

# Mesh Simplification Method Using Principal Curvatures and Directions

V. Ungvichian<sup>1</sup> and P. Kanongchaiyos<sup>1</sup>

**Abstract:** This paper describes an enhancement to Garland and Heckbert's mesh simplification method by using the principal curvatures and directions of each vertex. We calculate the values and directions, before using them to determine the absolute normal curvature in the direction of contraction, and multiplying the curvature with the edge length, the maximum absolute cosine of the angles between the edge and the normals of faces adjacent to either endpoint, and the quadric error of the collapse. We also apply penalties based on compactness and angular and dihedral deviations of the resulting faces. We have implemented these improvements and tested our algorithm on a sample of models from Purdue's Engineering Shape Benchmark. We observe that, while our algorithm tends to produce competitive Hausdorff distances than QEM up to 20% face count, and reduces models to between 20% and 50% of the original face count before significant distortion occurs (at a Hausdorff distance of approximately .05 of the bounding box diagonal), QEM still performs better at more drastic levels of simplification, especially on meshes with already low face count. Future research includes, among others, improving the factors to be more robust towards changes in the model during the simplification process.

**Keywords:** Three dimensional computer graphics, Mesh simplification, Curvature, Edge contraction

## 1 Introduction

Three-dimensional computer graphics have been widely used in various applications, such as simulation and animation, due to their suitability in representing objects in a realistic fashion and better practicality than actual physical objects. Along with the general capabilities of computer graphics hardware, the quality and complexity of three-dimensional polygonal models has increased. The high number of faces found in current 3D models makes them not only more detailed and

---

<sup>1</sup> Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.

realistic, but also more complex than can be efficiently rendered under the inherent limitations of typical graphics hardware. In many applications, rendering efficiency is more important than detail, for example, gaming, which requires real-time user interaction; also, some game engines set a limit on the number of polygons permissible in a single model. There are also cases where fine detail is unnecessary: For example, for various forms of analysis of objects for engineering, a face count in the thousands is sufficient for reasonably accurate results. In any case, it is necessary for such lower-polygon models to resemble the original model. Depending on the usage, either a visual or geometric resemblance is necessary.

Therefore, a commonly-researched topic in computer graphics research is determining methods to reduce the number of faces in the model to a much smaller number, while retaining as much visual (and/or geometric) resemblance to the original model as possible. This process is known as *mesh simplification*, and many methods and schemes have been proposed. The following paper presents a general-purpose mesh simplification method based on principal curvature directions.

## 2 Related work

Among the many mechanisms that have been described for mesh simplification (Luebke (2001)), the most commonly-used mechanism is edge contraction, in which edges that are deemed unimportant to the overall shape of the model are contracted, removing degenerate triangles. Such methods are well-suited for structures based on dynamic reduction, for example, Hoppe's Progressive Meshes (Hoppe (1996)), as the edge contraction steps can be stored as part of the structure, and then used to re-create the mesh at a given face count, providing an advantage over static reduction. One of the most notable and widely-used edge-contraction-based algorithms is Garland and Heckbert's Quadric Error Metric (Garland and Heckbert (1997)), or QEM, which uses  $4 \times 4$  matrices to score contractions based mainly on the distances of the resulting vertex from the original faces. The QEM method has been noted for providing a good trade-off between the quality of results and algorithm complexity, compared to previous methods.

However, Garland and Heckbert's original metric is based solely on geometric error, which may, for example, result in ambiguity around sharp corners, which may be contracted without introducing much quadric error. Subsequent papers have incorporated other factors into the original algorithm, such as curvedness-like measures (Xu, Chen, Liu, and Lv (2008)) (Li and Zhu (2008)), torsion (Jong, Tseng, and Yang (2006)), optimal placement based on various factors (Choi, Kim, and Lee (2008)), and higher-dimension feature sensitivity metrics (Wei and Lou (2010)), to preserve visual significant features. The lattermost method uses a  $6 \times 6$  matrix, more than doubling memory usage from the standard  $4 \times 4$  matrix, while the other

methods apply the factors to the  $4 \times 4$  matrix, either by multiplying the matrix with a penalty, or adding the penalty to the error score. These penalties all involve a single value per vertex, calculated from the vertex and its immediate locality.

Differential geometry states, however, that the curvature of a surface around a given point involves two principal curvature directions and their respective normal curvatures, which are the maximum ( $k_{\max}$ ) and minimum ( $k_{\min}$ ) curvatures of area around the given point. We also observe from the previous research that areas of high curvature are more important (visually and geometrically) to retain. Therefore, we expect that moving a vertex in a direction of little to no curvature (on a flat surface or a straight feature edge) should affect the overall structure less than moving the same vertex in a high curvature direction. As the principal curvatures and their directions can be used to determine the curvature in a given direction, we believe that using these values may help assist in the improvement of mesh simplification better than using a single value.

Other aspects that we believe influence the quality of a reduced model are facial orientation and regularity. The orientation of a given face (both on its own and relative to adjacent faces) is a major factor in determining its shade of color when rendering the model, while facial regularity (or compactness) is desirable in applications such as finite element analysis and design processes; also, we have observed in early experiments that highly irregular faces may produce high angular deviation without introducing significant geometric error.

### 3 The proposed method

The algorithms described in the previous section calculate a factor for each individual vertex for use in penalizing scores. For our algorithm, we calculate the principal curvatures and directions for each vertex, so that instead of using each individual vertex and its locality to calculate the penalty, we calculate the penalty for each edge, resulting in a lower score for contracting a given vertex in a direction with lower curvature. We assume that the mesh input is a non-textured model with triangular faces, and mostly manifold topology. The steps of our algorithm follow the same general framework as most edge-contraction based algorithms.

After receiving the input, we perform pre-processing by calculating each vertex's principal curvatures, principal directions and quadric matrix. We then determine the validity of contracting each edge, and where valid, the score and resulting vertex, by calculating the QEM score and applying other penalties to obtain the score and inserting the vertex with better score in a heap. We then perform the simplification by contracting the best scores on the heap and updating the heap as necessary until the desired level of simplification has been reached. Each step will now be

explained in detail.

### 3.1 Pre-processing

After receiving the input mesh data, we calculate the principal curvatures and their directions for each vertex, by calculating the normal vectors at each vertex, and then using Batagelo and Wu's method (Batagelo and Wu (2007)) of using linear least squares to estimate the curvature tensor, and then deriving the curvatures from the estimated tensor using eigenvalues and eigenvectors. This method is claimed to be fast and robust, producing fewer outliers than other methods. As the normal vector and two principal directions are all orthogonal, we need only store the normal vector and one principal direction, deriving the remaining direction using a cross product.

We then calculate the quadric matrices for each vertex in a similar fashion to Garland and Heckbert's method. For each face, we calculate its normal vector to determine its corresponding plane  $p$ , and convert  $p$  to a  $K_p$  matrix:

$$K_p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \quad (1)$$

where the plane equation of  $p$  is  $ax + by + cz + d = 0$  and  $a^2 + b^2 + c^2 = 1$ . To each of the face's constituent vertices, we weight  $K_p$  with the product of the area and incident angle on the vertex. After inspecting every face, we perform an area-weighted sum to obtain a quadric matrix  $Q(v)$  for each vertex:

$$Q(v) = \sum_{i=1}^k K_{f_i} \theta_{\langle v, f_i \rangle} w_{f_i} \quad (2)$$

where  $\theta_{\langle v, f_i \rangle}$  is the angle of face  $f_i$  incident on  $v$ ,  $w_{f_i}$  is the area of  $f_i$  (see Fig. 1), and  $K_{f_i}$  is the quadric matrix of the plane on which  $f_i$  lies. Fig. 2 shows the pseudocode for preparing the quadric matrices.

### 3.2 Scoring

Next, we determine how to contract each edge of the model, and the score of the contraction. We first determine the validity of contracting each edge. For each edge  $\langle v_x, v_y \rangle$ , we check the valences of the vertices immediately adjacent to  $v_x$  and  $v_y$ ,

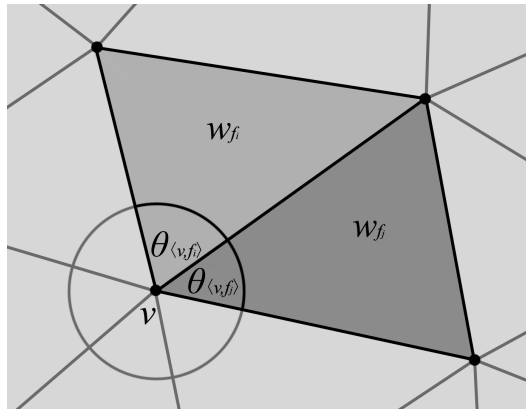


Figure 1: Explanation for weighting using angles

- 1 **for each** face  $f$  in model  $M$
- 2     determine normal vector  $\vec{N}$  and area  $w_f$  of  $f$
- 3     determine quadric matrix  $K_f$  from  $\vec{N}$
- 4     **for each** vertex  $v$  of  $f$
- 5         determine angle  $\theta_{(v,f)}$  of  $f$  on  $v$
- 6          $Q(v) \leftarrow Q(v) + K_f \theta_{(v,f)} w_f$

Figure 2: Pseudocode for preparing quadric matrices

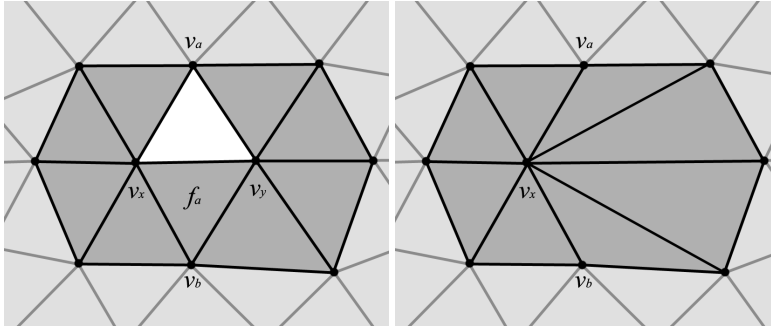


Figure 3: An example of topology being affected by contraction

discarding any edges where any such vertex has a valence of 3. We also determine the number of vertices that are adjacent to both endpoints and the number of faces adjacent to the edge, and discard edges where these two values are not equal. These checks prevent a contraction from altering the topology of the model. For example, in Fig. 3, an edge whose endpoints are adjacent to 2 vertices, but is itself adjacent to one face, is contracted, resulting in a hole being closed up.

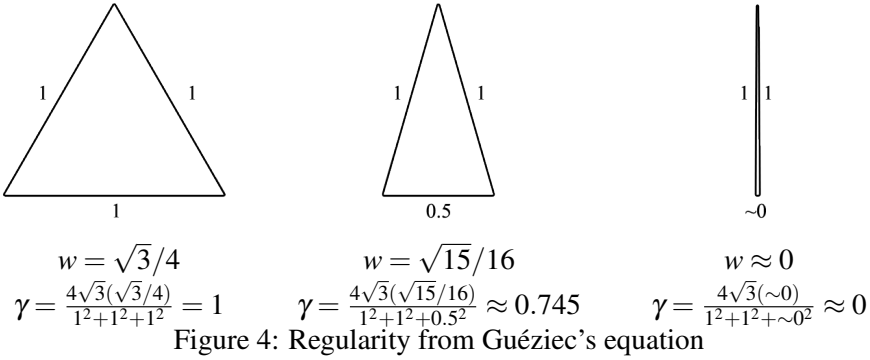
For edges that pass these checks, we then determine the point to which to contract the edge. In our method, we use a subset selection policy (i.e., selecting between the two endpoints), which is easier to implement than determining the optimal position. We first calculate the initial QEM score, starting by summing together the matrices of the edge's endpoints and using the summed matrices to calculate the raw QEM score as in Garland and Heckbert's method:  $v^T(Q(v_x) + Q(v_y))v$ . We next determine the absolute normal curvature from the contracted point to the result point; that is, when contracting  $v_x$  to  $v_y$ , we determine the normal curvature in the direction from  $v_x$  to  $v_y$ . Where  $k_{\max}$  and  $k_{\min}$  are, respectively, the maximum and minimum principal curvatures of  $v_x$ ,  $\vec{N}_{v_x}$  is the normal vector of  $v_x$ , and  $\vec{D}$  is the direction for  $k_{\max}$ , the absolute normal curvature  $k$  from  $v_x$  to  $v_y$  can be calculated as follows:

$$\vec{D}' = (\vec{N}_{v_x} \times (\vec{v}_y - \vec{v}_x)) \times \vec{N}_{v_x} / \|(\vec{N}_{v_x} \times (\vec{v}_y - \vec{v}_x)) \times \vec{N}_{v_x}\| \quad (3)$$

$$\cos^2 \theta = (\vec{D} \cdot \vec{D}')^2 \quad (4)$$

$$k = |\cos^2 \theta k_{\max} + (1 - \cos^2 \theta) k_{\min}| \quad (5)$$

Lastly, for each face  $f$  adjacent to exactly one endpoint of the edge, we determine the absolute cosine of the angle between its normal vector  $\vec{N}_f$  and the edge using



a dot product, and determine the maximum of these values ( $\cos_{\max} \phi$ ), as a lower cosine indicates a lower likelihood of the contraction affecting the surface in the area.

We multiply the raw QEM score with  $k$ ,  $\cos_{\max} \phi$  and the edge length  $\|(\vec{v}_y - \vec{v}_x)\|$ , as we consider that contracting a vertex in a low-curvature direction, in a low-cosine area and/or over a short edge length is less likely to significantly affect the model than contraction in a high-curvature direction and/or over a long edge. We obtain the initial QEM score as follows:

$$s' = (\langle v_x, v_y \rangle, v_x) = \|\vec{v}_y - \vec{v}_x\| \cos_{\max} \phi k v_x^T (Q(v_x) + Q(v_y)) v_x \quad (6)$$

Next, we calculate penalties, starting by calculating the regularity of each face adjacent to the resulting vertex, using Guézic's equation (Guézic (1995)):

$$\gamma = \frac{4\sqrt{3}w}{l_1^2 + l_2^2 + l_3^2} \quad (7)$$

where  $w$  is the area of a given face, and the  $l_i$  are the lengths of its edges. This equation produces a result of 1 for an equilateral triangle, and 0 for a degenerate triangle (Fig. 4). We determine the face with the least regularity, and penalize contractions that result in faces with lower than 0.5 thusly:

$$p_{reg} = \begin{cases} \left(\frac{0.5}{\gamma_{\min}}\right)^2 - 1 & \text{if } \gamma_{\min} \leq 0.5 \\ 0 & \text{if } \gamma_{\min} > 0.5 \end{cases} \quad (8)$$

Our choice of 0.5 allows for some degree of variance in facial regularity from being perfectly equilateral. Next, we determine the orientation of the resulting faces, and compare with the face's original orientation to determine the overall change in orientation  $\theta$ , before penalizing according to the largest change in orientation thusly:

$$p_{ang} = \begin{cases} \frac{\sin \theta_{max}}{1 - \sin \theta_{max}} & \text{if } \theta_{max} < 90^\circ \\ +\infty & \text{if } \theta_{max} \geq 90^\circ \end{cases} \quad (9)$$

We then compare the resulting orientation of each resulting face to the normal vectors of its vertices and calculate their curvedness-inverse-weighted average, as we consider that normal vectors at areas of low curvedness are more representative of the ideal facial orientation than those at areas of higher curvedness:

$$R_{v_i} = \sqrt{k_{min}(v_i)^2 + k_{max}(v_i)^2} / \sqrt{2} \quad (10)$$

$$\theta' = \frac{\sum \frac{\angle(\vec{N}_f, \vec{N}_{v_i})}{R_{v_i}}}{\sum \frac{1}{R_{v_i}}} \quad (11)$$

where  $v_i$  are the constituent vertices of the resulting face  $f$ ,  $k_{min}(v_i)$  and  $k_{max}(v_i)$  are the principal curvatures of each  $v_i$ ,  $R_{v_i}$  is the curvedness of each  $v_i$ , and  $\angle(\vec{N}_f, \vec{N}_{v_i})$  is the angle between the normal vector of  $f$  and the normal vector of  $v_i$ . We then apply the same penalizing method as above:

$$p_{avg} = \begin{cases} \frac{\sin \theta'_{max}}{1 - \sin \theta'_{max}} & \text{if } \theta'_{max} < 90^\circ \\ +\infty & \text{if } \theta'_{max} \geq 90^\circ \end{cases} \quad (12)$$

Lastly, we also consider the dihedral angles between the affected faces, and faces immediately adjacent to such faces, and compare them to the dihedral angles of the original orientations. The faces being compared need not have been originally adjacent to each other. We also consider that cases where the relative orientation between a given pair of faces has changed from concave to convex, or vice versa, produce subjectively "bad" results, hence, we detect and penalize such cases:



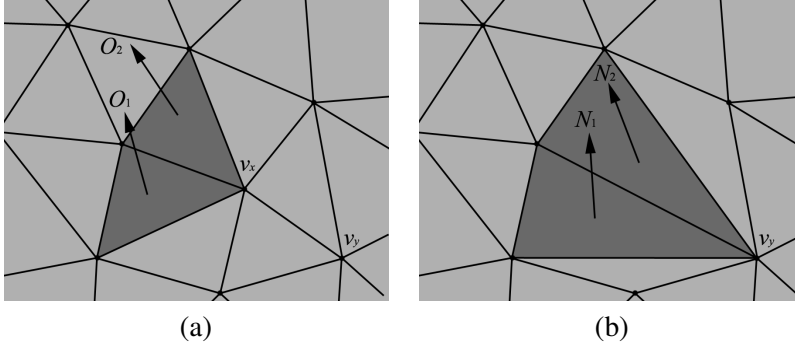


Figure 5: Dihedral angles between two faces before (a) and after contraction (b)

$$\hat{\theta}(\vec{O}_1, \vec{O}_2, \vec{N}_1, \vec{N}_2) = \begin{cases} |\sin^{-1}(\|\vec{N}_1 \times \vec{N}_2\|) - \sin^{-1}(\|\vec{O}_1 \times \vec{O}_2\|)| & \text{if } (\vec{N}_1 \times \vec{N}_2) \cdot (\vec{O}_1 \times \vec{O}_2) > 0 \\ 2\sin^{-1}(\|\vec{N}_1 \times \vec{N}_2\|) + \sin^{-1}(\|\vec{O}_1 \times \vec{O}_2\|) & \text{if } (\vec{N}_1 \times \vec{N}_2) \cdot (\vec{O}_1 \times \vec{O}_2) \leq 0 \end{cases} \quad (13)$$

Where  $\vec{O}_1$  and  $\vec{O}_2$  are the original unit normal vectors of two adjacent faces, and  $\vec{N}_1$  and  $\vec{N}_2$  are the unit normal vectors of the same faces after the contraction (Fig. 5). We then use the highest dihedral angle score to calculate the penalty, in the same way as the previous angles above:

$$p_{rdc} = \begin{cases} \frac{\sin \hat{\theta}_{\max}}{1 - \sin \hat{\theta}_{\max}} & \text{if } \hat{\theta}_{\max} < 90^\circ \\ +\infty & \text{if } \hat{\theta}_{\max} \geq 90^\circ \end{cases} \quad (14)$$

After calculating the penalties, we take the logarithm of the initial QEM score, and add the penalties thusly:

$$s(\langle v_x, v_y \rangle, v_x) = \log(s'(\langle v_x, v_y \rangle, v_x)) + \alpha(p_{ang} + p_{avg} + p_{rdc}) + \beta p_{reg} \quad (15)$$

where  $\alpha$  and  $\beta$  represent user-defined weights for each part of the score. In effect, this multiplies the QEM score with various powers of the calculated penalties. After we have obtained the final scores of contracting the edge to each endpoint, we insert the better score into a binary priority heap  $H$ , along with the following information:

the two endpoints (with the better-scoring endpoint listed first), the edge’s label ( $e_x$ ), and the update number when it was most recently updated ( $U(e_x)$ ), initially  $-1$ . These values are used to assist in the updating process.

### 3.3 Simplification and updating process

As in the QEM method, we repeatedly contract the edge with the least score, sum together the endpoints’ quadric matrices to produce the new vertex’s quadric matrix, and update the priority heap, until the desired level of simplification has been reached, or no valid contractions remain. When updating the priority heap, we use a “lazy” updating scheme, which updates the heap only as required, and only update the topmost portion, under the reasoning that scores in the topmost portion are more likely to remain near the top afterwards. This scheme has similarities to Cohen et al.’s “dirty bit” method (Cohen, Manocha, and Olano (1997)) (which can be described as a different case of our proposed updating scheme) and Wu and Kobbelt’s Multiple-Choice Techniques (Wu and Kobbelt (2002)). From our experiments, as a balance between updating more entries than necessary, and updating more frequently, we have chosen to update the top  $n - 6$  levels of the heap, which can be shown to contain between  $|H|/64$  and  $|H|/128$  entries, where  $|H|$  is the size of the heap.

As in Cohen’s method, we only update the heap when we encounter an edge whose score needs to be updated. We determine this by checking whether either of the edge’s endpoints have been affected since the most recent update of the score; that is, if the score of contracting the given endpoint may have changed due to changes in the immediate area. To achieve this, we maintain a counter  $U$  and increment it each time we perform an update, and store information on the update number when each vertex has been most recently affected ( $U(v)$ ). We consider that all vertices in triangles adjacent to the contracted vertex, or any triangle adjacent to such a triangle are affected. In other words, when contracting  $\langle v_x, v_y \rangle$  to  $v_y$  (Fig. 6):

$$\forall f \in F(v_x) : \forall v \in f : U(v) \leftarrow U + 1 \tag{16}$$

$$\forall f' \text{ adjacent to any } f \in F(v_x) : \forall v \in f' : U(v) \leftarrow U + 1 \tag{17}$$

$$U(v_x) \leftarrow U + 1, U(v_y) \leftarrow U + 1 \tag{18}$$

where  $F(v_x)$  is the set of faces adjacent to  $v_x$ . Our justification is as follows: All faces in  $F(v_x)$  change shape (possibly becoming degenerate) and orientation; therefore, any contractions involving vertices adjacent to these faces need their scores recalculated. Because of our consideration of dihedral angles, the same is also true

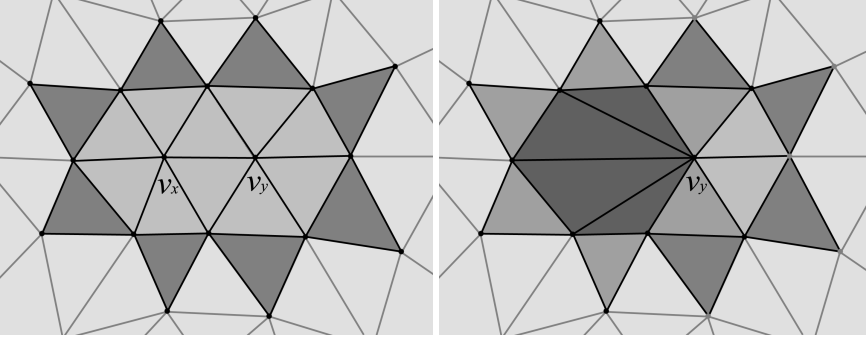


Figure 6: Before (*left*) and after (*right*) edge contraction: affected vertices marked in black

of any contractions involving vertices of faces adjacent to those in  $F(v_x)$ . Lastly, both  $v_x$  and  $v_y$  are affected, as  $v_x$  has been contracted (requiring the contractions of all adjacent edges to be updated), and  $v_y$  has a different  $F(v)$  and  $Q(v)$  than before. Note that while we update  $U(e_x)$  with the current update number  $U$  when we update a score for  $e_x$ , we set the  $U(v)$  of affected vertices to  $U + 1$ . This is so that when two contractions affecting the same vertex are encountered between updates, only the first will be performed, and the second will trigger another update. We also take advantage of our use of affected vertices to update only scores that require updates, in a similar fashion to detecting that the top edge needs updating. Going through the heap, we detect and discard duplicate edges, and update scores of edges where either endpoint has been affected since its most recent update, by re-calculating the score in the same way as during initialization (including topology checks). To assist in the update process, we use the following rules to track geometric changes:

*Point-face relationships:* When collapsing  $\langle v_x, v_y \rangle$  to  $v_x$ , the faces adjacent to exactly one of the endpoints become the faces adjacent to the remaining vertex, while faces adjacent to both endpoints (i.e., to the edge) become degenerate and can be removed; that is:

$$F(v_x) \leftarrow F(v_x) \cup F(v_y) - (F(v_x) \cap F(v_y)) \quad (19)$$

*Vertex mappings:* We use a many-to-one vertex mapping function  $S$  to convert a given vertex in the original model to its current position (vertex) in the simplified model, allowing us to (for example) determine a face's validity, and use the inverse  $S^{-1}$  to assist in updating. When collapsing  $\langle v_x, v_y \rangle$  to  $v_x$ , vertices that were originally mapped to either vertex are now mapped to  $v_x$ , that is, the resulting inverse is

the union of the two sets:

$$S^{-1}(v_x) \leftarrow S^{-1}(v_x) \cup S^{-1}(v_y) \quad (20)$$

$$\forall v \in S^{-1}(v_x) \cup S^{-1}(v_y) : S(v) \leftarrow v_x \quad (21)$$

To save on memory usage, we remove the quadric matrices and  $S^{-1}$  and  $F$  sets from contracted vertices immediately after contraction. It can be shown that with this method, the total memory usage for the  $S^{-1}$  sets remains constant throughout the execution, while the total memory usage for the  $F$  sets is proportional to the number of remaining valid faces, as in both cases, each vertex (or valid face) is represented exactly once (or 3 times) in total amongst the sets of  $S^{-1}$  (or  $F$ ).

#### 4 Experiment and results

We have implemented the above algorithm using Visual Basic .NET running on a Pentium Dual Core system with 2 GB RAM, and tested the algorithm on a selection of 15 models from Purdue's Engineering Shape Benchmark, ranging from 1,672 to 79,680 faces (details in Table 1). We reduce each model using our algorithm to 50%, 20%, 10%, 5%, 2%, and 1% (or until no valid contractions remain), and for comparison, we also reduce the models to the same percentages using the QSlim implementation of QEM (using a subset selection policy). To determine the quality of the reduced models, we compare them to the original models by using Cignoni et al.'s Metro program (Cignoni, Rocchini, and Scopigno (1998)) to determine the Hausdorff distance  $d_H$  between the original model and the reduced version (with respect to the bounding box diagonal of the original model). A graph comparing the averaged results from the models is shown in Fig. 7, and examples of good and bad visual results are shown in Figs. 8 and 9, with their corresponding Hausdorff results in Fig. 10. The visual and Hausdorff results from the other tested models are shown in Figs. 11 and 12 respectively.

#### 5 Discussion

A low Hausdorff distance (whether absolute or with respect to bounding box) is vital when evaluating a mesh simplification algorithm. This is especially so for the reduction of models for engineering, as a reduced mesh that has been highly distorted from its original version will not properly reflect the behavior of the original. A suitable limit to the Hausdorff distance is dependent on the structure of the model. Nevertheless, we observe that for most models, most of the reduced models

Table 1: Models used in the experiment

Name	Face count	Name	Face count	Name	Face count
1338386	21,730	2360536	5,360	MISUMI	2,906
1400407	10,750	ARIES147A	84,70	NOZAG_BFI	8,664
1411731	79,680	ASM11	4,322	REAR_PLATE	3,980
1417143	1,344	CD_SPINDLE	6,752	SPACER_67	1,672
1626TOP	5,578	GANTER_DIN	3,700	WORM_GEAR	13,670

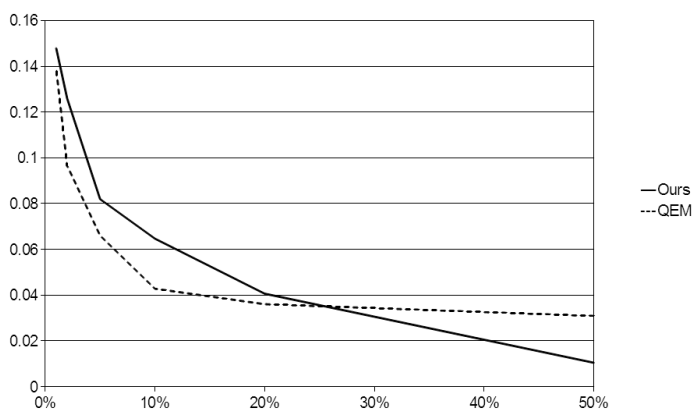


Figure 7: Comparing the average of our results (solid line) and QEM (dashed line)

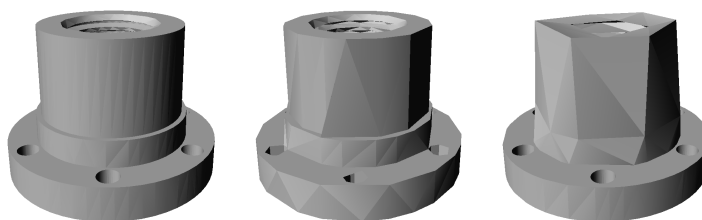


Figure 8: Example of a good result: (left to right) Full Nozag model (8,664 faces), Reduction to 20% with QEM (1,732 faces,  $d_H = 0.1117$ ) and our method (1,726 faces,  $d_H = 0.1010$ )

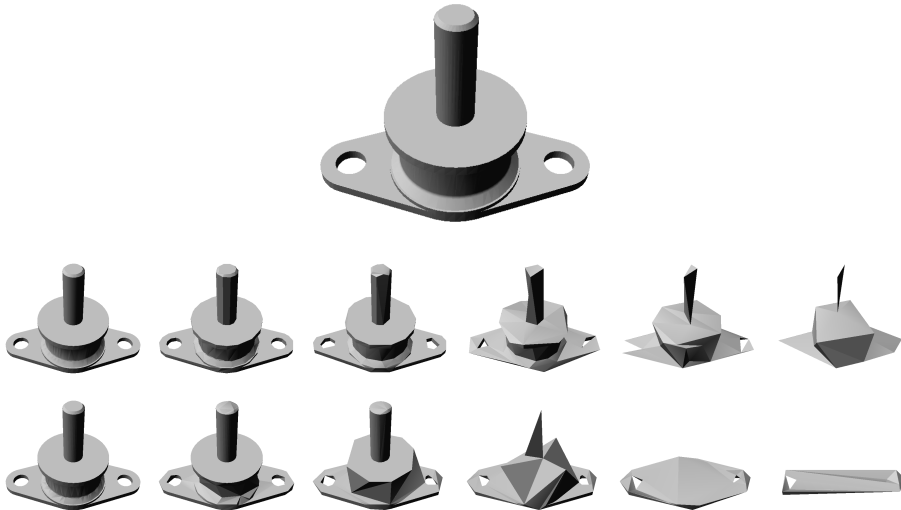


Figure 9: Example of a bad result: (*top*) Full Misumi model (2,906 faces), (*middle*) Reduction with QEM (l-r: 1,452 [ $d_H = 0.0013$ ], 580 [ $d_H = 0.0065$ ], 290 [ $d_H = 0.0157$ ], 144 [ $d_H = 0.0372$ ], 58 [ $d_H = 0.0521$ ], 28 [ $d_H = 0.0819$ ]), (*bottom*) Reduction with our method (l-r: 1,444 [ $d_H = 0.0017$ ], 576 [ $d_H = 0.0295$ ], 288 [ $d_H = 0.0675$ ], 138 [ $d_H = 0.1097$ ], 58 [ $d_H = 0.4517$ ], 30 [ $d_H = 0.4617$ ])

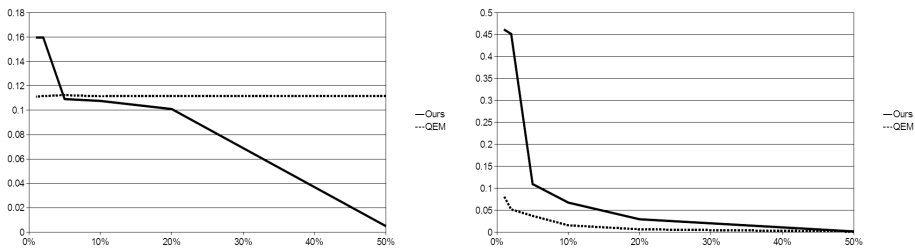


Figure 10: Comparison for Nozag (left) and Misumi (right) models

have a Hausdorff distance  $d_H$  of less than .05 (with respect to bounding box) from the original.

From our results, we observe that on most of the models, the Hausdorff results are competitive with Garland and Heckbert's QEM method in the early stages of reduction (up to 20%), and performs best on models with mostly smooth surfaces. In most cases, we observe that the models can be reduce to 20% or 50% of the original face count without significant distortion. However, we also observe that the algorithm tends to produce higher Hausdorff distances at more drastic levels of simplification, suggesting that although the curvature-related factors that we have chosen can help improve the QEM method initially, these factors become less useful in later stages of reduction, perhaps due to trying to retain topological structure and/or certain features at the cost of other parts of the figure, or changes in the locality around each vertex; for example, the faces adjacent to the vertex covering a larger area than originally used in calculating the curvature, and faces becoming more irregular in later stages. One of the worst cases from our algorithm is shown in detail in Fig. 9, in which the main shaft of the Misumi model is removed completely at the 2% level, while the base retains its "thickness" and holes better.

We also observe that on models with a low face count (with lengthy edges spanning features in different areas), our algorithm tends to exhaust available contractions before our 1% limit (unlike the QEM algorithm), as the algorithm deems that all possible contractions would violate our given restrictions. Conceptually, this should result in final models that are still easily recognizable as the original.

Another concern is that our curvedness-inverse-average-based penalty may not sufficiently take sharp features and overall figure shape into account. Although our logic that normal vectors at areas of high curvedness are less reliable as an indicator of ideal face orientation seems valid on smooth surfaces, we suspect that they may still exert some influence in our algorithm, especially when more than one vertex of a resulting face lies on such an area (for example, a triangle with an edge lying on a feature edge). We are currently considering methods to reduce the probability of such an occurrence resulting in a subjectively "bad" facial orientation.

## 6 Conclusions and future work

Mesh simplification allows for complex 3D models to be handled more easily, for cases such as real-time display and efficient scenery rendering, by reducing the number of faces, making it a popular research topic. Here, we have developed an enhancement to Garland and Heckbert's Quadric Error Metric mesh simplification method that uses the principal curvatures and their directions to provide preference for contracting edges in directions of low curvature, as an alternative to measures



















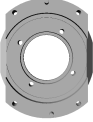

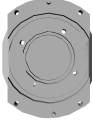




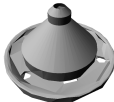







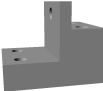
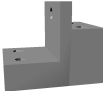




					
1338386	10,858 faces $d_H = .0037$	2,172 faces $d_H = .0367$	1400407	5,360 faces $d_H = .0118$	2,150 faces $d_H = .0331$
					
1411731	39,834 faces $d_H = .00046$	3,984 faces $d_H = .0443$	1417143	268 faces $d_H = .0287$	268 faces $d_H = .0238$
					
1626TOP	2,778 faces $d_H = .0151$	2,788 faces $d_H = .0316$	2360536	2,672 faces $d_H = .0104$	536 faces $d_H = .0226$
					
ARIES147A	4,220 faces $d_H = .0283$	1,694 faces $d_H = .1421$	ASM11	860 faces $d_H = .0576$	432 faces $d_H = .0099$
					
CD_SPINDLE	1,344 faces $d_H = .0349$	1,350 faces $d_H = .0248$	GANTER_DIN	184 faces $d_H = .0291$	184 faces $d_H = .0190$
					
REAR_PLATE	1,990 faces $d_H = .0125$	1,796 faces $d_H = .0445$	SPACER_67	334 faces $d_H = .0484$	334 faces $d_H = .0684$
					
	WORM_GEAR	1,366 faces $d_H = .0168$	1,364 faces $d_H = .0062$		

Figure 11: Other models. For all models (left to right) Full, least acceptable face count with our method, and with QEM



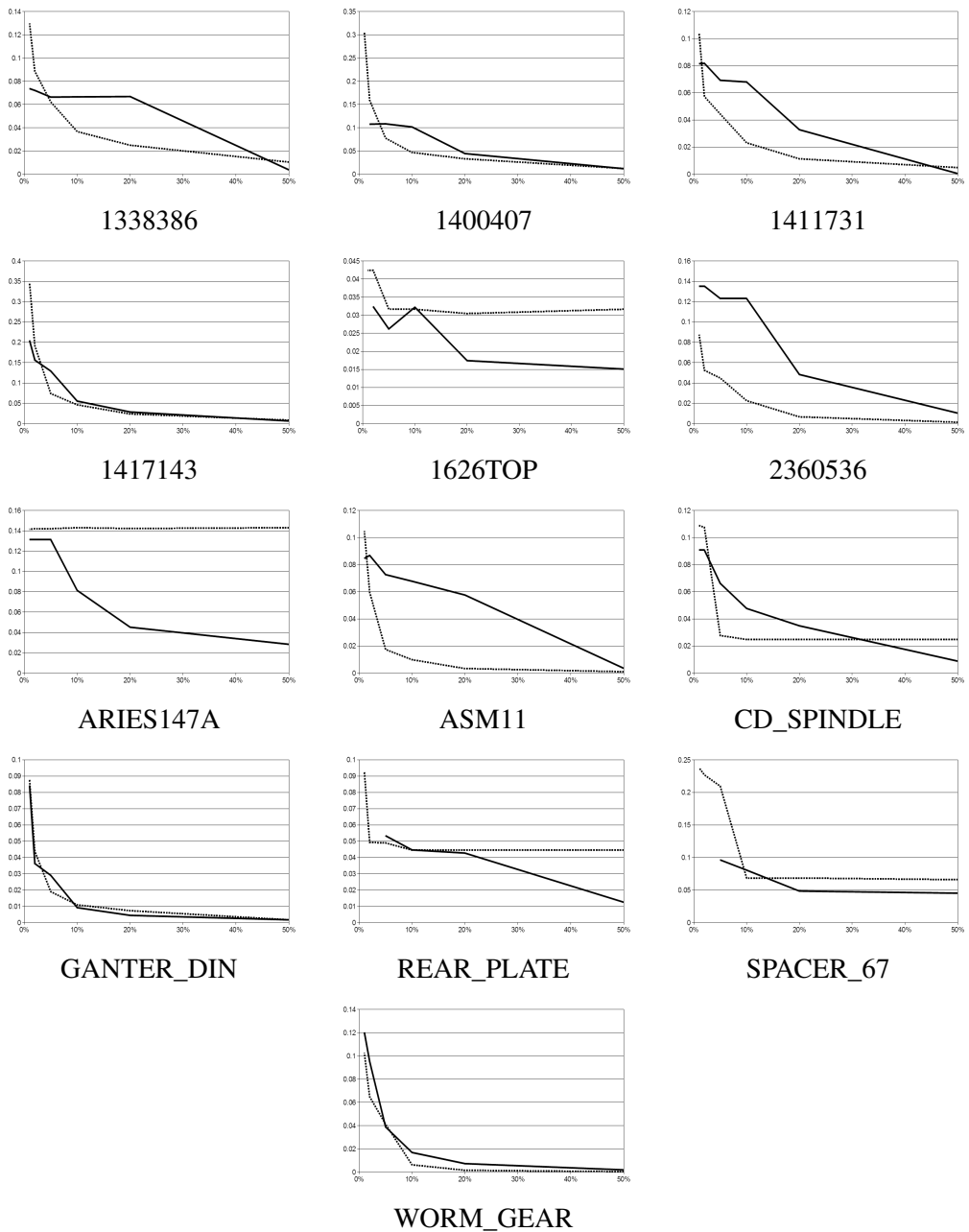


Figure 12: Hausdorff distances graphs for other models: Solid line is our algorithm, dashed line is QEM

that calculate such measures on single vertices.

The Hausdorff distance results and visual renders suggest that this approach produces results competitive to Garland and Heckbert's method at lower levels of reduction, especially on meshes with smooth surfaces; also, our curvature use has the effect of preserving feature areas of high curvature. However, the QEM algorithm still produces significantly better results at more drastic levels, suggesting that the factors we have used to assist the simplification become deficient in those cases.

Future research includes developing factors that are more robust towards changes in the model during simplification as well as lengthy areas of high curvature, along with detecting and taking drastic changes in the locality into account, to allow for improved reduction performance at more drastic levels. Other possibilities include taking the direction of contraction in relation to the normal vector of the contracted vertex into account, which may help to detect major surface changes, and discarding (or otherwise penalizing) normal vectors on vertices with high curvedness when calculating the curvedness-inverse-weighted average, to reduce the possibility of "bad" facial orientations. Another topic to be researched further is making the algorithm work well on meshes with multiple connected components, as our current algorithm may apply higher penalties to contractions on smaller components.

## References

- Batagelo, H. C.; Wu, S.-T.** (2007): Estimating curvatures and their derivatives on meshes of arbitrary topology from sampling directions. *Vis. Comput.*, vol. 23, pp. 803–812.
- Choi, H.; Kim, H.; Lee, K.** (2008): A mesh simplification method using noble optimal positioning. In Chen, F.; Jüttler, B.(Eds): *Advances in Geometric Modeling and Processing*, volume 4975 of *Lecture Notes in Computer Science*, pp. 512–518. Springer Berlin / Heidelberg.
- Cignoni, P.; Rocchini, C.; Scopigno, R.** (1998): Metro: Measuring error on simplified surfaces. Technical Report 2, 1998.
- Cohen, J.; Manocha, D.; Olano, M.** (1997): Simplifying polygonal models using successive mappings. In *Proceedings of the 8th conference on Visualization '97, VIS '97*, pp. 395–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.
- Garland, M.; Heckbert, P. S.** (1997): Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97*, pp. 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

**Guéziec, A.** (1995): Surface simplification with variable tolerance. In *Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)*, pp. 132–139.

**Hoppe, H.** (1996): Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96*, pp. 99–108, New York, NY, USA. ACM.

**Jong, B.-S.; Tseng, J.-L.; Yang, W.-H.** (2006): An efficient and low-error mesh simplification method based on torsion detection. *The Visual Computer*, vol. 22, pp. 56–67.

**Li, Y.; Zhu, Q.** (2008): A new mesh simplification algorithm based on quadric error metrics. In *Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference on*, pp. 528–532.

**Luebke, D. P.** (2001): A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, vol. 21, pp. 24–35.

**Wei, J.; Lou, Y.** (2010): Feature preserving mesh simplification using feature sensitive metric. *J. Comput. Sci. Technol.*, vol. 25, pp. 595–605.

**Wu, J.; Kobbelt, L.** (2002): Fast mesh decimation by multiple-choice techniques. In Greiner, G.(Ed): *VMV*, pp. 241–248. Aka GmbH.

**Xu, L.; Chen, W.; Liu, J.; Lv, T.** (2008): An improved quadric error metrics based on feature matrix. In *RAM*, pg. 582. IEEE.

