Laplacian-Based Feature Preserving Mesh Simplification

Lin Zhang, Zhen Ma, Zhong Zhou[*], and Wei Wu

State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
Beijing 100191, P.R. China
zz@vrlab.buaa.edu.cn

Abstract. We introduce a novel approach for feature preserving mesh simplification based on vertex Laplacians, specifically, the uniformly weighted Laplacian. Our approach is unique in three aspects: 1) a Laplacian based shape descriptor to quantize the local geometric feature sensitivity; 2) a Laplacian weighted cost function that is capable of providing different retaining rates of the geometric features; and 3) an optimal clustering technique which combines K-means and the Laplacian based shape descriptor to implement vertex classification. During simplification, the Laplacian based shape descriptors are firstly computed, and then a chosen error function to be optimized is penalized by our Laplacian weighted cost function, leading it to feature preserving. By applying the clustering technique, different simplification operators may be applied to different vertex groups for different purposes. Different error functions have been implemented to demonstrate the effectiveness, applicability and flexibility of the approach. Experiments conducted on various models including those of natural objects and CAD ones, show superior results.

Keywords: Mesh simplification, Laplace operator, Feature preservation, Feature detection.

1 Introduction and Related Work

Simplification of geometric models has become a hot topic today due to the rapid development of 3D scanning and acquisition technology. The acquired complex polygonal meshes face challenges in rendering, processing, and so on. Hence, after acquisition of geometric data, a simplification step is necessary, and its output should be a faithful approximation of the input. Generally, human vision is more sensitive to fine detail features and distinct sharp parts. Those detail features are usually visually meaningful fine scale components of a natural object, and typically require fine meshes to represent them well. This paper focuses on how to preserve these detail features during mesh simplification.

In the past decades, a fair amount of research on simplification techniques has been developed. In most existing simplification algorithms, edge collapse [1], which simplifies models by iteratively contracting edges has been adopted

[*] Corresponding author.

to provide it with precise results and high performance. Among numerous edge collapse algorithms, quadric error metric (QEM) [2], which estimates the error introduced by a pair collapse as the distance from a vertex to a quadratic surface, is the one with high performance but usually with the most precise results.

In order to preserve detail features of models during simplification, varieties of feature preserving mesh simplification algorithms have been proposed. One approach is to introduce new error arguments by geometric feature extraction. According to this viewpoint, algorithms based on QEM were introduced to avoid smearing out of real important features, such as [3,4,5]. The main difference between these algorithms lies in how they expands the quadric error matric with feature detection. Wang et al. [6] proposed a simplification algorithm based on feature extraction with the average curvature estimation to triangle mesh. Some direct methods such as [7] are proposed to control the relative importance of different surface regions by a user-guided approach. Rather than optimizing a piecewise-linear approximant of an original surface, [8] proposed an efficient variational approach which simplify models by mutual and repeated error-driven optimizations of a partition and a set of local proxies.

Most of the methods mentioned above do not provide users with different retaining rates of the geometric meaningful features. Vertex curvatures are widely used in these methods for feature detection, but computation of curvatures is obviously not cheap. The main contribution of this paper is to detect geometric features with a Laplacian based shape descriptor and to preserve the geometry feature with a Laplacian weighted cost function. Moreover, our approach is capable of providing different retaining rates of the geometric features. To the best of our knowledge, few works have been done before on feature preserving mesh simplification using the uniform Laplacian operator.

2 Uniform Laplacian Operator

Before describing the uniform Laplacian operator, we introduce notations. The surface mesh is represented as a graph $S = (V, E)$, with vertices V and edges E, where $V=[\mathbf{v}_1^T, \mathbf{v}_2^T, ..., \mathbf{v}_n^T]^T$, $\mathbf{v}_i = [v_{ix}, v_{iy}, v_{iz}]^T \in \mathbb{R}^3$. If two distinct vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are linked by an edge $\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ then we denote $j \in \mathcal{N}(i)$. The normal of vertex $\mathbf{v}_i$ is denoted as $\mathbf{n}_i$. Furthermore, δ_i is the Laplacian of $\mathbf{v}_i$, the result of applying the discrete Laplace operator to $\mathbf{v}_i$, i.e.

$$\delta_i = \sum_{j \in \mathcal{N}(i)} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \left[\sum_{j \in \mathcal{N}(i)} w_{ij}\mathbf{v}_j \right] - \mathbf{v}_i, \tag{1}$$

where $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$, and the choice of weights

$$w_{ij} = \frac{\xi_{ij}}{\sum_{k \in \mathcal{N}(i)} \xi_{ik}} \tag{2}$$

defines the nature of δ_i. One popular choice is $\xi_{ij} = 1$, which defines the uniform weights, i.e. the umbrella operator.

The umbrella operator of $\mathbf{v}_i$ points to the centroid of its neighboring vertices. When the umbrella operator is applied to the mesh, it smears out the real important features of the mesh, which named shrinking. Different from many works that try to avoid the shrinking problem, we utilize it and introduce an normal-dependent umbrella operator to quantize geometric sensitivity. The operator will be described and discussed in the next section.

However, the umbrella operator suffers from the problem of large inaccuracies for irregular meshes, which prevents us from using it for feature detection directly. It is a linear from implying the assumption that the mesh has edges of length 1 and all the angles between two adjacent edges around a vertex should be equal. This is obviously not true in actual meshes, which leads to inadequacy of the umbrella operator. Fig. 1(b) shows such a behavior, when the umbrella operator is applied, border lines are strangely distorted into curves. To deal with the problem caused by different length of edges, [9] presented a scale-dependent umbrella operator. Besides, [10] presented the mean curvature normal which compensates both for unequal edge lengths and for unequal face angles. However, they suffers from a difficulty in dealing with arbitrary surfaces, such as quad meshes. For more details refer to [11].

3 Normal-Dependent Umbrella Operator

To improve inaccuracies of the umbrella operator, we introduce the normal-dependent umbrella operator using the following formula:

$$\widetilde{\delta_i} = \left[\sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} (\mathbf{v}_j - \mathbf{v}_i)\mathbf{n}_i \right] \cdot \mathbf{n}_i, \tag{3}$$

where $|\mathcal{N}(i)|$ is the number of neighbors of vertex $\mathbf{v}_i$.

Note that when all edges are of size 1, and all the angles between two adjacent edges around a vertex are equal, $\widetilde{\delta_i}$ reduces to the umbrella operator. Fig. 1(e) shows an example of it for a triangle mesh vertex and its neighbors.

Fig. 1 demonstrates the normal-dependent umbrella operator performs more stable for an actual mesh than the original and scale-dependent ones, particularly for sharp features such as lines and corners, meaning that it quantizes the local geometric feature sensitivity precious more precisely. Moreover, the normal-dependent umbrella operator is actually the projection of the umbrella operator along the normal $\mathbf{n}_i$, which does not limit the umbrella operator to triangulated surface, and it is also computationally fast. Our Laplacian based shape descriptor which will be introduced later is based on it.

4 Mesh Simplification

In this section, the overall pipeline of the approach is first described. Laplacian based shape descriptor is then discussed in detail, followed by the Laplacian weighted cost function. Finally an optimal clustering technique to implement vertex classification is discussed.

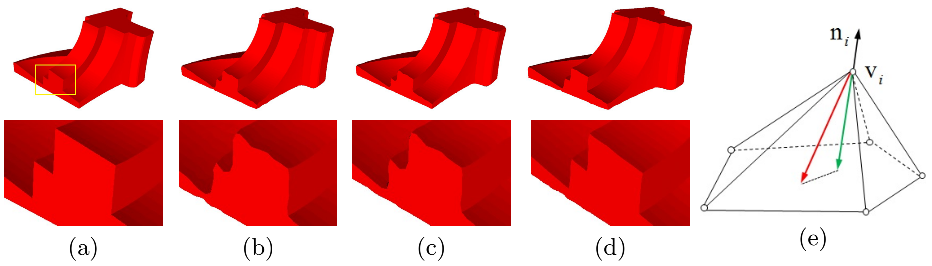


Fig. 1. (a) Original Fandisk model; (b), (c), (d) are smoothed results by applying umbrella operator, scale-dependent, and normal-dependent umbrella operator; (e) uniform (red) and normal-dependent (green) umbrella operator vectors for a vertex $\mathbf{v}_i$ and its 1-ring neighbours, as well as the vertex normal $\mathbf{n}_i$. While the other two operators distort lines and corners, the normal-dependent one does not.

4.1 Algorithm Description

Our simplification approach is based on two steps (see Fig. 2):

A Laplacian based feature detection and vertex classification: geometry features are first detected by calculating the Laplacian based shape descriptor, and then an optimal clustering algorithm can be applied to divide vertices into clusters, particularly for the CAD models.

A Laplacian-weighted simplification: a chosen error function to be optimized is penalized by the Laplacian weighted cost function while leading to a feature preserving simplification.

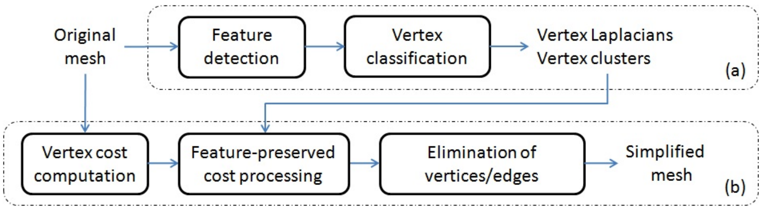


Fig. 2. Two steps of the approach. (a) Laplacian based feature detection and vertex classification; (b) Laplacian-weighted simplification.

4.2 Laplacian Based Shape Descriptor

We use the normal-dependent umbrella operator to detect geometric features. For each vertex $\mathbf{v}_i$, the length of its normal-dependent umbrella operator vector

$$|\widetilde{\delta_i}| = \frac{1}{|\mathcal{N}(i)|}\left[\sum_{j\in\mathcal{N}(i)} (\mathbf{v}_j - \mathbf{v}_i)\right]\cdot \mathbf{n}_i \tag{4}$$

is the projection of δ_i along the vertex normal $\mathbf{n}_i$. Clearly, $|\widetilde{\delta_i}|$ is larger at crease vertices than at flat ones. When $\mathbf{v}_i$ and its neighbors are in the same plane, $|\widetilde{\delta_i}|$ goes to zero. In addition, as discussed before, visually meaningful fine scale components or details typically require very fine short edges to represent them well, we thus score the geometric importance at vertex $\mathbf{v}_i$ as

$$l_i = \frac{|\widetilde{\delta_i}|}{c_i}, \text{ where } c_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} |\mathbf{e}_{ij}|. \tag{5}$$

In the remainder of this paper, we will denote this Laplacian based shape descriptor as the *Laplacian descriptor*. By computing l_i for each vertex $\mathbf{v}_i$, a quantitative 1D data array $L \in \mathbb{R}^{1\times n}$ is then generated as $L = [l_1, l_2, ..., l_n]$. Consequently, vertices with small l_i are those non-feature vertices of meshes, and vertices with large l_i are those meaningful feature details.

Fig. 3 shows the Laplacian descriptor and its comparison with popular curvatures. As we can see in Fig. 3(b) and Fig. 3(d), Laplacian descriptors of meaningful geometric feature vertices are larger than those of the non-feature vertices in flat regions. Fig. 3(e) and Fig. 3(f) shows comparison of Laplacian descriptors and curvatures. In Fig. 3(e) and Fig. 3(f), to enable a better visual comparison, vertices are firstly sorted in ascending order by their values of features descriptors, and then are plotted along the horizontal axis. In general, vertices in same region (e.g., flat surface, border line) should have similar values of shape descriptors. Take the Fandisk model in Fig. 3(a) as an example, vertices of it can be roughly clustered into several small groups, such as flat surface vertices, curved surface vertices, border vertices, and so on. Therefore, feature descriptors should also have such characteristic of clustering. As can be seen in Fig. 3(e), both the Laplacian descriptor and the mean curvature has obvious grouping results (staircase curve), meaning that they are more precise than the other two.

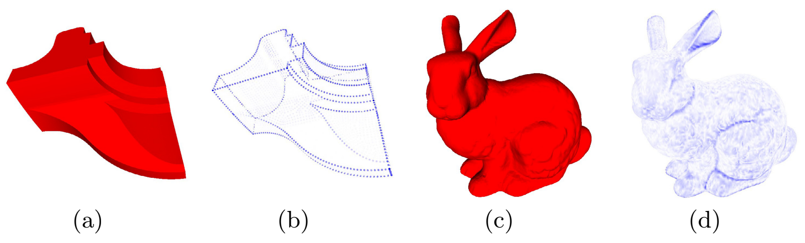
Fig. 4(a) is a partial enlarged view of Laplacian descriptors in Fig. 3(e), and is roughly divided into several segments. Fig. 4(b) is the 3D plot of vertices corresponding to Fig. 4(a). Each segment demonstrates a small group of vertices with the similar Laplacian descriptors, demonstrating that it is more accurate than the smooth curvature estimation in [6] and Gaussian curvature.

4.3 Laplacian Weighted Cost Function

Since the Laplacian descriptor l_i gives the geometric importance for vertex $\mathbf{v}_i$, we use it to guide an existing simplification method for a feature preserved simplification. Given a chosen cost function $g()$, such as QEM or shortest edge length, we apply the Laplacian weighted cost function that we design below.

$$f(\mathbf{v}_i) = g(\mathbf{v}_i)e^{\lambda l_i} \tag{6}$$

where λ is a scale factor and all l_i are linearly rescaled into range [0,1]. By changing the value of λ, we can get different retaining rate of the geometric features.



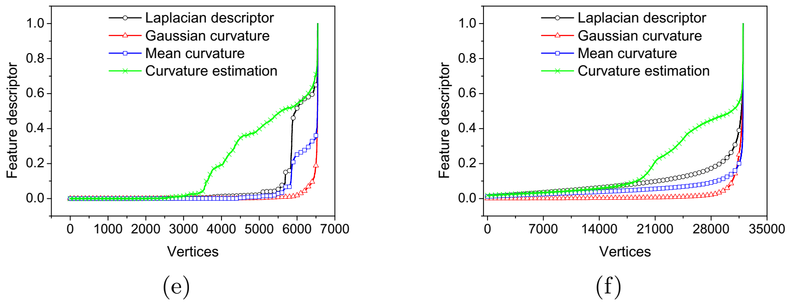


Fig. 3. Feature descriptor. (a), (c) input models; (b), (d) 3D plot of Laplacian descriptors; (e), (f) comparison of Laplacian descriptors and curvatures.

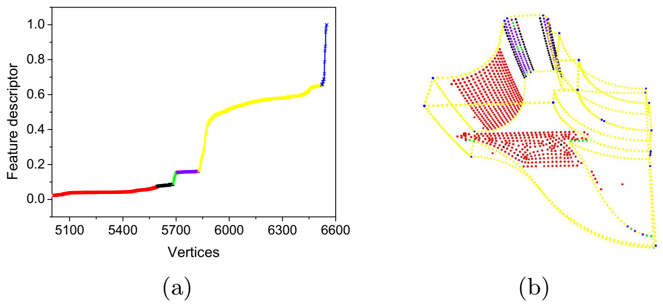


Fig. 4. (a) a partial enlarged color view of Laplacian descriptors in Fig. 3(f); (b) 3D plot of vertices corresponding to (a)

For ease of explanation, we take the shortest edge first algorithm as an example. It first computes cost for each vertex $\mathbf{v}_i$ as

$$g(\mathbf{v}_i) = min(|\mathbf{e}_{ij}|), \text{ where } j \in \mathcal{N}(i), \tag{7}$$

then the order of collapses is built on the basis of $g(\mathbf{v}_i)$. This straightforward algorithm can not preserve geometric features during simplification. Nevertheless, we can now improve it to a feature preserving simplification algorithm by simply applying Equation 6 on $g(\mathbf{v}_i)$, see Fig. 5. Most of the edge contraction algorithms, which can not lead to a feature preserving simplification, can also be improved in the similar way. In the remainder of this paper, we refer to those improved algorithms as Laplacian guided ones.

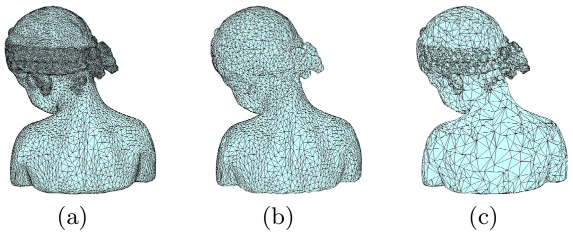


Fig. 5. Compare the guided shortest edge first altorithm with the original ($\lambda = 10$). (a) original girl model (30k tri.); (b) result of the original shortest edge first algorithm (11k tri.); (c) result of Laplacian-guided shortest edge first (11k tri.).

4.4 Vertex Classification

In many cases, one simple simplification algorithm cannot generate a satisfactory reconstruction. Fig. 6 demonstrates such an example. This suggests that different methods may best be applied for different purposes. Vertex classification is thus a good way to go further. For example, if we classify vertices into two non-feature and feature groups, we may apply different simplification operators with different parameter selections to pursue a better reconstruction.

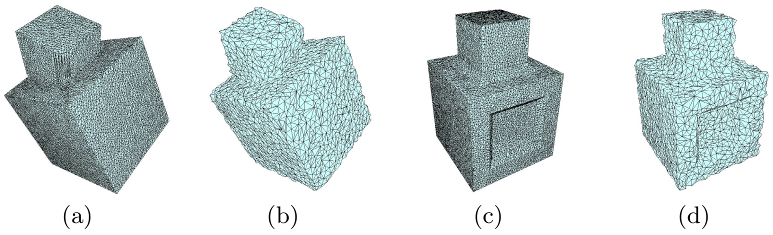
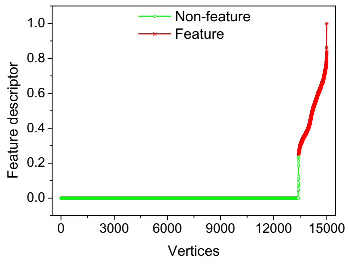


Fig. 6. Boundaries are distorted when the shortest edge first algorithm is applied to models. (a) original Mech A (30k tri.); (b) simplified Mech A (5k tri.); (c) original Mech B (30k tri.); (d) simplified Mech B (5k tri.).

Vertices of the mesh are classified by their Laplacian descriptors. The clustering is done via the K-means algorithm [12], allowing to divide vertices into k groups. Given the Laplacian descriptor array L, we first seek a partition $L_1, L_2, ..., L_k$ to minimize the objective function

$$\sum_{j=1}^{k} \sum_{l_i \in L_j} |l_i - \mu_j|, \tag{8}$$

where L_j is a set of vertices in the j-th cluster and $\mu_j = mean(\sum_{l_i \in L_j} l_i)$ is the center point over the j-th cluster. In current context, vertices are divided into two clusters, feature and non-feature (see Fig. 7). Vertex classification is usually useful for a CAD model but not for a natural object such as Bunny in Fig. 3(c). However, this statistical information maybe still be useful in other contexts.



(a)

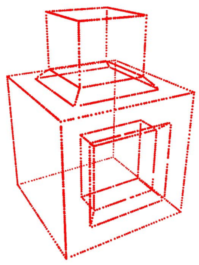
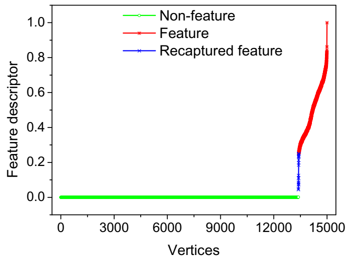

(b)

Fig. 7. (a) two clusters by K-means on Laplacian descriptors of model in Fig. 3(a); (b) 3D plot of feature vertices based on clusters in (a).

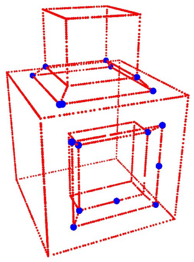
Fig. 7 demonstrates that our method yields rather pleasing results in terms of vertex clustering. But due to the diversities and complexity of different models, we may not get exact vertex classification even when the more advanced clustering techniques are employed. Some feature vertices might be wrongly grouped into non-feature cluster. Therefore, a post-clustering procedure is needed. Notice that boundary vertices for a CAD model should have at least two neighboring feature vertices. By making good use of this property, we apply the clustering and refinement procedure as below (see Fig. 8 for the result).

1) Compute the Laplacian descriptor array L.
2) Clustering: Apply the K-means clustering on L.
3) Refinement: If there exist vertices in the feature cluster which have less than two feature neighbors, send the vertex in the non-feature group whose Laplacian descriptor is the biggest into the feature cluster.
4) Repeat step 3 until each vertex in the feature cluster has more than two feature neighbors, or the approach reach a certain threshold.

There has been a considerable research work on boundary detection relevant to the problem of CAD mesh segmentation, but the main goal of the refinement approach here is to show the advantage of using the Laplacian descriptor array $\widetilde{L}$, and we do not pretend in this paper to contribute in the field of CAD mesh segmentation. To conduct a perfect boundary making of a complex model such as Fandisk model in Fig. 3(a), more advanced classification refinement techniques are certainly needed, for instance, contour tracking in [13].



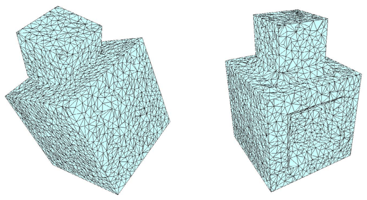
(a)



(b)

Fig. 8. Clustering refinement of example in Fig. 7. (a) recapturing potential feature vertices back into the feature cluster with the classification refinement procedure; (b) 3D plot of feature vertices where blue dots mark the recaptured vertices.

Since vertices are divided into to two clusters, we can now improve the boundary distorting problem in Fig. 6(b) and Fig. 6(d). For each boundary vertex, we compute its cost $g(\mathbf{v}_i)$ by finding the shortest edge only between its feature neighbors instead of Equ. 7. The improved results are shown in Fig. 9.



(a) (b)

Fig. 9. Improve the shortest edge first algorithm by using different cost functions based on vertex classification. (a) improved result of Fig. 6(b); (b) improved result of Fig. 6(d).

5 Case Study

All tested algorithms here are integrated into the same testing framework written by Somers. Fig. 1 demonstrates that the normal-dependent umbrella operator performs precisely on practical irregular meshes. Figs. 3 and 4 demonstrate that the Laplacian shape descriptor produces rather pleasing results. Fig. 10 demonstrates the successful extension of it to non-manifold and quad meshes. Table 1 summarizes the running time of computing the Laplacian shape descriptor array L, comparing with curvature-based methods on a PC with Intel Duo 3.30GHz processor and 4GB memory. Due to the fast normal-dependent umbrella operator, our algorithm is much more faster than the mean curvature. The mean curvature estimation method proposed in [6] costs the similar time with ours.

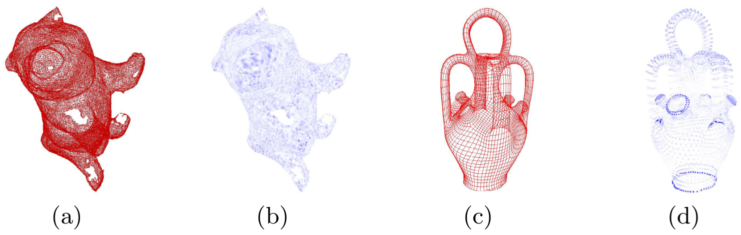


Fig. 10. (a) non-manifold cinghiale; (b) 3D plot of Laplacian descriptors of (a); (c) quad mesh; (d) 3D plot of Laplacian descriptors of (c)

Table 1. Time Comparison of feature detection (in s)

Model	Vertices	Faces	Laplacian operator	Mean Curvature	Curvature Estimation
Cow	2903	5804	0.004	0.009	0.006
Fandisk	6551	13098	0.009	0.021	0.012
Mech A	14999	29994	0.027	0.081	0.038
Mech B	15000	29996	0.026	0.08	0.03
Girl	15516	31028	0.028	0.122	0.043
Bunny	34834	69451	0.065	0.469	0.118

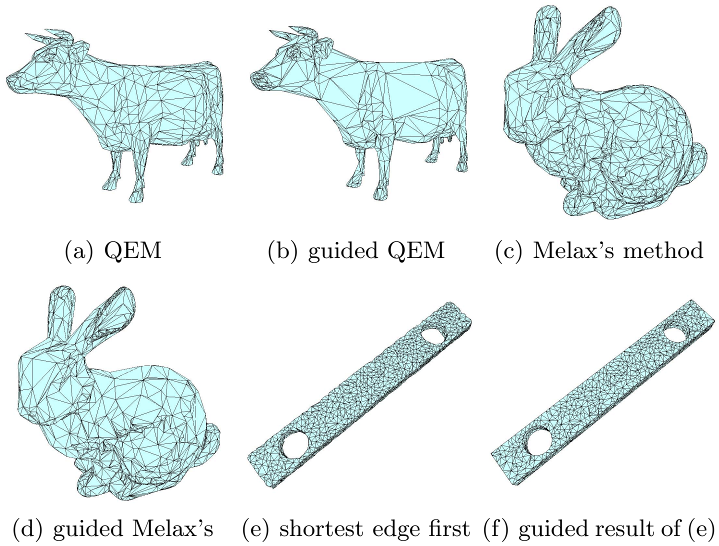


Fig. 11. Experiments on different error functions. (a),(b) Cow (5804 tri. to 2052 tri.); (c),(d) Bunny (69451 tri. to 4081 tri.); (e),(f) Mech (15k tri. to 4k tri.).

Figs. 5 and 11 show some example implementations of our Laplacian-based approach and demonstrate that it can improve different error functions to feature preserving. All test models has distinct feature regions including creases, corners and boundaries, those results demonstrate how important geometric structures are preserved in the output. Fig. 12 demonstrates that our method is capable of providing different retaining rates of the geometric features by only changing

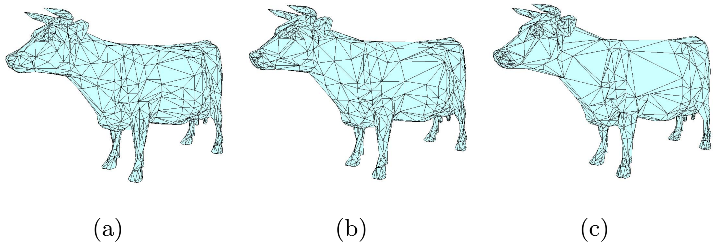


Fig. 12. Simplified Cow models from guided QEM algorithm with different retaining rate of features (all 3k tri. to 2k tri.). (a) $\lambda = 1$; (b) $\lambda = 10$; (c) $\lambda = 50$.

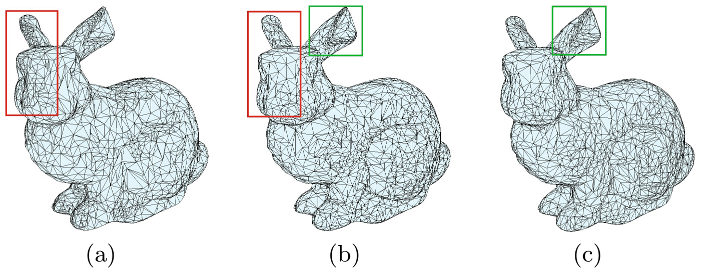


Fig. 13. Contrast with Curvature Estimation [6] and Curvature-weighted QEM [4] (35k tri. to 6924 tri.). (a) curvature estimation; (b) Laplacian guided QEM ($\lambda = 10$); (c) curvature-weighted QEM.

the value of scale factor λ, which is very useful in practical application. Fig. 13 shows the comparison results for the Bunny model. Both Laplacian guided and curvature-weighted algorithm preserve better feature details than the curvature estimation algorithm, because of their accuracy in shape description.

6 Conclusion

We have presented a new feature preserved simplification approach. Geometry features are first detected using a Laplacian based shape descriptor. An optional clustering technique which combines K-means and vertex Laplacians is developed to implement vertex classification, different simplification and refinement operators can then be applied on different vertex groups for different purposes. Moreover, we introduce a Laplacian weighted cost function which is capable of providing different retaining rates of the geometric features. In addition, our approach may be applied on different error functions, leading them to feature preserving. The Laplacian based shape descriptor could be easily extended to handle arbitrary mesh topology. Due to advantages of it, our method is computationally efficient and can be easily implemented. Experimental results demonstrate that our algorithm has strong applicability and yields superior results.

Acknowledgments. This work is supported by the National 863 Program of China under Grant No.2012AA011803, the Natural Science Foundation of China under Grant No.61170188, and the National 973 Program of China under Grant No. 2009CB320805.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: Proceeding of SIGGRAPH 1993, pp. 19–26 (1993)
2. Garland, M., Heckbert, P.: Surface Simplification Using Quadric Error Metrics. In: Proceeding of SIGGRAPH 1997, pp. 209–216 (1997)
3. Wei, J., Lou, Y.: Feature Preserving Mesh Simplification Using Feature Sensitive Metric. Journal of Computer Science and Technology 25, 595–605 (2010)
4. Li, L., He, M., Wang, P.: Mesh Simplification Algorithm Based on Absolute Curvature-weighted Quadric Error Metrics. In: 5th IEEE Conference on Industrial Electronics and Applications, ICIEA, pp. 399–403 (2010)
5. Jong, B.S., Teng, J.L., Yang, W.H.: An efficient and low-error mesh simplification method based on torsion detection. The Visual Computer 22, 56–67 (2005)
6. Wang, J., Wang, L.R., Li, J.Z., Hagiwara, I.: A feature preserved mesh simplification algorithm. Journal of Engineering and Computer Innovations 6, 98–105 (2011)
7. Kho, Y., Garland, M.: User-guided simplification. In: Proceedings of ACM Symposium on Interactive 3D Graphics, pp. 123–126 (2003)
8. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. In: Proceedings of ACM SIGGRAPH, pp. 905–914 (2004)
9. Fujiwara, K.: Eigenvalues of laplacians on a closed riemannian manifold and its nets. In: Proceedings of AMS, vol. 123, pp. 2585–2594 (1995)
10. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Proceedings of VisMath, Berlin, Germany (2002)
11. Sorkine, O.: Differential Representations for Mesh Processing. Computer Graphics Forum 25(4), 789–807 (2006)
12. Gersho, A., Gray, R.: Vector quantization and signal compression. Kluwer, Boston (1992)
13. Lavoué, G., Dupont, F., Baskurt, A.: A new CAD mesh segmentation method, based on curvature tensor analysis. Computer-Aided Design, 975–987 (2005)