A Visual and Geometry-based Hybrid Approach for Surface Simplification

Lian Ping Xing[1] and Kin Chuen Hui[2]

[1]The Chinese University of Hong Kong, lpxing@mae.cuhk.edu.hk
[2]The Chinese University of Hong Kong, kchui@mae.cuhk.edu.hk

ABSTRACT

In this paper, a novel visual and geometry-based simplification approach is proposed for simplifying polygonal meshes. This approach is driven by a visual importance weighted quadric error metric, which measures not only the geometry differentiation but also the local surface variation. We define the visual importance of one vertex based on its vertex curvature entropy which reflects the visually variation of the region centered at this vertex. We observe that such a definition of visual importance is capable of capturing visually interesting regions on a mesh. By combining the visual importance and quadric error metric to measure the edge collapse error, this algorithm produces simplified model closer to the original one according to visual similarity. One main application of this algorithm is the simplification of models for video games where the models are usually geometrically not very complex, and in which visual similarity is the most important requirement.

Keywords:mesh simplification, level of detail, information theory.
DOI: 10.3722/cadaps.2012.167-176

1 INTRODUCTION

With the advance in scanning technology, highly detailed three-dimensional (3-D) mesh models are easily acquired. However, such highly detailed models often post difficulties in the transmission, storage and rendering of the model. A variety of simplification algorithms have been proposed to reduce the complexity of the models to achieve higher efficiency. Most commonly used simplification methods use a technique based on geometric distance as a quality measure between the original mesh and the one obtained from simplification. With these methods we can achieve meshes with higher fidelity. However after a drastic simplification, most of these algorithms lose important shape features and induce visual degeneration. On the other hand, there are applications which require simplified meshes that appear visually similar to the original ones (e.g. video games).

In this paper, we introduce a novel visual and geometry-based simplification approach for polygonal meshes. This method uses a visual importance weighted quadric error metric to quantify the cost of an edge collapse. We define a visual importance for every vertex in the original model. The visual importance is an indicator that reflects the local variation of the region centered at this vertex. By combining the visual importance and quadric error metric, our method yields better visual and geometric performance than QSlim-based simplifications [8].

The rest of this paper is organized as follows. In Section 2, we survey the edge contraction and other works, the basic quadric error metric and entropy measures. Section 3 presents the definition of visual importance. In section 4, we describe our simplification algorithm. Results are shown in section 5. Finally, we conclude the paper in section 6.

2 BACKGROUND AND RELATED WORK

In this section, we review related work especially some methods based on edge contraction, basic quadric error metric and entropy measure.

2.1 Related Work

A number of algorithms have been published, which simplify models using topological operations such as vertex decimation[6],[18],[19], vertex clustering [14],[17], iterative edge contraction [5],[8],[10],[11],[20],[22] and face constriction [2],[21]. Edge contraction is the most common operator. Algorithms based on this operation can generate multi-resolution model supporting progressive transmission and rendering.

The first method using edge contraction operation was proposed by Hoppe [11]. In this algorithm, an energy function is defined to measure the quality of the simplified mesh. This method generates high quality representation of a given model, but the process of finding an optimal solution of the energy function is very slow. This scheme is hence not suitable for time-critical applications. Then, Hoppe extended his initial work [11] and proposed a new quadric metric that incorporates colors, normal and texture coordinates and introduced a progressive mesh scheme for storing and transmitting arbitrary triangle meshes [10]. Cohen et al. [5] developed an algorithm based on a texture deviation metric that samples the vertex position, normal and color attributes of the original mesh and then converts them to normal and texture maps. One of the notable methods based on edge contraction is Garland and Heckbert's QEM [8]algorithm which uses quadric error metric to evaluate the cost of vertex-pair contraction. Both the speed and efficiency of this algorithm are promising. Recently, Wu [22] gives a modified QEM based approach to define the contraction cost of every edge. Wei [20] presents a new method for feature preserving based on feature sensitive (FS) metric which extends quadric error to a high-dimentional FS space to measure both the geometric distance and surface normal variation.

Lindstrom et al. [13] introduced an image-driven simplification framework and addressed the problem of visual similarity by developing a pure image-based metric. Luebke et al. [15] presented a method to perform a view-depended polygonal mesh simplification using perceptual metrics. Zhang et al. [23] proposed a new view-independent algorithm based on the visibility function between the surfaces and a surrounding sphere of cameras.

Recently, Lee et al. [12] introduced the idea of mesh saliency as a measure of regional importance for graphics meshes and incorporated it into mesh simplification. Their approach consists in generating a mesh saliency map, and then constructing the edge collapse cost to be a function of the saliency of this edge. Though this approach can generate simplified meshes with visually interesting features, however, it sometimes does not consider the contour of the mesh models especially when the model is simplified to a high level.

2.2 Quadric Error Metric

In [8], Garland and Heckbert defines the QEM (quadric error metric) scheme on every vertex$v$as the sum of the squared distance of$v$to its adjacent planes $(planes(v))$. By representing the coordinate of a vertex in its homogeneous form, e.g.$v = [x, y, z]^T$ is represented as$v = [x, y, z, 1]^T$, the distance from a vertex$v$to a plane can be written as$D^2(v) = (p^T v)^2$, where$p = [a, b, c, d]^T$represents the plane defined by $ax + by + cz + d = 0$ and $a^2 + b^2 + c^2 = 1$. So the error at vertex v is defined as a quadric form:

$$\Delta(v) = \sum_{p\epsilon planes(v)} D^2(v) = \sum_{p\epsilon planes(v)} v^T(pp^T)v = v^T\left(\sum_{p\epsilon planes(v)} pp^T\right)v = v^T\left(\sum_{p\epsilon planes(v)} k_p\right)\mathrm{v} \qquad (2.1)$$

Where $k_p$is the matrix:

$$k_p = \mathbf{p}\mathbf{p}^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

This *fundamental error quadric*$k_p$can be used to find the squared distance of any point in space to the plane$p$. By summing these fundamental quadrics together and representing an entire set of planes by a single matrix$Q$. The Eqn. (2.1) can be rewritten as:

$$\Delta(v) = v^T Q v \qquad (2.2)$$

For an edge collapse$(v_1, v_2) \to \bar{v}$, the optimal position of the newly formed vertex$\bar{v}$is calculated by solving$\frac{\partial\Delta}{\partial x} = \frac{\partial\Delta}{\partial y} = \frac{\partial\Delta}{\partial z} = 0$ and is equivalent to:

$$\mathbf{v} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Whereq_{ij}denotes the element of row$i$and column$j$in matrix$Q$. The quadric for the simplified vertex$\bar{v}$is calculated simply by adding the two quadrics:

$$\bar{Q} = Q_1 + Q_2 (2.3)$$

Where $Q_1$is the quadric of v_1, Q_2 is the quadric of $v_2$and the error for the new vertex is:

$$\Delta(\bar{v}) = \bar{v}^T \bar{Q} \bar{v} \qquad (2.4)$$

2.3 Entropy Measure

Let$X = \{x_1, x_2, \ldots, x_n\}$be a set of discrete random variables with the probability distribution$\{p_1, p_2, \ldots, p_n\}$ where$p_i = pr\{X = x_i\} i \in \{1,2,\ldots,n\}$denoting the probability that the random variable$X$takes on the valuex_i. The entropy$H(X)$ of this set is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log(p(x)) \qquad (2.5)$$

All logarithms are base 2 and the convention$0\log 0 = 0$is adopted [7]. The term $-\log(p(x))$ represents the self-information of random variable$x$, which is a measure of the information content associated with the outcome of the random variable$x$. The amount of self-information contained in a probabilistic event depends on the probability of that event: the smaller its probability, the larger the self-information associated with receiving the information that the event indeed occurred. The entropy$H(X)$measures the average amount of information expressed by the whole random variables. The entropy indicates the global statistical property of the information source and is the measurement of medial uncertainty of the entirety. There is a unique entropy value for a certain source and the entropy may vary as its statistical properties changes.

3 VISUAL IMPORTANCE COMPUTATION

Lindstrom and Zhang's [13],[23] methods are effective techniques for computing visual similarity or visibility measures based on 2D images. Lee [12] introduces a method to calculate a saliency map for 3D meshes. He defines mesh saliency in a scale-dependent manner. However, it is difficult to determine multiple scales to express an exact mesh saliency. Information theory tools have been used to study scene visibility [3]. Our method for computing visual importance for 3D meshes uses the center-surround mechanism as [12].

One of the features we associate with the visual quality of a 3D mesh is the amount of information it provides us with. Unlike images, where color is the most important information, for 3D meshes, the geometry and its variation of 3D meshes is the most important contributor to the mesh information. Intuitively, the changes in the curvatures on the mesh well reflect the characteristics of the mesh model [12]. We are also encouraged by the success of entropy technique on measuring variation of an event. This has led us to formulate visual importance in terms of the mean curvature used with the center-surround mechanism and entropy measurement.

The first step of the visual importance computation involves computing surface curvatures. There are a number of excellent approaches that generalize differential-geometry-based definition of curvatures to discrete meshes [1],[9],[16]. One can use any of these to compute the curvature of a discrete mesh at a vertex. We have tried several methods and found that the method presented in [1] gave us better results and that is what we use here.

Let the neighborhood of a vertex$v_i$be the set of vertices$X = \{v_{i_0}, v_{i_1}, \dots, v_{i_n}\} where v_{i_0} = v_i$which is composited of the vertex in consideration and its associated vertices. The vertex curvatures of this set can be$K = \{k_{i_0}, k_{i_1}, \dots, k_{i_n}\}$. Then we define vertex curvature entropy for the vertexv_i as:

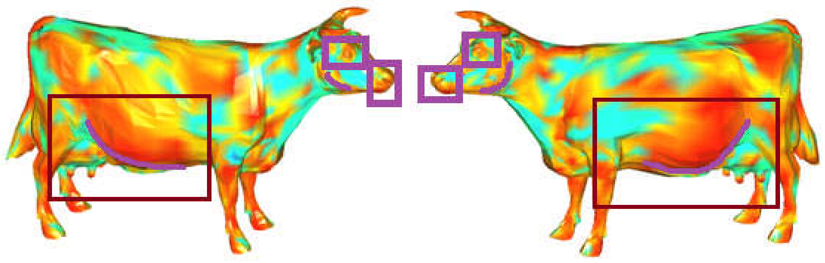
$$H_i = -\sum_{j=0}^{n} p_{i_j} \log p_{i_j} \qquad (3.1)$$

Where$p_{i_j} = \frac{k_{i_j}}{\sum_{j=0}^{n} k_{i_j}}$ and we normalize it by computing$C_i = \frac{H_i}{H_{i,max}} = \frac{-\sum_{j=0}^{n} p_{i_j} \log p_{i_j}}{\log(n+1)}$to generate its value within [0,1].

Let$I(v_i)$ denotes the visual importance of the vertexv_i, and we define it as:

$$I(v_i) = C_i \qquad (3.2)$$

The visual importance of the vertex is based on the distribution of curvatures of all these vertices in its adjacent planes. $p_{i_j} = \frac{k_{i_j}}{\sum_{j=0}^{n} k_{i_j}}$ represents the geometry importance (i.e. feature or information) of vertexv_i with respect to the region it locates.C_i (i.e. normalizedH_i)measures the average geometry information the region contains. The larger the visual importance is, the region around this vertex contains more geometry information, e.g. a relatively big variation, so visually we will pay more attention to the content of this surface area. So the visual importance of the vertex continuously reflects the local feature of the mesh model. As an example, Fig. 1 shows the visual map of the cow model which is measured by using Eqn. (3.2).



(a)Front (b)Back

Fig. 1: Visual map of the cow model. Warmer colors (reds and yellows) show high visual importance and cooler colors (greens and blues) show low visual importance.

We can see from Fig. 1 that this visual importance metric can capture the visually significant features and surface variations of the model, e.g. eyes, nose, silhouette of mandible and abdomen of the cow model.

4 SIMPLIFICATION ALGORITHM

We combine visual importance measure and the basic quadric error metric to formulate a new visual and geometry-based hybrid approach for polygonal mesh simplification. Obviously, visually important surface regions should be preserved in the early simplification process in order to produce simplified models with high fidelity.

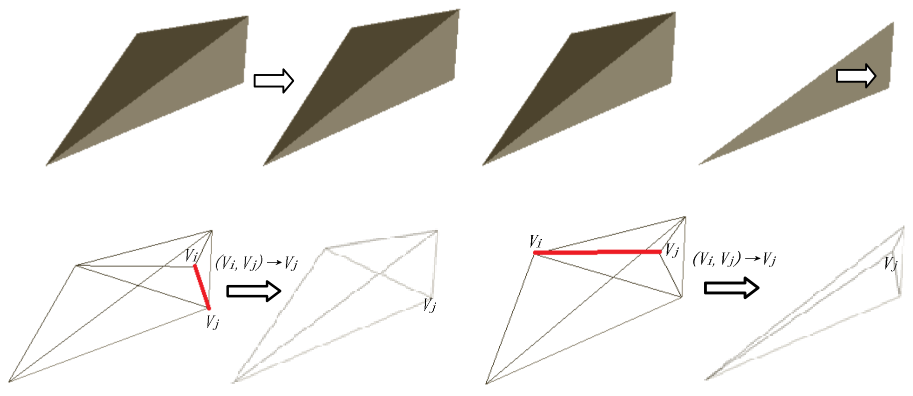
4.1 Error Metric

Let$E(v)$denotes the edge collapse error in our algorithm. For vertex$v$, we define it according to Eqn. (2.2) as:

$$E(v) = v^T(I(v) * Q)\, v \qquad (4.1)$$

This metric not only measures the geometry differentiation of the surface model but also the local surface variation by multiplying the quadric error$Q$of each vertex by its visual importance. Vertices with high visual importance will have higher edge collapse costs while vertices with low visual importance will have lower edge collapse costs. The differentiation between vertices with high visual importance and the ones with low visual importance will be increased. The characteristic surface features are mostly concentrated in regions with high visual importance, therefore the characteristic surface features are retained by using this error metric.

By using our error metric, the order of edge collapses is used to generate a simplified surface model with a low geometric and visual error. As an example, one step of the simplification process of a simple model with 5 vertices and 6 faces using our error metric is shown in Fig. 2(a) and the quadric error metric [8] is shown in Fig. 2(b). The wireframe of the models are shown in the bottom row of Fig. 2(a) and (b). The highlighted red edges are the edges to be collapsed at this iteration. v_i andv_j are the end points of the edges. In this example, we use the half-edge collapse, i.e. $(v_i, v_j) \rightarrow v_j$.From Fig. 2, we can see that our algorithm changes the order of edge collapses and the simplified model generated by using our error metric is visually almost the same as its original model and its geometric error measured by Metro [4], a public tool that evaluates the difference between the original mesh model and its simplified representation, is 0.028338, while the geometric error of the simplified model in Fig. 2(b) is 0.047106.



(a)Our error metric (b)Quadric Error Metric

Fig. 2: Comparison of simplified model using our error metric and quadric error metric.

4.2 The Algorithm

Our simplification algorithm is based on half-edge contractions. Original model$M$is represented using an adjacency graph structure, which stores a table of vertices and encodes edges and faces as doubles and triples of indices into the vertex table. In practice, the index representations are more flexible even though memory access is indirect: using indices into vertices enables efficient memory relocation and simpler and more compact memory management.So the space complexity in our algorithm is proportional to the complexity of the model or the number of vertices in the model just as QSlim's method [8]. In our algorithm scheme, we use simple vertex curvature estimation which only needs traversal of the vertex table and gets information adjacent to a vertex from the system adjacency graph structure. The running time in our algorithm is also proportional to the complexity of the model.

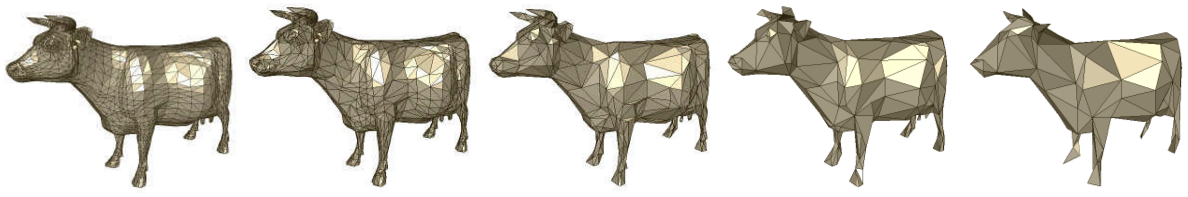
Our algorithm is as follows: at first, the visual importance I and Q matrices are computed for all the initial vertices. Then we compute the contraction target for the edge (v_i, v_j) by calculating the two possibilities $E(v_i) = v_i^T(I(v_i) * Q_i + I(v_j) * Q_j)\, v_i$ and $E(v_j) = v_j^T(I(v_i) * Q_i + I(v_j) * Q_j)\, v_j$ and applying the direction $(v_i \to v_j)$ if $E(v_i) < E(v_j)$ or $(v_j \to v_i)$ if $E(v_i) > E(v_j)$. The algorithm iteratively contracts the pair with the minimum contraction cost. In each iteration after an edge collapse, we choose a small group of edges that are affected by the edge collapse and then recalculate the cost for these edges that are adjacent to the vertices belonging to the collapsed edge. The algorithm can be summarized as:

```
/*Build an initial priority queue for edge collapses, M represents the mesh model*/
For($e \in M$)
    Determine contraction target of edge $e = (v_i, v_j)$
Compute edge collapse cost $E$
    Insert the duple $(e, E)$ in queue $q$
End for
/*Update the mesh*/
While (queue $q$ not empty)
    Delete the edge $e$ with the lowest cost $E$ from the queue
    Perform the collapse of edge $e$ and update the triangles adjacent to edge $e$
    Recalculate the cost of every edge in the neighborhood of edge $e$ and update the queue $q$
End while
```

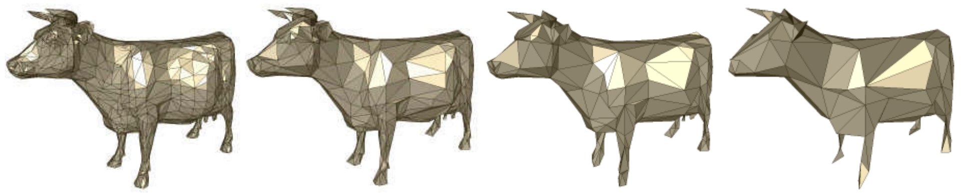
5 RESULTS

We have implemented our algorithm using C++ language and carried out tests with various models including low complexity models from CAD programs. All the experimental results are obtained by running the algorithm on a 2.4 GHz machine with 4.00 GB RAM. We compare our results obtained at the same simplification level to the results with QSlim [8], using the half-edge collapse. Some aspects such as geometry error, visual quality and speed are analyzed in the comparison.

Simplification results of the cow, car and helicopter models without boundaries using our algorithm and QSlim are shown in Fig. 3, Fig. 4 and Fig. 5. Results using our algorithm are shown on the top row of Fig. 3, Fig. 4and Fig.5 while the results using QSlim's methodare shown on the bottom row. In Fig. 3, where the cow model is simplified to 50%, 80%, 90% and even 95% of the original cow mesh model, the simplified shapes still look the same as its original one. Meanwhile, at different simplification levels, some visually important features such as the eye, nose, silhouette of mandible and abdomen of the cow model are better preserved than those in models simplified with QSlim. By adjusting the order of edge collapses, our algorithm can generate simplified models with better mesh quality than that generated withQSlim. The same analysis can also be obtained from the results of Fig. 4 and Fig. 5. For example, in the car model the contour and mesh are better preserved, and in the helicopter model the same can be said for the main body and the wings.



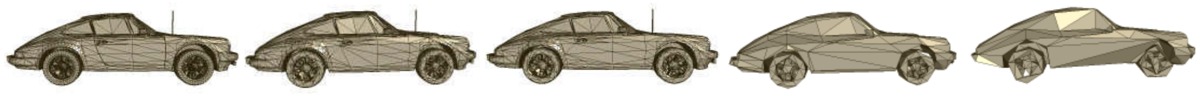
(a)Simplification by our algorithm



50% simplified (2914 triangles) 80% simplified (1180 triangles) 90% simplified (602 triangles) 95% simplified (292 triangles)

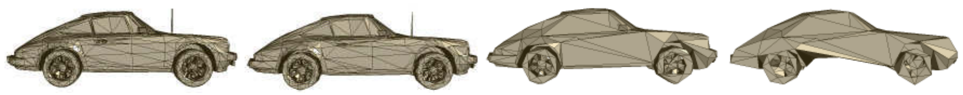
(b)Simplification by QSlim

Fig. 3: Simplification results of cow models using our algorithm and QSlim.



Original50% simplified 70% simplified 90% simplified 95% simplified
(10474 triangles) (5240 triangles) (3142 triangles) (1056 triangles) (530 triangles)

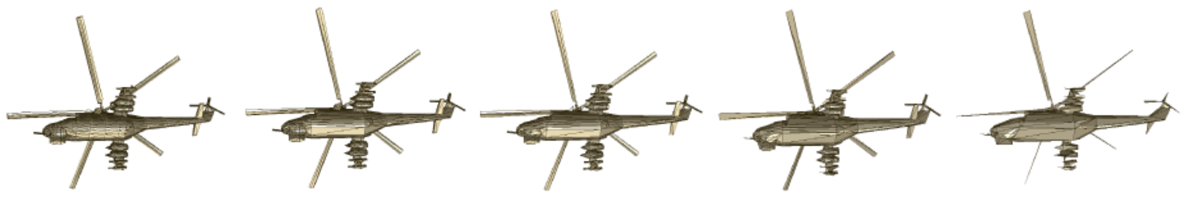
(a)Simplification by our algorithm



50% simplified (5240 triangles) 70% simplified (3142 triangles) 90% simplified (1056 triangles) 95% simplified (530 triangles)

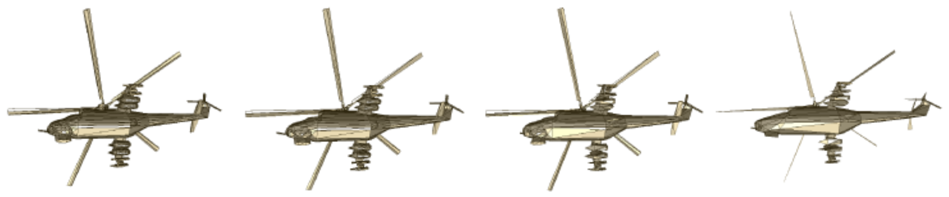
(b)Simplification by QSlim

Fig. 4: Simplification results of car models using our algorithm and QSlim.



Original (6448 triangles) 50% simplified (3247 triangles) 70% simplified (1920 triangles) 80% simplified (1358 triangles) 90% simplified (698 triangles)

(a)Simplification by our algorithm

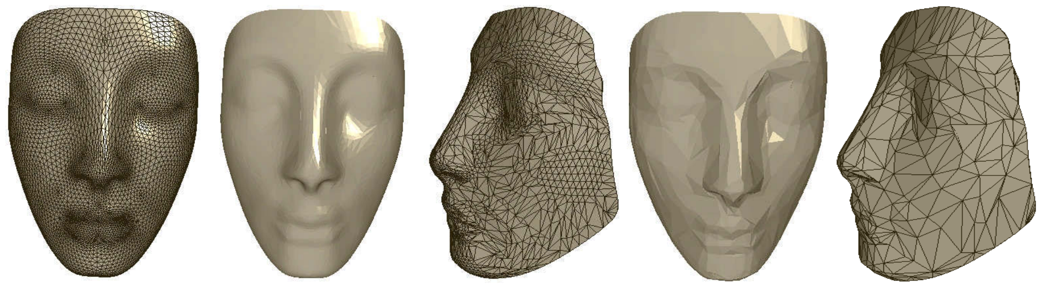


50% simplified (3247triangles) 70% simplified (1920 triangles) 80% simplified (1358 triangles) 90% simplified (698 triangles)

(b)Simplification by QSlim

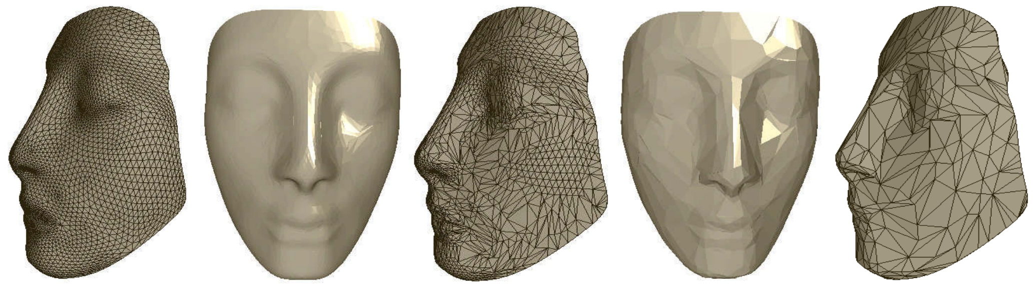
Fig. 5: Simplification results of helicopter models using our algorithm and QSlim.

Fig. 6 shows the simplification results of the face model which has boundaries on the mesh. We can see that at different simplification levels, our algorithm can preserve the boundaries of the model as well as QSlim [8]. However, our algorithm can generate more regular simplified mesh than QSlim.



Original 50% simplified 90% simplified

(a)Simplification by our algorithm



Boundary illustration 50% simplified 90% simplified

(b)Simplification by QSlim

Fig. 6: Simplification results of face models with boundaries using our algorithm and QSlim.

In Tab. 1 we present the geometric error committed in our experiments. These geometric errors are measured by Metro [4], a public tool that evaluates the difference between the original mesh model and its simplified representation. We can see that the cow, car and helicopter models generated by our algorithm give smaller geometric error than those of QSlim at the same simplification level respectively mainly because our algorithm preserves the visually important features better and generates simplified models with better mesh quality than QSlim.

Mesh Model	Primitives	Simplified (%)	Error	
			Our Algorithm	QSlim
cow	5804	95%	0.133618	0.151724
car	10474	95%	0.011453	0.028257
helicopter	6448	90%	0.031697	0.039431

Tab. 1: Comparison of the geometry error of our algorithm with QSlim.

Tab. 2 summarizes the running time of our algorithm and QSlim using the models shown in this paper. This time is proportional to the complexity of the model and the number of triangles in the simplified model. Initialization time includes loading and displaying the mesh model, computing visual importance and initial quadric error matrices, calculating edge collapse costs and building an edge collapse list. Simplification time includes the iterative contraction of edges. In Tab. 2, the simplification time is calculated by simplifying all the models to 50% of their original mesh models. We can see that the QSlim algorithm is faster in the initializationstage, however, our algorithm takes shorter time in the simplification stage. The reason is our algorithm will compute vertex curvature and entropy during the initialization stage whichis more time consuming.

Mesh Model	Primitives	Init(s)		Simplify(s)	
		Ours	QSlim	Ours	QSlim
cow	5804	13.089	5.335	0.281	0.327
car	10474	32.183	9.719	0.515	0.515
helicopter	6448	14.836	5.709	0.297	0.343

Tab. 2: Comparison of the running time of our algorithm with QSlim.

6 CONCLUSION

In this paper, we present a visual and geometry-based hybrid approach for simplifying polygonal mesh models. We define the visual importance for a vertex to reflect the surface variation around the vertex. From the example we have shown in this paper, one can see that our model of visual importance is able to capture what usually classified as interesting regions on a mesh. Not all such regions necessarily have high curvatures. By combining the visual importance and the quadric error metric, we propose a new error metric to guide the mesh simplification process. Experimental results show that our algorithm performs simplification with lower visual and geometric error than QSlim.

ACKNOWLEDGEMENT

This work is partially supported by a grant from the University Grant Council of the Hong Kong Special Administrative Region (No. 412508).

REFERENCES

[1] Alliez, P.; Cohen, S.; Devillers, O.; Levy, B.; Desbrun, M.: Anisotropic Polygonal Remeshing, Proceedings of ACM Siggraph, 3(22), 2003, 485-493.
[2] Bernd, H.: A Data Reduction Scheme for Triangulated Surfaces, Computer Aided Geometric Design, 11(2), 1994, 197-214. doi:10.1016/0167-8396(94)90032-9
[3] Castello, P.; Sbert, M.; Chover, M.; Feixas, M.: Viewpoint Entropy-driven Simplification, Proceedings of WSCG 2007, 249-256.
[4] Cignoni, P.; Rocchini, C.; Scopigno, R.: Metro: Measuring Error on Simplified Surfaces, Computer Graphics Forum,17(2), 1998, 167-174. doi:10.1111/1467-8659.00236
[5] Cohen, J.; Olano, M.; Manocha, D.: Appearance Preserving Simplification, Proceedings of SIGGRAPH'98, 32, 1998, 115-122.

[6] Cohen, J.; Varshney, A.; Manocha, D.: Simplification Envelopes, Computer Graphics (SIGGRAPH' 96 Proceedings), 1996, 119-128.

[7] Cover, T. M.; Thomas, J. A.: Elements of Information Theory, Wiley Series in Telecommunications, 1991. doi:10.1002/0471200611

[8] Garland, M.; Heckbert, P.: Surface Simplification Using Quadric Error Metrics, Computer Graphics (SIGGRAPH' 97 Proceedings), 1997, 209-216.

[9] Hamann, B.: Curvature Approximation for Triangulated Surfaces, Computing Supple, 13(8), 1993, 139-153.

[10] Hoppe, H.: Progressive Meshes, In SIGGRAPH' 96 Proceedings, 1996, 99-108.

[11] Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J; Stuetzle, W.: Mesh Optimization, In SIGGRAPH' 93 Proceedings, 1993, 19-26.

[12] Lee, C. H.; Varshney, A.; Jacobs, D.: Mesh Saliency, ACM Trans. Graph. 24, 3, 2005, 659-666.

[13] Lindstrom, P.; Turk, G.: Image-driven simplification, ACM TOG 19, 3, 2000, 204-241.

[14] Luebke, D.; Erikson, C.: View-dependent Simplification of Arbitrary Polygonal Environments, In SIGGRAPH'97 Proceedings, 1997.

[15] Luebke, D.; Hallen, B.: Perceptually-driven Simplification for Interactive Rendering, Proceedings of the 12th Eurographics Workshop on Rendering Techniques, 2001, 223-234.

[16] Meyer, M.; Desbrun, M.; Schoroder, P.; Barr, A. H.: Discrete Differential-geometry Operators for Triangulated 2-manifolds, In Visualization and MathematicsIII (Proceedings of VisMath 2002), Springer Verlag, Berlin, Germany, 35-54.

[17] Rossignac, J.; Borrel, P.: Multi-resolution 3D Approximations for Rendering Complex Scenes, Modeling in Computer Graphics: Methods and Applications, 1993,455-465.

[18] Schroeder, W. J.; Zarge, J. A.; Lorensen, W. E.: Decimation of Triangle Meshes, Computer Graphics (SIGGRAPH' 92 Proceedings), 26(2), 1992, 65-70.

[19] Soucy, M.; Laurendeau, D.: Multiresolution Surface Modeling based on Hierarchical Triangulation, Computer Vision and Image Understanding, 63(1), 1996, 1-14. doi:10.1006/cviu.1996.0001

[20] Wei, J.; Lou, Y.: Feature Preserving Mesh Simplification using Feature Sensitive Metric, Journal of Computer Science and Technology, 25(3), 2010, 595-605. doi:10.1007/s11390-010-9348-7

[21] Wu, J. H.; Hu, S. M.; Tai, C. L.; Sun, J. G.: An Effective Feature-preserving Mesh Simplification Scheme Based on Face Constriction, Proceedings of Pacific Graphics, 2001, 12-21.

[22] Wu, Y.; He, Y.; Cai, H.: QEM-based Mesh Simplification with Global Geometry Features Preserved, Computer Graphics (SIGGRAPH' 04 Proceedings), 2004, 50-57.

[23] Zhang, E.; Turk, G.: Visibility-guided Simplification, Proceedings of IEEE Visualization, 31, 2002, 267-274.