Volume 20 (2001), number 2 pp. 115–123 COMPUTER GRAPHICS *forum*

Constrained Fairing for Meshes

Xinguo Liu[1], Hujun Bao[1], PhengAnn Heng[2], TienTsin Wong[2] and Qunsheng Peng[1]

[1]State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou, 310027, People's Republic of China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong, People's Republic of China

Abstract

In this paper, we present a novel fairing algorithm for the removal of noise from uniform triangular meshes without shrinkage and serious distortion. The key feature of this algorithm is to keep all triangle centers invariant at each smoothing step by including some constraints in the energy minimization functional. The constrained functional is then minimized efficiently using an iterative method. Further, we apply this smoothing technique to a multiresolution representation to remove arbitrary levels of detail. A volume-preserving decimation algorithm is presented to generate the multiresolution representation. The experimental results demonstrate the combined algorithm's stability and efficiency.

Keywords: curves & surfaces, geometric modeling, level of detail, algorithms, mesh generation

1. Introduction

Polygonal meshes have been widely used in computer aided design and the computer graphics community because of their simplicity and powerful ability to model complex shapes. Current 3D scanning technology enables us to conveniently capture detailed and dense sample points of object surfaces. These samples are then organized as a triangular mesh for later use [1–5]. However, the initial reconstructed models are usually noisy. Therefore, mesh processing algorithms, such as denoise, editing, simplification, etc. are usually necessary for further refining of the reconstructed model.

Recently multiresolution methods have become a popular technique in addressing these issues. Apart from the classical multiresolution analysis and subdivision techniques [6–8], many excellent mesh simplification algorithms [9–15] have been proposed to construct the multiresolution representation of meshes without subdivision connectivity. The challenge is how to process the dense meshes efficiently and generate meshes of high quality.

As multiresolution methods, the filtering technique is a common fairing method to smooth the triangular meshes [8,16–23]. The basic idea is to apply the concept of continuous energy minimization in CAGD [21,22] to smooth the triangular meshes discretely. With a special parametrization, the Laplacian algorithm, which is well known in image processing, can be extended to fair 3D meshes [8,16]. A local iterative procedure is then used to solve the global system. Unfortunately, due to the irregular connectivity, the resulting meshes may be dramatically shrunk and distorted. Moreover, the iteration is sometimes numerically unstable, which seriously slows down the convergence.

In this paper, a constraint of keeping the positions of triangle centers unchanged during the fairing process is introduced into the discrete energy functional of the mesh to be faired. The discrete fairing is then converted to a constrained minimization, which can be solved using a fast and robust iterative method. Our approach efficiently prevents the original meshes from shrinkage and serious distortion. To control the degree of smoothing, a novel volume-preserving decimation algorithm is presented to generate the multiresolution representation, which automatically preserves the features of original meshes. Finally, we apply this smoothing technique to the multiresolution representation to selectively remove different levels of details from meshes.

2. Related Work

As mentioned before, fairing algorithms smooth meshes by minimizing the energy functional. The most common energy

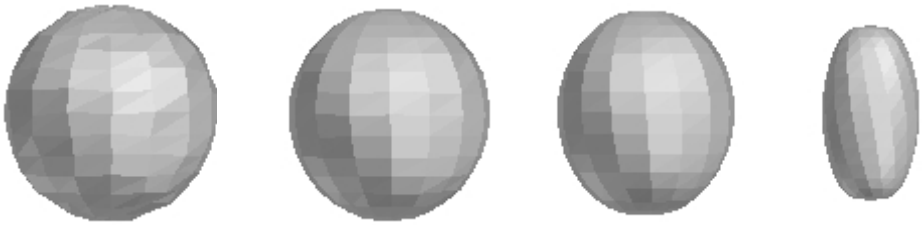


Figure 1: *A sphere smoothed by standard Laplacian method.*

functional are the membrane energy and thin-plate energy of a parametric surface $S : X = X(u, v)$ [21,22]:

$$E_{\text{memb}}(S) = \frac{1}{2}\int_S (X_u^2 + X_v^2)\,\mathrm{d}u\,\mathrm{d}v$$
$$E_{\text{thin}}(S) = \frac{1}{2}\int_S (X_{uu}^2 + 2X_{uv}^2 + X_{vv}^2)\,\mathrm{d}u\,\mathrm{d}v.$$

With a special parametrization, their variational derivatives correspond to the Laplacian and the second Laplacian respectively can be expressed as [8,16,19]:

$$L(X) = X_{uu} + X_{vv}$$
$$L^2(X) = L \circ L(X) = X_{uuuu} + 2X_{uuvv} + X_{vvvv}. \quad (1)$$

Since the application of standard Laplacian algorithm to mesh fairing may result in serious shrinkage and distortion (see Figure 1), in this paper we try to introduce constraints to the standard Laplacian algorithm to prevent the undesired distortion of the mesh. This idea is inspired by the work on modification of the Laplacian operator. The fairing algorithm for irregular meshes is first proposed by Taubin [16], in which he adopted two derivatives as well as an explicit iterative procedure to fair the meshes and minimize the shrinkage. The resulting meshes, however, heavily depend on the choice of the two mesh-related constants. Recently, Vollmer *et al.* [17] present an improved Laplacian smoothing method to attenuate the shrinkage, whose basic idea is to move the vertices of the smoothed mesh back towards their previous locations by some distance. Desbrun *et al.* [18] presented an implicit fairing approach to smooth meshes more efficiently and stably. A scale-dependent Laplacian operator and a constraint of volume preservation are used to reduce the degree of shrinkage and distortion. They also introduced a curvature flow operator in discrete differential geometry to remove small scale of details and prevent the distortion.

Another stream of fairing techniques aim at fairing meshes in multiple resolutions. Zorin *et al.* [6] combined subdivision and Taubin's fairing algorithm to construct a multiresolution editing algorithm for irregular meshes. Kobbelt *et al.* [8] described an excellent connectivity-based discrete Laplacian operator to efficiently remove different levels of details on progressive meshes. Based on their smoothing technique, they proposed an editing algorithm for irregular meshes, in which users may modify the shape of any user-specified local region. By generalizing the concept of Taubin's signal processing, Guskov *et al.* [23] considered both the connectivity and the geometry of meshes in their smoothing scheme to achieve better fairing of irregular meshes. By combining the mentioned algorithms, they also demonstrated several applications such as multiresolution fairing, editing.

3. Constrained Fairing

Before describing the details of our approach, we first define some notations of triangular meshes. In the following, a triangular mesh M is represented as a triple $\langle V, T, X\rangle$, where $V = \{1, 2, \ldots, N\}$ is its vertex index set, $T \subseteq 2^V$ is its triangle set composed of all triangles in the mesh. Each triangle $t \in T$ is defined as an ordered vertex index triple $t = \langle i, j, k\rangle$. The geometric realization of the mesh $X : V \to R^3$ is a mapping from the vertex indices to their locations in 3D space. Let $S(i) \subseteq V$ be the set of the 1-ring neighboring vertices of vertex i, and n_i be its valence (i.e. the number of its 1-ring neighbors). For short hand, we use x_i to replace $X(i)$ to describe the position of vertex i, so the geometry of M can be defined as a vector X consisting of all its vertices, namely, $X = [x_1, x_2 \cdots x_N]^{\mathrm{T}}$.

3.1. Constrained fairing equation

Practically, the Laplacian algorithm uses the following discrete functional to approximate the membrane energy:

$$E(M) = \|L(X)\|^2 = \sum_{i\in V} |L(x_i)|^2 \quad (2)$$

where $L(x_i) = \sum_{j\in S(i)} \omega_{ij}(x_j - x_i)$, ω_{ij} are the non-negative weights satisfying $\sum_{j\in S(i)} \omega_{ij} = 1$, and $\omega_{ij} = 0$ when $j \notin S(i)$. Writing the equation in matrix form, we have

$$E(M) = X^{\mathrm{T}}(K^{\mathrm{T}}K)X \quad (3)$$

where $K = (\omega_{ij})_{N\times N}$. Obviously, K is a sparse matrix.

There are several approaches for minimizing the discrete energy $E(M)$. The most popular one is to use an iterative

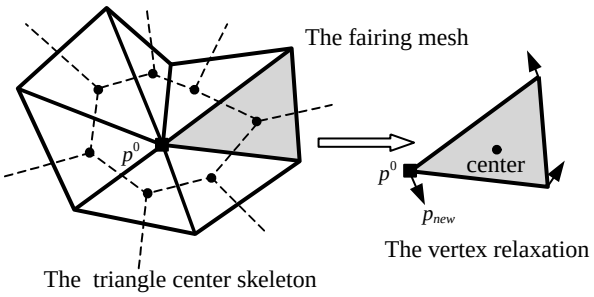


Figure 2: *The illustration of the center constraints.*

operator to locally solve the system, which implements the smoothing on meshes as a diffuse process. Taubin [16] and Desbrun [18] discussed the explicit and implicit integration for the solution, respectively. In our experience, the implicit approach is more stable and efficient. The resulting mesh, however, may be seriously shrunk and distorted due to the irregular connectivity of the original mesh. Although several researchers have proposed method to handle the problems in different manners [8,16–18,23], only a few can ensure the smoothing to be stable and have exact volume preservation without extra processing.

By carefully analyzing the problem of shrinkage and distortion, we find that the main reason is that the Laplacian operator causes too much relaxation on the vertices. Therefore, a natural solution is to reduce the vertex relaxation by introducing extra constraints to equation (3). In the following, we use the lever principle for reference to control the vertex relaxation by fixing all the triangle centers in position during fairing. As all triangle centers form a fixed skeleton to support the mesh, the vertex relaxation is limited to some degree (Figure 2). Hence, these constraints effectively prevent its shrinkage and distortion.

Let X^0 be the initial vertex positions of mesh M, then the barycenter constraint of triangle $\langle i, j, k\rangle$ can be described as:

$$(x_i + x_j + x_k) - (x_i^0 + x_j^0 + x_k^0) = 0.$$

Thus, the constraints on the whole mesh can be written in the following matrix form:

$$H(X - X^0) = 0. \tag{4}$$

Here, each triangle corresponds to a row in the $N \times N$ matrix H. Let $\langle i, j, k\rangle$ be the t-th triangle, then the t-th row of H is:

$$h_{ts} = \begin{cases} 1 & s = i, j \text{ or } k \\ 0 & \text{otherwise.} \end{cases}$$

Then the fairing for mesh X can then be described as:

$$\begin{cases} \min & X^{\mathrm{T}}(K^{\mathrm{T}}K)X \\ \text{subject to} & H(X - X^0) = 0. \end{cases} \tag{5}$$

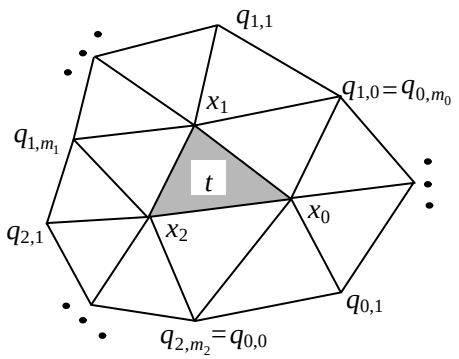


Figure 3: *Triangle t and its 1-ring neighborhood.*

This is a typical constrained optimization problem, which can be converted to an unconstrained one using the well-known penalty approach [24]:

$$\min X^{\mathrm{T}}(K^{\mathrm{T}}K)X + \mu(X - X^0)^{\mathrm{T}}(H^{\mathrm{T}}H)(X - X^0)$$

where μ is a positive constant. Solving a large linear system is usually time and memory consuming for a mesh with a large data set. In the following we will describe an iterative solution to equation (5).

3.2. Iterative solution

The key issue of our local iterative operator is to minimize the local discrete energy in the neighborhood of each triangle under the center constraint. Given a triangle t illustrated in Figure 3, let x_0, x_1, x_2 be its three vertices, the discrete energy in the neighborhood of t can be defined as:

$$\begin{aligned} E(t) &= \sum_{i=0}^{2} L(x_i) = \|AP - Q\|^2 \\ &= (P^{\mathrm{T}}A^{\mathrm{T}} - Q^{\mathrm{T}}) \cdot (AP - Q) \end{aligned}$$

where $A = \begin{bmatrix} 1 & -\omega_{01} & -\omega_{02} \\ -\omega_{10} & 1 & -\omega_{12} \\ -\omega_{20} & -\omega_{21} & 1 \end{bmatrix}$, $P = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$, $Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix}$, $q_i = \sum_{j=0}^{m_i} \omega_{ij} q_{i,j}$ $(i = 0, 1, 2)$. As illustrated in Figure 3, $q_{i,j}$ are the 1-ring neighboring vertices of triangle t, and m_i is equal to the valence of vertex x_i minus 2. The center constraint of a triangle can be rewritten as

$$BP = C$$

where $B = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, $C = BP^0$, $P^0 = \begin{bmatrix} x_0^0 & x_1^0 & x_2^0 \end{bmatrix}^{\mathrm{T}}$ is the original position of the triangle.

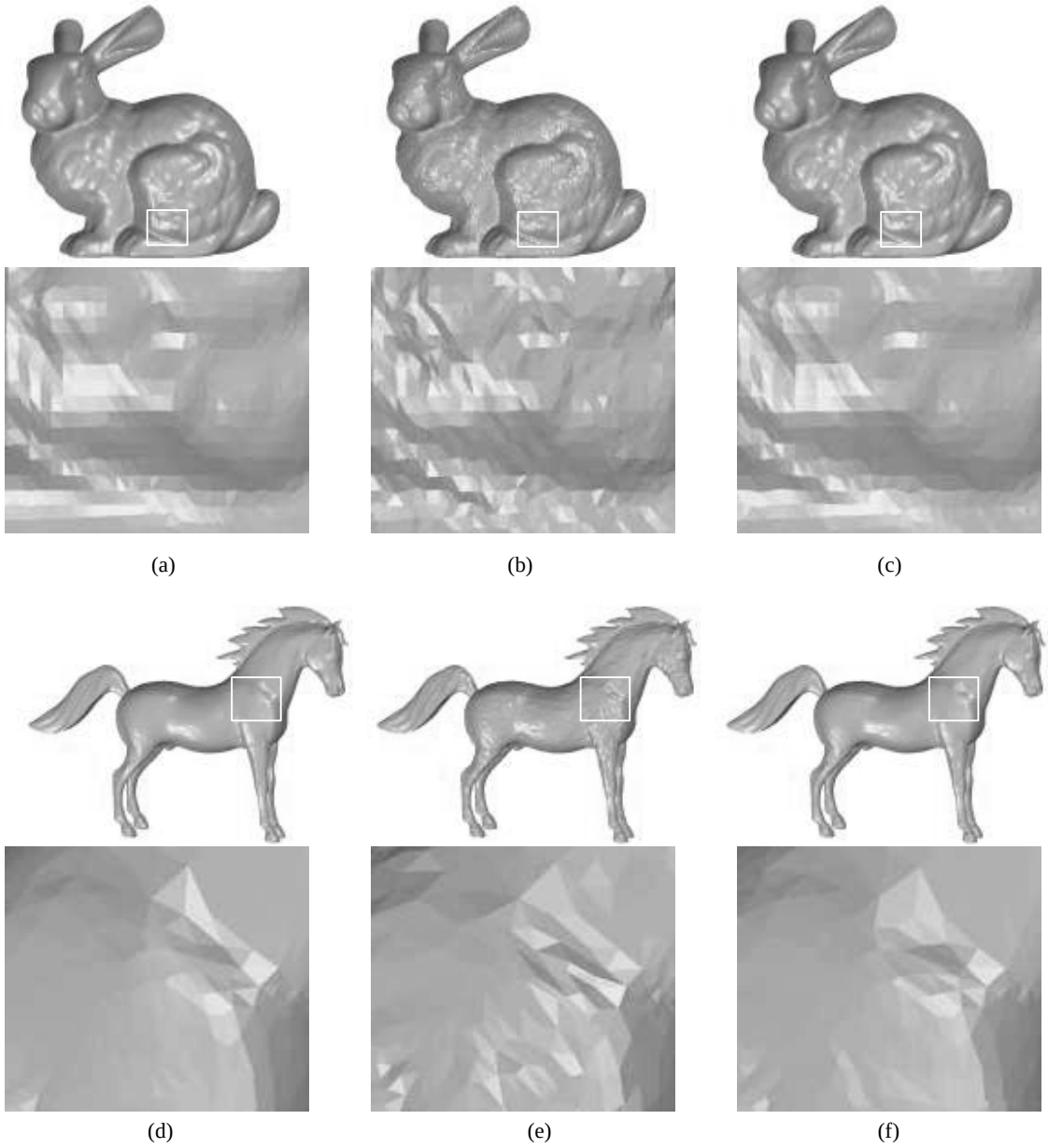


Figure 4: *Smoothing examples. (a), (d): the original models; (b), (e): models with white noise; (c), (f): smoothed models. In these figures, the lower is the enlargement of the region with box in the upper.*

Consequently, the local relaxation in the neighborhood of triangle t can be defined as the minimizer of $E(t)$ under the center constraints:

$$\begin{cases} \min & (P^{\mathrm{T}}A^{\mathrm{T}} - Q^{\mathrm{T}}) \cdot (AP - Q) \\ \text{subject to} & BP = C. \end{cases} \tag{6}$$

Since the functional is quadratic, it can be solved using the Lagrangian multiplier method [24]:

$$P = A^{-1}Q + \frac{1}{\|BA^{-1}\|^2}(C - BA^{-1}Q) \cdot A^{-1}(A^{-1})^{\mathrm{T}}B^{\mathrm{T}}. \tag{7}$$

Obviously, the new locations of the three vertices depend on the weights only. There are two popular ways to choose weights ω_{ij}, one depends only on the local connectivity of

meshes while the other takes both the geometry and the connectivity of meshes into account. Here, we adopt the first method to define the weights:

$$\omega_{ij} = \begin{cases} \frac{1}{n_i} & j \in S(i) \\ 0 & \text{otherwise.} \end{cases}$$

This choice guarantees that the matrices A^{-1}, BA^{-1} and $A^{-1}(A^{-1})^{\mathrm{T}}B^{\mathrm{T}}$ in equation (7) depend only on the valence of vertex i. In most cases, the valence of a vertex is among 5, 6 or 7, instead of computing these matrices on fly, we build look-up tables of the matrices for each valence number in a preprocessing step. Hence, we can efficiently calculate the new locations of three vertices of triangle t with these look-up tables.

The method mentioned above simultaneously relaxes three vertices of a triangle to minimize the local energy every time. Note that there are several triangles surrounding a vertex, say p, for complex meshes, which means one may to obtain several new locations $p_1^{\text{new}}, p_2^{\text{new}}, \ldots, p_m^{\text{new}}$ for vertex x (m is its valence) is by relaxing it for each of its surrounding triangles. In our current implementation, the final location of vertex x is determined by averaging these evaluated locations:

$$x_{\text{new}} = \frac{1}{m}\sum_{i=1}^{m} p_i^{\text{new}}.$$

As the result, the centers of the triangles may be slightly displaced, but it does not matter much since they are constrained to their original positions. In summary, our iteration operator for closed meshes can be described as follows:

Pold: input array of the orginal vertex locations.
Pnew: output array of the new vertex locations.
Cnew: temporary array of the number of new locations.

```
Initialization: Pnew = 0; Cnew = 0;
for each interior triangle t = ⟨i, j, k⟩ of the mesh
    Use equation (7) and Pold to compute new
        locations Newi, Newj, Newk of vertices i, j, k;
    Pnew[i] = Pnew[i] + Newi;
    Cnew[i] = Cnew[i] + 1;
    Pnew[j] = Pnew[j] + Newj;
    Cnew[j] = Cnew[j] + 1;
    Pnew[k] = Pnew[k] + Newk;
    Cnew[k] = Cnew[k] + 1;
end for
for all vertices v
    if Cnew[v] > 0 then
        Pnew[v] = Pnew[v]/Cnew[v];
    end if
end for
return;
```

In Figure 4(b) and (e), we introduce some white noise to the original models, the noisy models are then smoothed using the proposed algorithm. All the models are flat shaded. Comparing the resultant models with the original ones, our algorithm can effectively remove noise while preserving their original features. And only a few iterative steps are needed for small noise. Of course, if one wants to smooth the given models further, the basic fairing algorithm may not meet this requirement. The problem may be solved by imposing and relaxing the center constraints on the mesh by turns during smoothing, or using the multiresolution fairing algorithm presented in Section 4.

3.3. Boundary fairing

For open meshes, we also need to smooth their boundary curves. There are two ways to do so. One may attach some virtual triangles outside the boundary curve, and then treat the smoothing just like the closed meshes. The other way is to extract the boundary curves from a mesh, and then smooth the curves separately. In this paper, we take the second method.

Similarly, we first introduce the Laplacian operator for a discrete curve, say $p_0, p_1, \ldots, p_l$ which is defined as follows:

$$L(p_i) = \omega_{i,-1}(p_{i-1} - p_i) + \omega_{i,1}(p_{i+1} - p_i)$$

where $\omega_{i,-1}, \omega_{i,1} > 0$, and $\omega_{i,-1}+\omega_{i,1} = 1$. For simplicity, they may be given as $\omega_{i,-1}, \omega_{i,1} = \frac{1}{2}$. Adhering to the same idea mentioned before, the center constraints may also be introduced to prevent the curves from shrinkage and distortion. It is easy to deduce the local iterative equation:

$$\begin{bmatrix} p_{i-1} \\ p_i \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} p_{i-2} \\ p_{i+1} \end{bmatrix} + \begin{bmatrix} d \\ d \end{bmatrix}$$

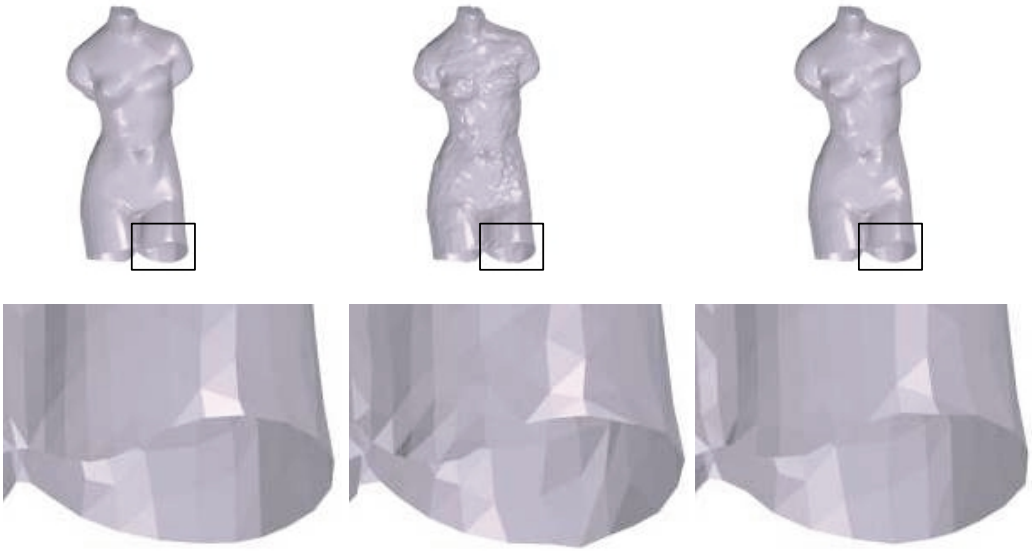
with

$$d = \tfrac{1}{2}[(p_{i-1}^0 + p_i^0) - (p_{i-2} + p_{i+1})].$$

Figure 5 illustrates the open Venus models and the smoothing result by our algorithm. Note that the boundary curve is smoothed nicely in Figure 5(c).

4. Multiresolution Smoothing

The above algorithm can efficiently remove the noise from the mesh. As demonstrated in Figures 4 and 5, the features are well preserved. For some applications, we need to remove not only noise, but also some undesired details from the mesh. This can be achieved by implementing the constrained fairing on the multiresolution representation. There are many published works on



(a) (b) (c)

Figure 5: *Another smoothing example. The Venus model is an open mesh, whose boundaries are smoothed by our curve smoothing method. A rather satisfactory result is obtained. The lower row are the zoomed views of the regions with a box in the upper. (a) The original model; (b) the noisy model; (c) the smoothed model.*

the topic of multiresolution representations for triangle meshes [9–13], which can be classified into two main categories: one is based on the subdivision connectivity, while the other is based on incremental mesh simplification. The former requires the subdivision connectivity of meshes. Hence a re-sampling process is always necessary to approximate the original model. This may introduce some undesired error before anything is done. Therefore, we adopt the later representation in our work for handling meshes with arbitrary connectivity.

4.1. Mesh hierarchy construction

We gradually simplify the original mesh to construct its multiresolution representation by vertex decimation. As shown by previous works, the order in which these simplification operations are performed greatly affects the shape graduation of simplified meshes. Naturally, most algorithms perform the simplification operations in the order of increasing error according to some metric. It is widely recognized that for the purpose of multiresolution smoothing, the metric should be carefully designed such that the vertices corresponding to small-scale details will be decimated first. According to this principle, we propose the following cost function to prioritize the vertices in the mesh:

$$\text{cost}(x) = \alpha \cdot \min\_\text{edge}(x) + \beta \cdot \max\_\text{edge}(x)$$

where α, β are two constants, $\min\_\text{edge}(s)$, $\max\_\text{edge}(x)$ are respectively the shortest and the longest lengths of incident edges of vertex x. α is always much greater than β in our current implementation, for example $\alpha = 1$, $\beta = 0.01$. Therefore vertices of less importance will be removed earlier due to the large α value, and extreme spindly triangles can be avoided too by the introduction of $\max\_\text{edge}(x)$. Additionally we add an *independent* set constraint in order to simplify the mesh as uniform as possible over the whole mesh.

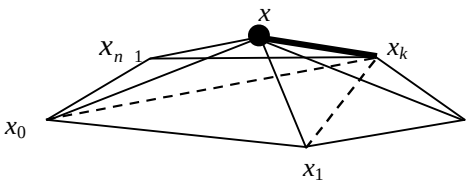
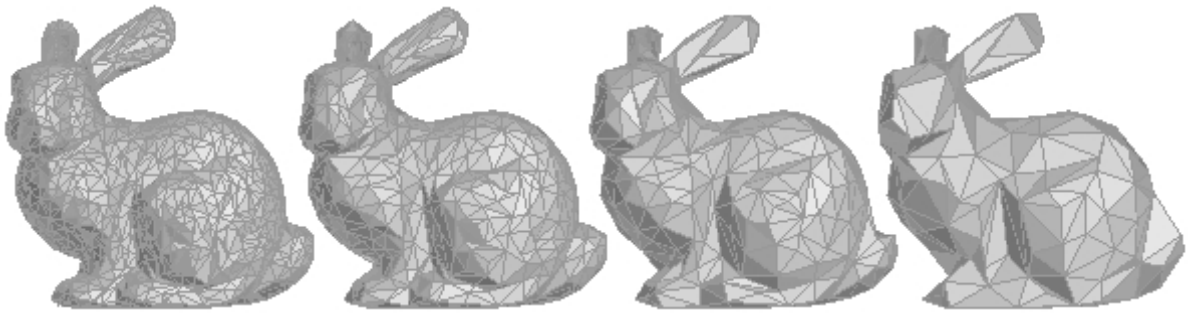


Figure 6: *Deformation caused by vertex decimation.*

To avoid any holes after decimating a vertex, our method removes the vertex to be decimated by merging it into a neighboring vertex. In order to preserve the global feature of the shape, we use a local volume metric for choosing the neighboring vertex. Let x be a vertex to be removed, and x_j, $j = 0, 1, 2, \ldots, n-1$ (n is its valence) be its 1-ring neighbors. Stemming from the spirit of half edge collapse, and suppose that vertex x is merged into vertex x_k, as shown in Figure 6. Then the cone at x bounded by



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

Figure 7: *Multiresolution representation with 24 levels of detail for the bunny model (see Figure 4(a)). (a), (b), (c) and (d) are respectively simplified models with 1320, 690, 361 and 198 vertices.*

its surrounding triangles $\Delta x x_j x_{j+1}$ and the bottom triangles $\Delta x_j x_{j+1} x_k (j = 0, 1, 2, \ldots, n-1)$ will disappear, the local deformation can be measured by the volume of the cone:

$$\left| \sum_j (x_j - x_k) \otimes (x_{j+1} - x_k) \cdot (x - x_k) \right|.$$

The above volume can be used as a cost function of the operation by merging x into x_k. By checking all the 1-ring neighboring vertices of x, we can easily determine the operation with the minimum cost.

There is another similar volume constraint defined in [14,15], which determines the optimal representative point for a collapsed edge such that the signed volume bounded by the original mesh and the simplified mesh is *zero*. Compared with our approach, it involves many more triangles, i.e. all triangles incident to the collapsed edge, in the volume computation, and it also requires solving a linear system to obtain the representative point.

With this gradual simplification method, we can build a multiresolution representation for the original mesh M_0: $\{M_0, M_1, M_2, \ldots, M_n\}$, which have decreasing complexity. Some experimental results on the bunny model shown in Figure 7 demonstrate that the volume criterion preserves the global features of the shape very well. There are in total 24 levels of simplified meshes in this multiresolution representation of the bunny model, which is generated in 4.84 s on a Pentium III 500 MHz machine.

4.2. Multiresolution fairing

Incorporated with the above generated multiresolution representation $\{M_0, M_1, M_2, \ldots, M_n\}$, the constrained fairing algorithm proposed in Section 3 can generate new surfaces with controllable smoothness. The basic idea of the multiresolution fairing algorithm is to perform mesh smoothing and mesh refining by turns until the original mesh is reached. The details can described as follows:

(1) Choose a starting level k of the simplified mesh to smooth.

(2) For i from k to 1:

  (2.1) Smooth M_i by the method introduced in Section 3.

  (2.2) Refine mesh from M_i to M_{i-1} using vertex split, i.e. the inverse operation of edge collapse.

In step (2.2), when a vertex is split out, it is initially put at the averaged position of its 1-ring neighbors, then adjusted by the method described by Kobbelt [8]:

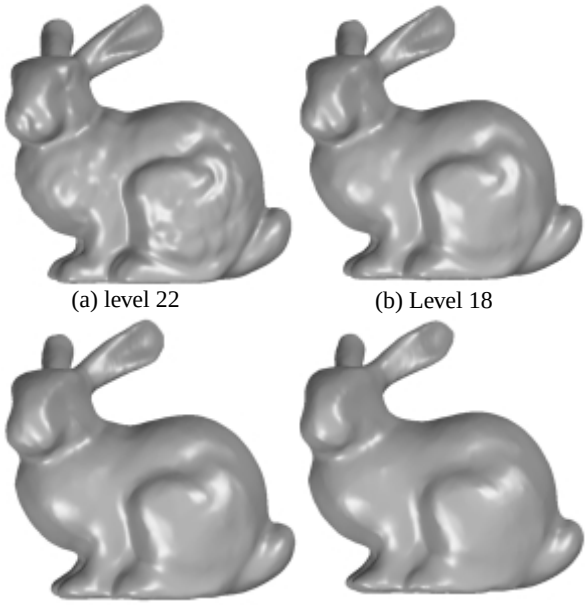
$$x_i \leftarrow x_i - \frac{1}{\nu} L^2(x_i), \quad \nu = 1 + \frac{1}{n_i} \sum_j \frac{1}{n_{ij}}$$

where n_i and n_{ij} are the valences of the center vertex x_i and its jth neighbor respectively.

Figure 8 shows the results by applying our algorithm to the bunny model (Figure 8(a)). In this example a multiresolution representation with 24 levels of simplified meshes is built, some of which are shown in Figure 7. Figure 8(b)–(d) are some fairing results starting respectively from a different level of simplified mesh. As illustrated in the figures, the coarser the selected mesh is, the smoother the resulting mesh will be obtained. Therefore one may conveniently discard details of different scales to obtain models with desired smoothness using our multiresolution fairing technique.

5. Conclusions

In this paper, we have presented a novel fairing algorithm to remove noise from triangular meshes. By introducing the barycenter constraints, our algorithm can produce a smooth surface without undesired shrinkage and distortion. It is very useful for smoothing the noisy models reconstructed from measured data. We have also designed a multiresolution



(a) level 22 (b) Level 18

(c) Level 16 (d) Level 14

Figure 8: *Multiresolution fairing results for the bunny model. (a), (b), (c) and (d) are the smoothed models discarding different level of details.*

fairing tool using the noise removal algorithm as the kernel, which allows the user to interactively discard any level of details from a fair surface. Meanwhile, we present a very fast volume-preserving simplification algorithm to build the multiresolution representation, which keeps the global feature of the model at each level very well. The future work is to apply the multiresolution fairing algorithm for shape modeling and reverse engineering.

Acknowledgements

The project is partly supported by the National Natural Science Funds of China for Distinguished Young scholars (No. 69925204) and for Creative Research Groups, and Hong Kong Research Grant Council Co-operative Research Centers Scheme (title: Virtual Reality, Visualization and Imaging Research Center). We appreciate the anonymous reviewers for their valuable comments.

References

1. N. Amenta, M. Bern and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH 98 Conference Proceedings*, pages 415–421.
2. Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH 96 Conference Proceedings*, pages 303–312.
3. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH 94 Conference Proceedings*, pages 295–302.
4. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 92 Conference Proceedings*, pages 71–78.
5. Hugges Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle. Mesh optimazation. In *SIGGRAPH 93 Conference Proceedings*, pages 19–26.
6. D. Zorin, P. Schroder and W. Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH'97 Conference Proceedings*, pages 259–268.
7. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings*, pages 173–182.
8. L. Kobbelt, S. Campagna, J. Vorsatz and H. P. Seidel. Interactive multiresolution modeling on arbitrary meshes. In *SIGGRAPH 98 Conference Proceedings*, pages 105–114.
9. Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–218.
10. Hugues Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108.
11. Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Grek Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks and William Wirght. Simplification envelopes. In *SIGGRAPH 96 Conference Proceedings*, pages 119–128.
12. A. Ciampalini, P. Cignoni, C. Montani and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13:228–246, 1997.
13. Willian J. Schroeder, Jonathan A. Zarge and William E. Lorensen. Decimation of triangle meshes. In *SIGGRAPH 92 Conference Proceedings*, pages 65–70.
14. P. Lindstrom and G. Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2), April, 1999.
15. P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *Proceedings of 9th Annual IEEE Conference on Visualization*, pages 279–286. ACM Press, October, 1998.

16. Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 95 Conference Proceedings*, pages 351–358.

17. J. Vollmer, R. Mencl and H. Müller. Improved Laplacian smoothing of noisy surface meshes. In *EUROGRAPHICS 99 Conference Proceedings*, pages 131–138.

18. M. Desbrun, M. Meyer, P. Schroder and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 99 Conference Proceedings*, pages 317–324.

19. L. Kobbelt. Discrete fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997.

20. W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH 94 Conference Proceedings*, pages 247–256.

21. H. P. Morton and C. H. Sequin. Functional optimization for fair surface design. In *SIGGRAPH 92 Conference Proceedings*, pages 167–176.

22. W. Welch and A. Witkin. Varational surface modeling. In *SIGGRAPH 92 Conference Proceedings*, pages 157–166.

23. I. Guskov, W. Sweldens and P. Schroder. Multiresolution signal processing for meshes. In *SIGGRAPH 99 Conference Proceedings*, pages 325–334.

24. A. Schabak and H. Werner. *Numerische Mathematik*. Springer, New York, 1993.