All feature edges form a set E_f. We select additional sample points as follows:

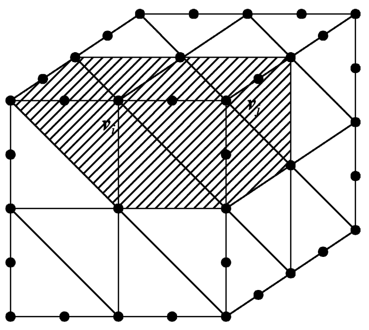
$$V_\mathrm{f} = \left\{ \boldsymbol{v} \mid \boldsymbol{v} = \frac{1}{2}(\boldsymbol{v}_i + \boldsymbol{v}_j),\ \{i, j\} \in E_\mathrm{f} \right\} \tag{1}$$

The final sample points are

$$V_\mathrm{s} = V \cup V_\mathrm{f} \tag{2}$$

After sampling, all sample points are distributed to the triangles on which they lie (see Fig.2(a)). While computing the contraction cost, if an edge ($\boldsymbol{v}_i\boldsymbol{v}_j$) is deleted, only its 1-ring neighbor area (shaded area in Fig.2) is modified. Therefore, the contraction cost, i.e., the error caused by this edge contraction, can be computed locally by using the sample points of the involved triangles.

If a triangle is deleted, its sample points will be redistributed to relatively live triangles, which will be discussed later. This makes sure that each contraction


(a)

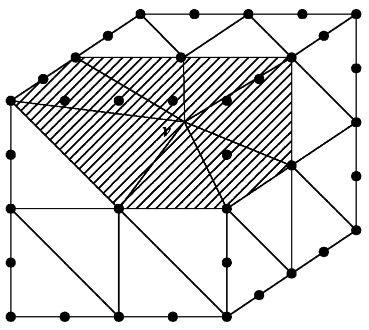

(b)

Fig.2 Sample points. (a) Before contraction; (b) After contraction

is computed based on the information from the original model.

1.3 Contraction cost

An edge contraction deletes an edge and its two neighbor faces (one if it is a boundary edge), thus changes the surface of the model (see Fig. 2). There are two factors that decide the cost of an edge contraction. One is how to place the target vertex; the other is error evaluation policy.

1.3.1 Vertex placement

After an edge was deleted, we need some way to place the target position of $\boldsymbol{v}$. How to place the target position decided how the shape of modified area changed into. When choosing the target vertex placement, we must trade space and time efficiency against the approximation quality.

Subset placement is the simplest strategy, which is quite efficient. We simply select one of the endpoints as the target position. To choose it between endpoints, we merely find the vertex which has smaller contraction error. Under this policy, an approximation will use a subset of the original vertices in their original positions. Subset placement is more efficient when we create the MRM or progressive mesh, since it can significantly save storage. This becomes even more important when meshes possess material properties.

However, when an edge is collapsed, subset placement contracts the model volume in convex regions and inflates it in concave regions. In order to preserve the model volume better, we develop a new method called volume optimal placement.

Fig.3 shows the volume change after an edge $\boldsymbol{v}_i\boldsymbol{v}_j$ was contracted. $\boldsymbol{v}$ is the new vertex. From Fig. 3 we can see that each triangle in the 1-ring neighborhood of $\boldsymbol{v}_i\boldsymbol{v}_j$ and the new vertex $\boldsymbol{v}$ form into a tetrahedron. The model volume change is made up of all these tetrahedrons. For each tetrahedron, the volume V_i is

$$V_i = \frac{1}{6}\boldsymbol{n}_i^{\mathrm{T}}(\boldsymbol{v} - \boldsymbol{v}_0) = \frac{1}{6}(\boldsymbol{n}_i^{\mathrm{T}}\boldsymbol{v} + d) \tag{3}$$

where $\boldsymbol{n}_i$ is the normal vector of each triangle, $\boldsymbol{v}$ is the new vertex, $\boldsymbol{v}_0$ is a vertex of each triangle, and $d = -\boldsymbol{n}_i^{\mathrm{T}}\boldsymbol{v}_0$.

Discarding the magnitude coefficients, the squared volume of a tetrahedron is given by the following equation:

$$\begin{aligned} V_i^2 &= (\boldsymbol{n}_i^{\mathrm{T}}\boldsymbol{v} + d)^2 = (\boldsymbol{v}^{\mathrm{T}}\boldsymbol{n}_i + d)(\boldsymbol{n}_i^{\mathrm{T}}\boldsymbol{v} + d) = \\ &\boldsymbol{v}^{\mathrm{T}}\boldsymbol{n}_i\boldsymbol{n}_i^{\mathrm{T}}\boldsymbol{v} + 2(d\boldsymbol{n}_i)^{\mathrm{T}}\boldsymbol{v} + d^2 = \\ &\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{v} + 2\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{v} + c_i \end{aligned} \tag{4}$$

The summation of squared volume change is

$$\begin{aligned} f(\boldsymbol{v}) &= \sum V_i^2 = \sum(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{v} + 2\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{v} + c_i) = \\ &\boldsymbol{v}^{\mathrm{T}}\sum\boldsymbol{A}_i\boldsymbol{v} + 2\sum\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{v} + \sum c_i = \\ &\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v} + 2\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} + c \end{aligned} \tag{5}$$

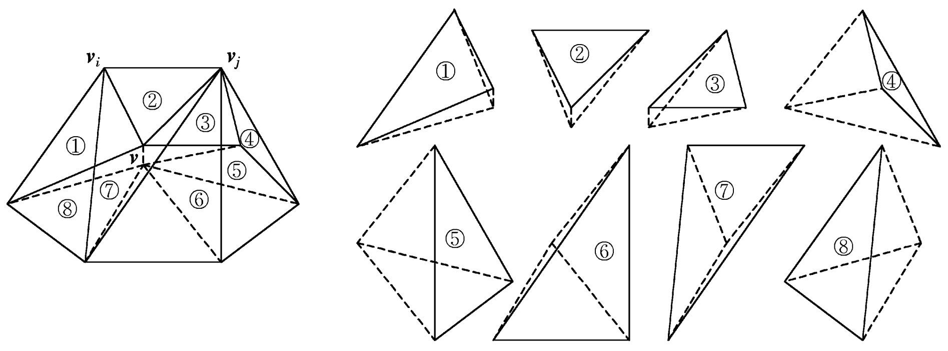


Fig.3 Volume change caused by an edge contraction

In order to best preserve the model volume during the simplification, we want to place the new vertex $\boldsymbol{v}$ so that the volume change caused by an edge contraction is minimized. Since $f(\boldsymbol{v})$ is quadratic, finding its minimum is a linear problem. Taking partial derivatives, we can see that the gradient of $f(\boldsymbol{v})$ is

$$\nabla f(\boldsymbol{v}) = 2\boldsymbol{A}\boldsymbol{v} + 2\boldsymbol{b} \tag{6}$$

Solving for $\nabla f(\boldsymbol{v}) = 0$, we find the optimal position is

$$\boldsymbol{v} = -\boldsymbol{A}^{-1}\boldsymbol{b} \tag{7}$$

Sometimes, a unique optimal position may not exist. If $\boldsymbol{A}$ is singular, its inverse does not exist, and we cannot solve for $\boldsymbol{v}$ using Eq.(7). In this situation, subset placement is used.

1.3.2 Error evaluation

How to evaluate the change is very important to a simplification algorithm. L_2 error evaluation is an accumulation of errors, thus it is dependent on the triangulation and sampling method. The L_2 errors of those areas with more sample points are greater than those with fewer sample points. L_∞ error does not have this disadvantage. Theoretically, Hausdorff distance is used to describe the L_∞ error measurement between two meshes. In practice, however, this error metric can be prohibitively expensive to compute. Some localized measurements must be found to save both time and memory. From Fig.3, we can see that the changed area after an edge contraction is the union of the 1-ring area of the 2 end points. Therefore we find a localized way to calculate the L_∞ error of an edge contraction and use this error as the contraction cost.

1) Distance from a point to a triangle

Given a point $\boldsymbol{v}$ and a triangle $\boldsymbol{v}_0\boldsymbol{v}_1\boldsymbol{v}_2$ in 3-D Euclidean space (see Fig.4(a)), let $\boldsymbol{v}'$ be the projection of $\boldsymbol{v}$ on the triangle, $\boldsymbol{n}$ be the normal vector of the triangle, d be the signed distance between $\boldsymbol{v}$ and $\boldsymbol{v}'$, then $\boldsymbol{v}'$ can be written as

$$\boldsymbol{v}' = \boldsymbol{v} + d\boldsymbol{n} \tag{8}$$

Because $\boldsymbol{v}'$ is on the triangle $\boldsymbol{v}_0\boldsymbol{v}_1\boldsymbol{v}_2$, we can get

$$\boldsymbol{v}' = \boldsymbol{v} + d\boldsymbol{n} = a\boldsymbol{v}_0 + b\boldsymbol{v}_1 + c\boldsymbol{v}_2 \tag{9}$$

$$a + b + c = 1 \tag{10}$$

Eq.(9) has three scalar equations, together with Eq.(10), the values of d, a, b, c can be calculated. According to the values of a, b and c, the distance from $\boldsymbol{v}$ to the triangle D_t can be computed as follows:

① $a \geqslant 0$, $b \geqslant 0$, $c \geqslant 0$ (see Fig.4(a)), $D_t = |d|$;

② There is one value less than 0, say $a < 0$ (see Fig.4 (b)), $D_t = \sqrt{d^2 + |\boldsymbol{v}'\boldsymbol{v}''|^2}$;

③ There are two values less than 0, say $a < 0$, $c < 0$ (see Fig.4(c)), $D_t = \sqrt{d^2 + |\boldsymbol{v}'\boldsymbol{v}_1|^2}$.

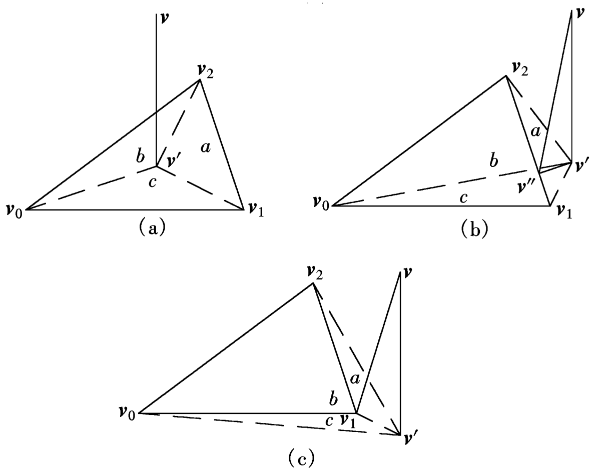


Fig.4 Distance from a point to a triangle

2) Cost of an edge contraction

Let V_s be all sample points belonging to the 1-ring triangles of $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, F_{ij} be the 1-ring triangles of the target vertex $\boldsymbol{v}'$, then the cost of edge contraction of edge $\boldsymbol{v}_i\boldsymbol{v}_j$ is as follows:

$$c_{ij} = \max_{\boldsymbol{v} \in V_s}\left(\min_{T \in F_{ij}}(D_t(\boldsymbol{v}, T))\right) \tag{11}$$

where $D_t(\boldsymbol{v}, T)$ is the distance from a point $\boldsymbol{v}$ to a triangle T.

1.4 Repeating operations

When all candidates have been sorted into a queue keyed on the cost with the minimum cost pair at the top, we can start to repeatedly contract the candidate pairs until the desired approximation is reached. After each contraction, two operations must be done, which are the recalculation of the cost of pairs involving modified area and the redistribution of sample points.

1.4.1 Recalculating contraction cost

An edge contraction will delete and modify several triangles. The costs and consistencies of some candidates are changed. Therefore recomputing the costs and the consistencies must be done as well as the resorting of those modified candidates. Most algorithms suggest that only the 1-ring edges of the target vertex $\boldsymbol{v}$ need to be recalculated. However, from experiments, we find that this is not strong enough to prevent the folding over of triangles. At least 2-ring neighboring edges should be recalculated.

1.4.2 Redistribution sample points

Each contraction will delete one or two triangles and modify several other triangles. If we do not redistribute those sample points belonging to these triangles, the information of the original mesh will be lost. Therefore, redistribution of sample points can ensure that each contraction is recomputed using the information of the original model. The operation of redistribution is carried on while computing the costs of contractions:

1) Collect all sample points belonging to the union of the 1-ring triangles of $\boldsymbol{v}_i\boldsymbol{v}_j$.

2) According to the values of a, b, c (section 1.3.2):

① If $a \geqslant 0$, $b \geqslant 0$, $c \geqslant 0$ (see Fig.4(a)), assign the sample point to this triangle only;

② If there is one value less than 0, say $a < 0$ (see Fig.4(b)), assign the sample point to the neighboring triangles of this edge;

③ If there are two values less than 0, say $a < 0$, $c < 0$ (see Fig.4(c)), assign the sample point to the neighboring triangles of this vertex.

2 Multiresolution Model

After finishing the sampling process, we can create the MRM for meshes. Data structures for multiresolution meshes must represent, either explicitly or implicitly, the modifications and the dependency relations between these modifications. For this purpose, data structures of MRM are usually organized into encoding modifications and encoding dependency relations. Our method encodes the modifications not as the new generated vertices, but as real modification. And based on that, we encode the dependency relations as relations between these modifications.

2.1 Encoding modifications

Encoding of modifications should provide sufficient information to perform the tasks of refining and coarsening mesh, which are necessary to the implemention of multiresolution mesh.

For a modification m which is based on edge contraction, our data structure is designed as follows:

```
struct Modification{
Vertex*          vertex[2];
float            DeltaPosition1[3];
float            DeltaPosition2[3];
Triangle*        pModifiedTriangle;
byte             Pivot;
Triangle*        pDeadTriangle;
Modification*    pPreModifications;
Modification*    pPostModifications;
byte             bSplit;
float            error;
}
```

All pre-modifications of m are generated before m during the initial downsampling process. All modifications are generated in the order of global error with the smallest being generated first. Therefore, if m satisfies the error specification, so do all its ancestors. Procedure contraction mesh is defined analogously.

2.2 Encoding dependency

Encoding dependency is more complicated than encoding modification. A robust dependency relation should ensure that all triangles that appeared while an MRM mesh was changing its resolution are among those triangles that appeared during the original simplification sequence. This can further prevent unexpected mesh from appearing, which may have such flaws as slim triangles and fold-up triangles.

Our dependency relations are defined on edge contraction. And its base rule is that a vertex $\boldsymbol{v}$ can be split only if all its adjacent vertices have been split if they have a split. Fig.5 shows a sequence of edge collapse of a mesh. During the simplifying of mesh, the dependency relations are generated. Each vertex is attached a split operation, if it is an original vertex, its value is null. First, edge 11-12 is collapsed. And a split operation S_1 is generated and attached to the target vertex 11. Vertex 12 is inactivated. Since the adjacent vertices of the new vertex 11 are all original vertices, no dependency relations generated. Second, edge 4-5 is collapsed, a split operation S_2 is generated and attached to the target vertex 4. Vertex 5 is inactivated. This time, since vertex 11 is adjacent to vertex 4, the split operation S_1 is therefore dependent on split operation S_2. Repeating this procedure until the simplification is over and all dependency relations are created.

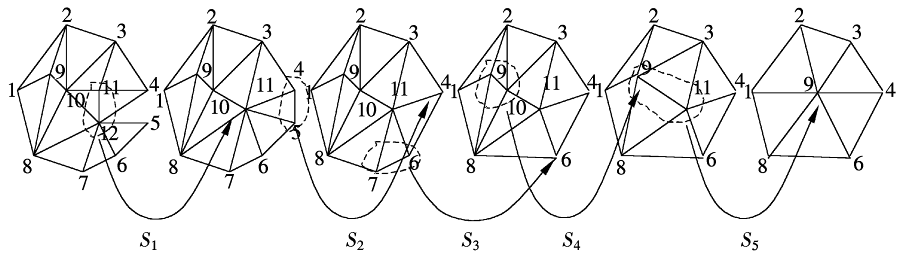


Fig.5 Evolution of a mesh through a sequence of edge collapsed

We find that the dependency relations are a directed graph (see Fig.6). In this way, all splits are sorted in partial order. Therefore, topological sorting can be used to deduce the dependent sequence of splits operation of a split. For example, the dependent sequence of S_1 is $S_5 \rightarrow S_4 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1$. Of course, more than one sequence is right, $S_5 \rightarrow S_3 \rightarrow S_2 \rightarrow S_4 \rightarrow S_1$ is also an appropriate choice. Our method can capture all dependency relations of the MRM. Thus can ensure the robustness of resulted multiresolution mesh.

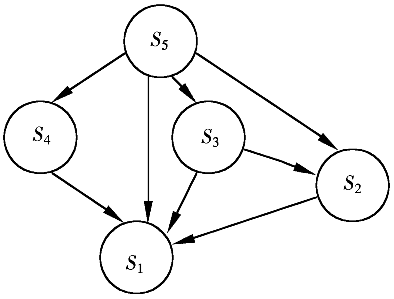


Fig.6 Directed graph of dependency relations

3 Results

We tested our algorithm using many models, two of which are the famous bunny from Stanford University and an oil pump model. The bunny is selected because it contains several regions with boundaries on its underside, so that we can test our method's effectivity when dealing with boundaries.

3.1 Multiresolution

Multiresolution mesh can be realized straightforwardly using our method. Our MRM mesh is composed of a base mesh and a sequence of splits. All these splits are sorted so that the modification to the model during the change of resolution can be done in the smoothest way. The current level of the model is represented as i, which means that the first i splits have been executed. Fig. 7 shows the result of the bunny. Fig.7(a) is the model at 5% of full resolution with the maximum error of 5.580×10^{-3}, Fig.7(b) is the model at 30% of full resolution with the maximum error of 1.174×10^{-3}, Fig.7(c) is the full resolution model.

3.2 Selective refinement

Selective refinement is widely used in many applications, such as view-dependent refinement. The algorithm of selective refinement is also based on the algorithm of multiresolution. During the generating of MRM, each split is attached a box value, which is the outer bound box of the 1-ring area of the target vertex. When executing selective refinement, all splits that fall into the select area are executed. The important thing is that if a split is going to be executed, its dependent splits should be executed first. Fig.8 shows the selective refinement results of the oil pump head. Fig.8(a) is the model at 50% of full resolution with the maximum error of 4.749×10^{-3}, Fig.8(b) is the selective refinement of pump, Fig.8(c) is the full resolution model.

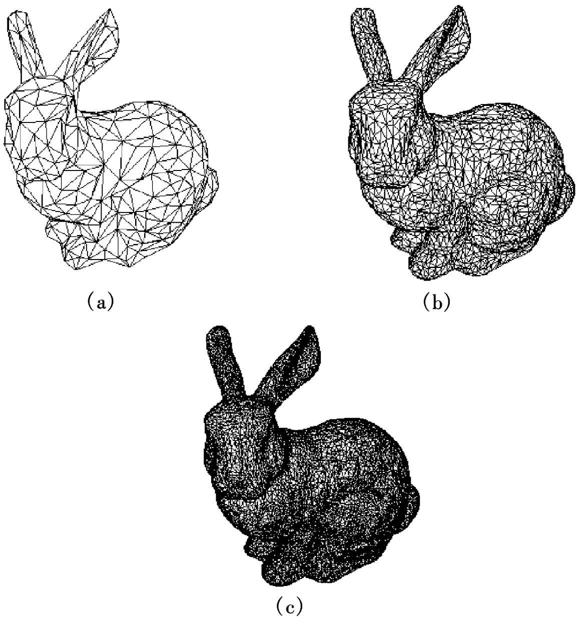


Fig.7 Multiresolution model of bunny

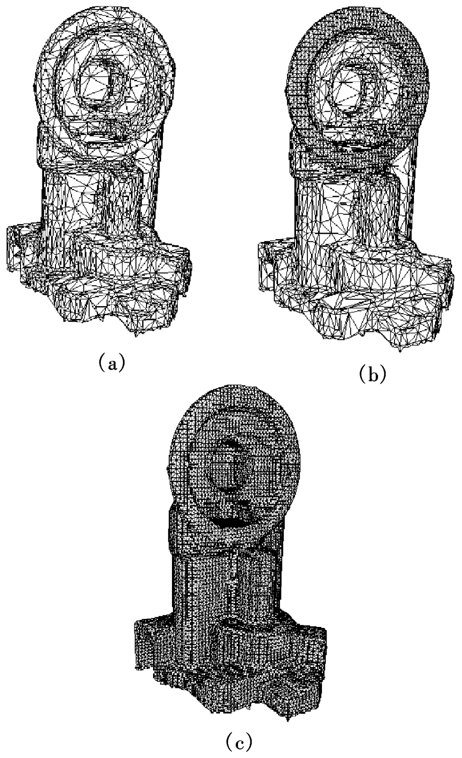


Fig.8 Multiresolution model of oil pump

4 Conclusion

We have introduced an efficient framework for generating an MRM of arbitrary meshes without the requiring of subdivision connectivity. A new mesh simplification algorithm based on global error control is used as sampling method. The MRM is composed of a base mesh plus a sequence of edge split operations, which are organized as a directed graph and sorted in the smoothest way. Each split operation encodes two parts of information, the modification to the mesh and the dependency relations between these splits. Using this MRM representing can easily realize the refining, coarsening and selective refining of arbitrary meshes.

Possible future research includes: ① Memory management for large scale models; ② Color and texture MRM; ③ Error control in selective refinement for mechanical utilization.

References

[1] Schroeder W J, Jonathan A Z, William L. Decimation of triangle meshes[A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*[C]. Chicago, Illinois, 1992. 65 – 70.

[2] Turk Greg. Re-tiling polygonal surfaces[A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. Chicago, Illinois, 1992. 55 – 64.

[3] Hoppe H, DeRose T, Duchamp T, et al. Mesh optimization [A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. Anaheim, California, 1993. 19 – 26.

[4] Garland M, Heckbert P. Surface simplification using quadric error metrics [A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. Los Angeles, California, 1997. 209 – 216.

[5] Kim S J, Kim C H, Levin D. Surface simplification using a discrete curvature norm [J]. *Computer & Graphics*, 2002, **26**(5): 657 – 663.

[6] Hoppe H. Progressive meshes [A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. New Orleans, Louisiana, 1996. 99 – 108.

[7] Eck M, DeRose T, Duchamp T, et al. Multiresolution analysis of arbitrary meshes[A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. Los Angeles, California, 1995. 173 – 182.

[8] Guskov I, Sweldens W, Schroeder P, et al. Multiresolution signal processing for meshes[A]. In: *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH* [C]. Los Angeles, California, 1999. 325 – 334.

三角网格模型的多分辨分析: 方法和数据结构

唐 杰　　张福炎

(南京大学软件新技术国家重点实验室, 南京 210093)

摘要: 提出了一个健壮有效的网格模型多分辨分析方法. 该方法面向任意网格模型且不需要具有子分连通性, 通过删除边和拆分点操作进行网格模型的向下采样和向上采样, 将网格模型表示为由一个低分辨率的网格和一系列修改操作组成的多分辨模型. 该算法在向下采样时, 重点考虑了简化误差对模型精度的影响, 在生成网格多分辨模型时, 将细化操作分解为对网格模型的几何修改信息和各细化操作之间的关系信息, 确保了多分辨网格模型的健壮性. 实验结果证明了本算法的有效性.

关键词: 网格简化; CAD/CAM; 多分辨模型; 几何造型

中图分类号: TP391