Mesh Simplification with Vertex Color

Hyun Soo Kim, Han Kyun Choi, and Kwan H. Lee

Gwangju Institute of Science and Technology
1 Oryong-dong, Buk-gu, Gwangju 500-712, Korea
{hskim,korwairs,khlee}@gist.ac.kr
http://ideg.gist.ac.kr

Abstract. In a resource-constrained computing environment, it is essential to simplify complex meshes of a huge 3D model for visualization, storing and transmission. Over the past few decades, quadric error metric(QEM) has been the most popular error evaluation method for mesh simplification because of its fast computation time and good quality of approximation. However, quadric based simplification often suffers from its large memory consumption. Since recent 3D scanning systems can acquire both geometry and color data simultaneously, the size of model and memory overhead of quadric increases rapidly due to the additional color attribute. This paper proposes a new error estimation method based on QEM and half-edge collapse for simplifying a triangular mesh model which includes vertex color. Our method calculates geometric error by the original QEM, but reduces the required memory for maintaining color attributes by a new memory-efficient color error evaluation method.

Keywords: mesh simplification, level of detail, vertex color, multi-resolution.

1 Introduction

Representing complex 3-dimensional models as triangular mesh has been very popular across a broad spectrum of applications such as reconstructing surfaces, creating objects in virtual reality, and rendering of 3D objects. The triangular mesh of a 3D object can be acquired by a 3D scanning system. With the advance of non-contact optical scanners, it is possible to generate a high-resolution model, which contains millions of points with speed and accuracy. Due to the occlusion problem of the optical scanner, multiple scans are generally required for reconstructing a complete 3D model. The acquisition by scanning operation often leads to the size of model unnecessarily larger. Although the capability of graphics hardware has grown rapidly, the size of models has become too big to be managed by the computing hardware and network. Simplification of a model is necessary to visualize, store and transmit large size models in a resource constrained environment. Over the past few decades, preservation of the geometry of a model has been the main subject of simplification as reviewed in [1]. But nowadays many commercial scanners can obtain both geometry and RGB color attribute of a model simultaneously by using the digital camera techniques. Furthermore, the color of a model becomes more and more important in many application areas such that there has been a renewal of interest in simplifying the model with respect to both geometry and color.

Simplification algorithms have been evaluated in terms of memory consumption, computation time and the quality of approximation. Many algorithms have been published and each one has both advantages and disadvantages in view of the above factors[4]. Some algorithms[3,5] are fast but generate poor results, on the other hands, some methods[6,7] produce nice surfaces but take a long computation time. The QEM based algorithm proposed by Garland[2] has been considered as one of the most popular simplification methods providing a good combination of speed, robustness, and fidelity. But it suffers from large memory overhead. Garland also extended his algorithm in [9] to a color model. However, this caused the use of additional memory for having vertices of the model placed in a 6-dimensional space. This paper proposes a new error estimation method based on QEM and half-edge collapse for simplifying a triangular mesh model which also includes vertex color. Our method calculates geometric error by using the original QEM, but reduces memory consumption for maintaining color attributes.

The rest of the paper is organized as follows. We reviewed some of previous work related to simplification in section 2. Among simplification error metrics, we made an overview of quadric based error metrics in section 3. Section 4 presents the details of our error estimation method. In section 5, we applied our method to various models and discussed the performance of our method by comparing with the original QEM algorithm. Section 6 concludes the paper.

2 Previous Work

The simplification process of meshes generally consists of two parts; one to apply a simplification operator and the other to decide on the priority. Simplification operators reduce the complexity of a mesh by removing geometric primitives of a mesh such as a vertex, edge and triangle, and modifying the connectivity of primitives respectively. The choice of simplification operator depends on the conditions imposed by the computing environment, the ease of implementation, the target application and the data structure of mesh. In the simplification process, it is necessary to decide the priority of mesh primitives for applying the simplification operator in order. The determination of the priority is performed by measuring the error between the original model and the simplified model, where the error measure largely affects the quality of the simplification result.

Among various algorithms that have been used for simplification, some recent simplification methodologies are briefly reviewed below according to the types of simplification operator and the methods of measuring error.

2.1 Simplification Operator

Many simplification operators have been published and used over the past few decades. Researchers[1,10] have surveyed simplification algorithms with respect to simplification operators. We discussed some well-known simplification operators below.

Polygon merging is one of the earliest simplification operators[15,16]. This operator merges nearly coplanar and adjacent polygons into bigger ones. For triangular mesh, re-triangulation is performed after merging the polygons. This gives the equivalent result as using the vertex removal operator[3,17]. Merging criteria are often based on distance measure or surface curvature. Polygon merging has been applied under different names, such as superfaces[15] and face clustering[16]. The polygon elimination operator simplifies a mesh by collapsing a polygon to a single vertex[19]. The size of polygons and surface curvature can be used as the priority for elimination. We can replace a polygon elimination operator by a combination of edge collapses.

The edge collapse operator proposed by Hoppe et al.[19] is the most popular operator for mesh simplification. This operator collapses an edge to a new vertex, preventing the need of re-triangulation. Additionally, faces that include the target edge are also removed after collapse. Fig. 1 depicts an edge collapse operation. When an edge collapses from $\boldsymbol{v_i}$ to $\boldsymbol{v_j}$, the new position of the vertex can be chosen either by calculating the optimal position or by moving to a surviving vertex of the collapsed edge. The edge collapse using the first strategy for determining the new vertex position is called the full-edge collapse, and using the latter strategy is called the half-edge collapse. While the full-edge collapse in Fig. 1(a) requires recalculating the optimal position of the destination vertex additionally, the half-edge collapse just needs to choose one vertex on the collapsed edge in Fig. 1(b). Since the half-edge collapse is easier and more efficient in constructing progressive mesh than the full edge collapse, we used half-edge collapse as our simplification operator in this paper. Progressive mesh can be easily constructed by storing inverse transformation of edge collapse[7].

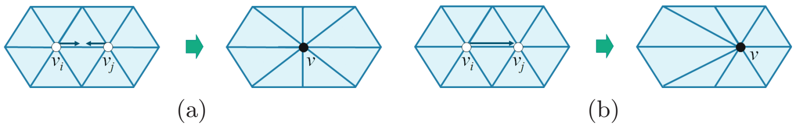


Fig. 1. (a) full-edge collapse (b) half-edge collapse

2.2 Error Measure of Simplification

The determination of the priority in simplification process is made by measuring the error during and after the simplification. The output quality of simplification algorithm generally depends on the error measure. Since most algorithms simplify meshes iteratively, the incremental error in each step by the simplification operator is very important for optimization and implementation. On the other hand, the total error between the original mesh and the simplified mesh is more useful for a good approximation. However, measuring the total error often increases the computing time as the model gets simplified. A variety of errors can be used as simplification criteria. The types of errors used in the previous studies are discussed below.

For a geometric error measure, simplification algorithms incorporated several conceptual elements such as the distance in Euclidean geometry, the size of polygons and the surface curvature. Lindstrom and Turk[8] defined their geometric error as the sum of squared tetrahedral volume. Hussain et al.[4] measured the geometric error by calculating the change of the face normals around a vertex. Other algorithms[3, 13, 14] were based on the distance error metric. The distance based algorithms usually promised speed and ease of implementation. Kobbelt et al.[11] constrained their algorithm by placing a limit on the approximated mesh. Garland and Heckbert proposed the quadric error metric[12] which offers a combination of speed, robustness and good approximation of the surface. The quadric error metric basically computes the sum of squared distances between a point and a set of planes. This algorithm begins by pairs of vertices and collapses those pairs by the priority until it reaches the desired level of simplification. Quadric based simplifications have been widely used in recent years because of its advantages such as the quality of the result mesh, fast computation time and robustness. As such, quadric based methods have created many variations. For example, Kim et al.[20] combined the tangent and curvature error to the original quadric error. Their error metric could preserve the feature region with small geometric error. Recent papers address the problem of representing a multiresolution model from a data structure point of view. [21,22] propose a new data structure for non-manifold simplical meshes in nD Euclidean space. Their data structure is compact and efficient to representing n-dimensional mesh and can be integrated with a multiresolution framework. In [23], Cignoni et al. also try to manage external memory for huge-size meshes by octree based data structure.

Although many researchers have proposed their own geometric error measures, few of them have addressed the simplification of the 3D color model. Garland and Heckbert[9] generalized their earlier QEM scheme[2] to deal with color attributes. They used a 6-dimensional space which consists of 3-dimensional coordinate (x, y, z) of vertices for shape information and additional color triples (r, g, b) of corresponding vertices for color representation. However, this expansion of dimension caused QEM to increase the number of required variables per a vertex from 10 to 28. Another problem of their work was that their algorithm could not handle color discontinuity. Hoppe proposed a more intuitive error metric and reduced the number of variables to 23 in [13], but the memory consumption still remained heavy compared to using only the 3-dimensional coordinate values. To solve the discontinuity problem, he introduced wedges into the data structure. Rigiorli et al.[24] evaluate the simplification error of color mesh in CIE-Luv color space. Certain et al.[8] propose simplification with texture color by using the wavelet transform and Fan et al.[25] take face color into account. These studies are appropriate to handle meshes with texture or face color. However, the raw data from 3D scanners usually consists of geometry and vertex color. Accordingly, our concern is to simplify large-size mesh with vertex color.

3 Quadric Error Metrics

In this section, we briefly review the concept of quadric error metric before introducing our proposed method.

3.1 Quadric for Geometry

Garland's original QEM[2] defines the geometric error as the sum of squared distances between an arbitrary point and a set of planes which contains neighboring faces of a vertex. In a 3-dimensional space, the standard representation of a plane is,

$$\boldsymbol{n}^T\boldsymbol{v} + d = 0 \tag{1}$$

where $\boldsymbol{n} = [a \quad b \quad c]^T$ is a unit normal and d is a scalar constant.

For a vertex $\boldsymbol{v} = [x \quad y \quad z]^T$ with a set of planes, the geometric error can be written as follows

$$E_{planes}(\boldsymbol{v}) = \sum_i D_i^2(\boldsymbol{v}) = \sum_i (\boldsymbol{n}_i^T\boldsymbol{v} + d_i)^2 \tag{2}$$

where i = neighbor triangles

Garland rewrites the squared distance into a new form as follows:

$$\begin{aligned} \sum D^2(\boldsymbol{v}) &= \sum (\boldsymbol{n}_i^T\boldsymbol{v} + d_i)^2 \\ &= \sum \Big(\boldsymbol{v}^T(\boldsymbol{n}_i\boldsymbol{n}_i^T)\boldsymbol{v} + 2d_i\boldsymbol{n}_i^T + d_i^2\Big) \\ &= \boldsymbol{v}^T\Big(\sum(\boldsymbol{n}_i\boldsymbol{n}_i^T)\Big)\boldsymbol{v} + 2\Big(\sum d_i\boldsymbol{n}_i^T\Big)\boldsymbol{v} + \sum d_i^2 \\ &= \boldsymbol{v}^T\mathrm{A}\boldsymbol{v} + 2\mathrm{b}^T\boldsymbol{v} + c \end{aligned} \tag{3}$$

where A is a symmetric 3×3 matrix, b is a column vector of size 3 and c is a scalar[2].

From [2], a quadric is defined as a triple as follows.

$$\mathrm{Q} = (\mathrm{A}, \mathrm{b}, c) \tag{4}$$

In a homogeneous matrix form, the quadric Q can be given by

$$\mathrm{Q} = \begin{pmatrix} \mathrm{A} & \mathrm{b} \\ \mathrm{b}^T & c \end{pmatrix} \tag{5}$$

Given this matrix, we can compute the geometric error $Q(\boldsymbol{v})$ using the quadric form below.

$$Q(\boldsymbol{v}) = \widetilde{\boldsymbol{v}}^T\mathrm{Q}\widetilde{\boldsymbol{v}} \qquad \text{where} \quad \widetilde{\boldsymbol{v}} = [\boldsymbol{v} \quad 1]^T = [x \quad y \quad z \quad 1]^T \tag{6}$$

The addition of quadrics can be defined component-wise: $\mathrm{Q}_i(\boldsymbol{v}) + \mathrm{Q}_j(\boldsymbol{v}) = (\mathrm{Q}_i + \mathrm{Q}_j)(\boldsymbol{v})$ where $(\mathrm{Q}_i + \mathrm{Q}_j) = (\mathrm{A}_i + \mathrm{A}_j, \mathrm{b}_i + \mathrm{b}_j, c_i + c_j)$.

Thus, the geometric error in (2) is computed by the sum of the quadrics Q_i which is determined by a set of planes.

$$E_{planes}(\boldsymbol{v}) = \sum D_i^2(\boldsymbol{v}) = \sum \mathrm{Q}_i(\boldsymbol{v}) = Q(\boldsymbol{v}) \quad \text{where } Q = \sum \mathrm{Q}_i \tag{7}$$

In equation (7), each quadric of a plane is given an equal weight. However, the area-weighted quadric is used in practice as follows:

$$Q = \sum area(f_i) \cdot \mathrm{Q}_i \quad \text{where } f_i \text{ is a neighboring face of the given vertex} \tag{8}$$

Since a quadric matrix is symmetric, 10 coefficients are required to store a quadric Q. Let us now consider the quadric error when contracting an edge $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ to $\boldsymbol{v}$. If the sum of quadrics is assigned to each vertex on the original mesh, the quadric of an edge is given by

$$\mathrm{Q}_{\boldsymbol{v}} = \mathrm{Q}_{\boldsymbol{v}_i}(\boldsymbol{v}) + \mathrm{Q}_{\boldsymbol{v}_j}(\boldsymbol{v}) \tag{9}$$

After the collapse of each edge, it is necessary to determine the position of a new vertex $\boldsymbol{v}$. The simplest way is to choose one of the end points as the target point. Another solution is to find the optimal position which minimizes the quadric error. The minimum of the quadric error occurs where the gradient of the quadric$\big(\nabla Q(\boldsymbol{v}) = 2\mathrm{A}\boldsymbol{v} + 2\mathrm{b}\big)$ equals to zero.

Solving for $\nabla Q(\boldsymbol{v}_{\min}) = 0$, the optimal position becomes $\boldsymbol{v}_{\min} = -\mathrm{A}^{-1}\mathrm{b}$.

3.2 Quadric for Geometry and Attributes

Garland and Heckbert generalized the quadric error metric which includes vertex attributes such as normal, vertex color and texture in [9]. The original quadric error metric lies in R^3. They placed their framework into a n-dimensional space where n equals to $3+m$, and m is the number of vertex properties. For example, m equals to 3 in the case of the colored surface, because color consists of RGB triple. This makes the quadric of a vertex formulated as the distance-to-hyperplane in R^n space. They assumed that all attributes are linearly interpolated over triangular faces and can be measured by the Euclidean distance between them. For scale-invariance, the model is scaled within the unit cube in R^n.

Let us consider the triangle $T = (v_1, v_2, v_3)$ as depicted in Fig. 2. Given a colored mesh, each vertex can be represented by $v_i = (x_i \quad y_i \quad z_i \quad r_i \quad g_i \quad b_i)^T$.

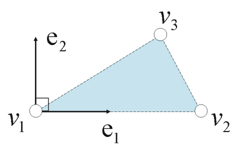


Fig. 2. Triangle in R^n and its corresponding orthonormal vectors[9]

To construct the quadric in the n-dimensional space, Garland and Heckbert computed two orthonormal unit vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. These vectors are given by Gram-Schmidt orthogonalization

$$\boldsymbol{e}_1 = \frac{\boldsymbol{v}_2 - \boldsymbol{v}_1}{\|\boldsymbol{v}_2 - \boldsymbol{v}_1\|} \tag{10}$$

$$\boldsymbol{e}_2 = \frac{\boldsymbol{v}_3 - \boldsymbol{v}_1 - (\boldsymbol{e}_1 \cdot (\boldsymbol{v}_3 - \boldsymbol{v}_1))\boldsymbol{e}_1}{\|\boldsymbol{v}_3 - \boldsymbol{v}_1 - (\boldsymbol{e}_1 \cdot (\boldsymbol{v}_3 - \boldsymbol{v}_1))\boldsymbol{e}_1\|} \tag{11}$$

Given these orthonormal unit vectors, we can formulate the quadric as $D^2(\boldsymbol{v}) = \boldsymbol{v}^T \mathrm{A}\boldsymbol{v} + 2\mathrm{b}^T\boldsymbol{v} + c$ where

$$\begin{aligned} \mathrm{A} &= \mathbf{I} - \boldsymbol{e}_1\boldsymbol{e}_1^T - \boldsymbol{e}_2\boldsymbol{e}_2^T \\ \mathrm{b} &= (\boldsymbol{v}_1 \cdot \boldsymbol{e}_1)\boldsymbol{e}_1 + (\boldsymbol{v}_1 \cdot \boldsymbol{e}_2)\boldsymbol{e}_2 - \boldsymbol{v}_1 \\ c &= \boldsymbol{v}_1 \cdot \boldsymbol{v}_1 - (\boldsymbol{v}_1 \cdot \boldsymbol{e}_i)^2 - (\boldsymbol{v}_2 \cdot \boldsymbol{e}_2)^2 \end{aligned} \tag{12}$$

In this generalized quadric, A is a symmetric $n \times n$ matrix, b is a column vector of size n and the total number of coefficients becomes $\Big((n+1)(n+2)\Big)/2$. For the vertex color, therefore, $28((7 \times 8)/2 = 28)$ coefficients are required to store a quadric.

4 Evaluation of Edge Cost

Our simplification method is similar to other algorithms and consists of two parts. First, we calculate the half-edge cost and place all half-edge collapses in a priority heap. Then we apply the simplification operator to the half-edge $(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1)$ having the lowest cost and update the costs of all half-edges involving $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$. In order to reduce memory overhead, we split the simplification error into two types of error term: the geometric error and the color error. We use the original QEM for our simplification of geometry because of its quality of approximation. On the other hand, our color error is classified into two terms: visual importance and collapse color error. We first define the visual importance, because we adopt half-edge collapse as our simplification operator. In the case of half-edge collapse, one of the vertices on an edge is removed while the other remains after collapse. Thus, it is desired to decide on the value that helps the final model to keep visually important vertices with respect to the global distribution of their color difference. On the other hand, the collapse color error represents the incremental color change as a result of each half-edge collapse. These two terms are described in section 4.1 and 4.2 and the details of the overall procedure are presented in section 4.3.

4.1 Visual Importance for Vertex Color

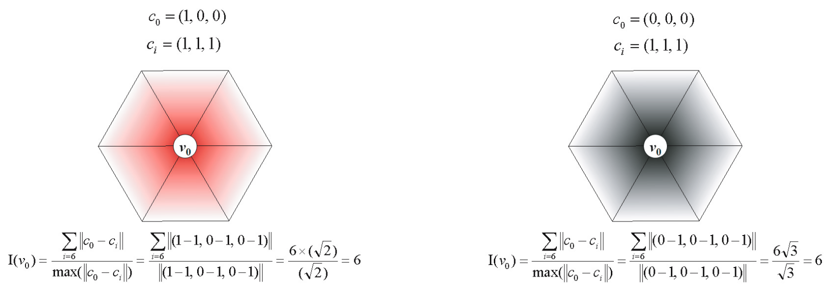
Visual importance $\mathrm{I}(\boldsymbol{v}_0)$ determines the color difference, that is, RGB distance between a vertex and its neighboring vertices in the original mesh. In other words, this value reflects the initial color difference. Since our color error does not store any history of color changes, we propose to use visual importance to represent the color change between the initial mesh and the result mesh. The colors of a triangular mesh are typically stored as (r, g, b) triples with each value in the range [0, 1]. The most straightforward way to measure the color difference is to treat the color space as a Euclidean space and compute RGB distance between two vertices as the equation below.

$$\|c_0 - c_i\| = \sqrt{(r_0 - r_i)^2 + (g_0 - g_i)^2 + (b_0 - b_i)^2} \tag{13}$$

where $c_0(r_0, g_0, b_0)$ is the color triple of vertex $\boldsymbol{v}_0$ and c_i is that of its neighboring vertex $\boldsymbol{v}_i$.

We define the visual importance of vertex $\boldsymbol{v}_0$ as follows:

$$\mathrm{I}(\boldsymbol{v}_0) = \begin{cases} \frac{\sum_{i=\text{neighbor vertices}} \|c_0 - c_i\|}{\max\|c_0 - c_i\|} & ,\text{when } \max(c_0 - c_i) \neq 0 \\ 0 & ,\text{when } \max(c_0 - c_i) = 0 \end{cases} \tag{14}$$


$c_0 = (1, 0, 0)$
$c_i = (1, 1, 1)$
v_0
$\mathrm{I}(v_0) = \frac{\sum_{i=6}\|c_0 - c_i\|}{\max(\|c_0 - c_i\|)} = \frac{\sum_{i=6}\|(1-1, 0-1, 0-1)\|}{\|(1-1, 0-1, 0-1)\|} = \frac{6\times(\sqrt{2})}{(\sqrt{2})} = 6$
$c_0 = (0, 0, 0)$
$c_i = (1, 1, 1)$
v_0
$\mathrm{I}(v_0) = \frac{\sum_{i=6}\|c_0 - c_i\|}{\max(\|c_0 - c_i\|)} = \frac{\sum_{i=6}\|(0-1, 0-1, 0-1)\|}{\|(0-1, 0-1, 0-1)\|} = \frac{6\sqrt{3}}{\sqrt{3}} = 6$

(a) (b)

Fig. 3. Examples of visual importance

The above definition means that the visual importance is zero if the color of vertex $\boldsymbol{v}_0$ is equal to its neighbor vertices, otherwise it is a real value determined by the maximum color difference and the color difference with the neighboring vertices. We divide the sum of color difference by the maximum color difference to measure the relative color difference. Fig. 3 shows the visual comparison of two vertices having different color distance. The color triples of vertex $\boldsymbol{v}_0$ in Fig. 3(a) and (b) are different, but it is obvious that vertex $\boldsymbol{v}_0$ has different color from neighbors in both cases. Consequently, the visual importance values in Fig. 3(a) and (b) are equal. We cannot represent the human perception of color just using the Euclidean distance in RGB space unlike the geometric error. For example, the distance between white and red is much shorter than that between white and black in the RGB space, but human does not perceive the color difference proportional to the distance. The visual importance metric, therefore, is added to emphasize the local color difference between vertices.

After the half-edge collapse ($\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1$), the visual importance of remaining vertex $\boldsymbol{v}_1$ is updated as

$$\mathrm{I}^{\text{after}}(\boldsymbol{v}_1) = \mathrm{I}^{\text{before}}(\boldsymbol{v}_0) + \mathrm{I}^{\text{before}}(\boldsymbol{v}_1) \tag{15}$$

In order to store and update visual importance, we need two floats per vertex, that is, the visual importance itself and the maximum color distance.

4.2 Collapse Color Error

The priority for simplification is stored in a heap with the lowest cost at the top. All half-edges have their own collapse error. The priority of a vertex $\boldsymbol{v}_0$ is determined by the minimum cost out of its out-going half-edges. Since the collapse color error should estimate the color change before and after the collapse, we define the collapse color error of half-edge collapse $\mathbf{C}(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1)$ as

$$\mathbf{C}(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1) = \sum_{i=\text{neighbor vertices of } \boldsymbol{v}_0} \Big| \|c_0 - c_i\| \cdot \|v_0 - v_i\| - \|c_1 - c_i\| \cdot \|c_1 - c_i\| \Big| \tag{16}$$

When using vertex color, we interpolate the color of a triangle by colors of three vertices. Hence, the length of an edge should be short in order to preserve colors of triangles in the semantic area where the color rapidly changes. It means that the color error of half-edge collapse is proportional to the length of the half-edge. Consequently, we multiply the length of the half-edge to the color difference for each half-edge in [15]. In the case of half-edge collapse, the color change of collapse always occurs on out-going half-edges of vertex $\boldsymbol{v}_0$ which is inside the gray area in Fig. 4. The connectivity in the white area is not changed by collapse.

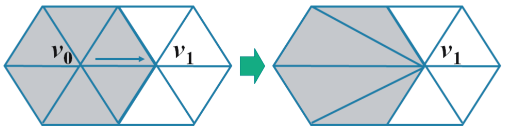


Fig. 4. The area of color change by a half-edge collapse

4.3 Simplification Algorithm

In the previous sections, we described two types of color error: visual importance and collapse color error. The final expression of our simplification algorithm combines the geometric error with these two color error terms. The total cost of half-edge collapse is now defined as

$$\begin{aligned} \text{error}(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1) &= (1 + \text{quadric}) \cdot (1 + \text{visual importance}) \cdot (1 + \text{color error}) \\ &= (1 + Q(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1)) \cdot (1 + \mathrm{I}(\boldsymbol{v}_0)) \cdot (1 + \mathbf{C}(\boldsymbol{v}_0 \rightarrow \boldsymbol{v}_1)) \end{aligned} \tag{17}$$

Since each of the above three error terms can be zero, we add one to each term to prevent the total error from becoming zero. Upon completion of computing the cost of every half-edge, the minimum cost among the out-going half-edge of each vertex is chosen as the vertex cost.

In summary, our simplification algorithm performs the following to simplify a 3D color mesh.

1. Compute the quadrics, visual importance and collapse color error.
2. Evaluate the cost of half-edge using (2)
3. Place all the half-edge collapses in a priority heap and collapse the half-edge which has the lowest cost.
4. Update the cost of all half-edge collapses involving vertices on the half-edge collapsed in step 3.
5. Repeat step 3 and 4 until there is no half-edge in the priority heap.

5 Results and Discussion

We compared the proposed method and the color QEM by Garland and Heckbert [11] using several 3D color models. They were compared in terms of computation time, memory consumption, geometric distance and color error. All models used were scanned by a commercial 3D scanner which can acquire both geometry and color. We ran these models on an Intel Pentium4-3.5GHz machine with 2G RAM. Fig. 5 shows the results of simplified models using the color QEM and our method, respectively. The reduction ratio of each model is described in Table 1. From the results, it is observed that our method can keep the geometry better and reduce the memory usage down to about a half of what is needed for the color QEM. However, the color QEM is faster and maintains color details better than our algorithm.

5.1 Computation Time

Table 1 lists the running time of the color QEM and our method in simplifying the three models at the reduction ratio presented in the table. Our method takes more time due to recalculating the color error that considers the change of connectivity after half-edge collapse, while the color QEM simply adds two quadrics after collapse. Although it depends on the size and connectivity of each model, our method is about 1.6 times slower than that of the color QEM.

5.2 Memory Consumption

The color QEM requires 28 floats per vertex. It means that the memory overhead is 112n bytes, where n is the number of vertices. The new quadric proposed by Hoppe[15] reduces the memory consumption to 23 floats per vertex. However, it is still heavy to construct a quadric for a huge size mesh. We use the original QEM which needs 10 floats for geometry. Additionally, we consume 2 floats for color error; one for the visual importance and the other to store the maximum color difference. Consequently, our method only spends 48n bytes of memory in total, which gives a significant advantage compared to the color QEM.

Table 1. Reduction ratio and computation times with respect to color QEM and our method

Model	Model Size (# of vertices/ # of faces)	Reduction Ratio(%) (# of vertices/# of faces)		Computation time(sec)	
		Color QEM	Our Method	Color QEM	Our Method
Monkey	50,503/ 101,002	90% (5,050/ 10,096)	90% (5,050/ 10,096)	3.13	5.46
Wood Duck	50,052/ 99,997	90% (5,005/ 9,925)	90% (5,005/ 9,926)	3.13	5.34
GIST Ari	1,175,653/ 2,361,302	97% (35,269/ 70,534)	97% (35,269/ 70,534)	83.78	143.20

5.3 Geometric Comparison

For geometric comparison between the original model and the simplified model, we used the well-known surface comparison tool - Metro[16] which is used by many researchers. In this research, we measured the mean distance and the RMS distance between the simplified mesh and the original mesh. Table 2 shows the results of the geometric comparison. The Mean and RMS distances of our method are always smaller than those of the color QEM for all tested models.

Table 2. Mean and RMS geometric error by Metro[14]

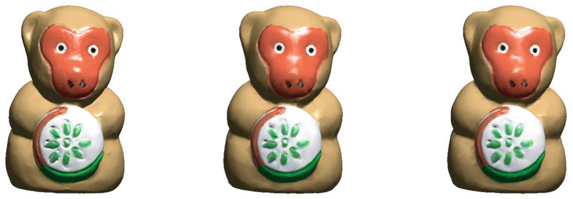
Model	Color QEM		Our Method	
	Mean error	RMS error	Mean error	RMS error
Monkey	3.95	5.30	3.44	4.69
Wood Duck	7.60	11.32	5.92	10.14
GIST Ari	1.45	2.35	0.82	1.41

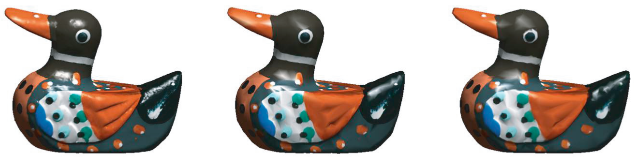
5.4 Similarity of Color Appearance

For a given view, the appearance of a model is determined by the corresponding raster image which a renderer would produce[1]. For this, we implemented our own triangular mesh viewer which renders a mesh by OpenGL. To compare the

Table 3. Color error with comparison of rendered images

Model	Color QEM	Our Method
	RMS error	RMS error
Monkey	3.60	4.44
Wood Duck	9.07	10.41
GIST Ari	7.73	9.62


(a) GIST Monkey(10% of the original model)


(b) Wood Duck(10% of the original model)

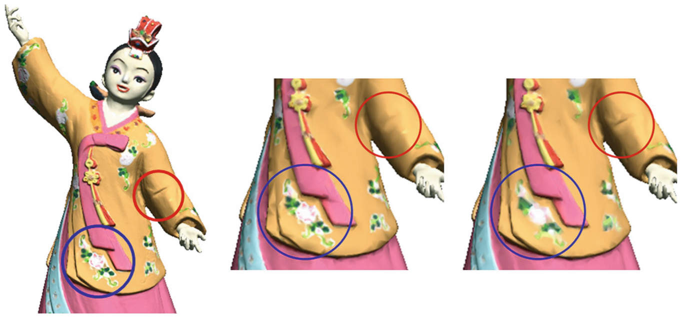

(c) GIST Ari(3% of the original model)

Fig. 5. Simplified results of color QEM and our method. The leftmost column is the original model. The middle column is the color QEM result and the right column is ours.

original color of vertex, we disabled the lighting condition in OpenGL. After rendering a model, we obtained two 24-bit bitmap images of the original model and its simplified model separately. We compare color differences between two images pixel by pixel and calculate the RMS error of Euclidean distances between corresponding pixels of two images. Table 3 shows the RMS error between the rendered images of the original and the simplified models. As shown from Table 3, the color QEM can maintain the color distribution better than our algorithm.

6 Conclusion and Future Work

In this paper, we have presented a novel and memory-efficient half-edge cost evaluation method to simplify 3D mesh with vertex color. The main benefit of our method is to reduce memory consumption less than half of the original color QEM. Additionally, our method can keep the geometric shape better and the running time is also comparable to that of the color QEM.

However, our proposed method shows a weakness in maintaining accurate color details in an area with fine shapes. We will try to solve this problem by integrating hierarchical multiresolution framework. Our future work will focus on solving this problem and making our algorithm faster. In the proposed algorithm, it is also ignored that the RGB space is not perceptually linear. In our future work, a uniform color space[24]]such as CIE-Lab or XYZ space will be used in replacement of RGB space. The discontinuity problem of the attribute value will also be addressed for preserving the color attribute.

Acknowledgments. This research is supported by the NRL (National Research Laboratory) funded by the Ministry of Science and Technology of Korea and the CTI development project supported by the MCT and KOCCA in S. Korea.

References

1. Garland, M.: Multiresolution Modeling: Survey & Future Opportunities. In: Eurographics 1999, State of the Art Report (1999)
2. Garland, M., Heckbert, P.: Surface Simplification Using Quadric Error Metric. In: Computer Graphics(SIGGRAPH 1997 Proceeding), pp. 209–216 (1997)
3. Schröeder, W.J., Zarge, A., Loresen, W.E.: Decimation of Triangle Mesh. Computer Graphics(SIGGRAPH 1992 Proc.) 26(2), 65–70 (1992)
4. Hussain, M., Okada, Y., Niijima, K.: Efficient and Feature-Preserving Triangular Mesh. Journal of WSCG 12(1-3), 167–174 (2004)
5. Brodsky, D., Watson, B.: Model Simplification through Refinement. In: Graphics Interface 2000, pp. 221–228 (2000)
6. Ciampalini, A., Cignoni, P., Montani, C., Scopigno, R.: Multiresolution Decimation based on Global Error. The Visual Computer 13, 223–246 (1997)
7. Hoppe, H.: Progressive Mesh. In: Proc. SIGGRAPH 1996, pp. 99–108 (1996)
8. Certain, A., Popović, J., DeRose, T., Duchamp, T., Salesin, D., Stuetzle, W.: Interactive Multiresolution Surface Viewing. In: Proceedings of ACM SIGGRAPH 1996, pp. 91–98 (1996)
9. Garland, M., Heckbert, P.: Simplifying Surfaces with Color and Texture Using Quadric Error Metrics. In: Proc. Of IEEE Visualization 1998, pp. 263–270 (1998)
10. Wünsche, B.: A Survey and Evaluation of Mesh Reduction Techniques. In: Proc. IVCNZ 1998, pp. 393–398 (1998)
11. Kobbelt, L., Campagna, S., Seidel, H.P.: A General Framework for Mesh Decimation. In: Graphics Interface 1998 proceedings, pp. 43–50 (1998)
12. Garland, M., Heckbert, P.: Surface Simplification Using Quadric Error Metric. In: Visualization 98 proceedings, IEEE, pp. 263–269 (1998)
13. Hoppe, H.: New Quadric Metric for Simplifying Meshes with Appearance Attributes. In: IEEE Visualization 1999, pp. 59–66 (1999)
14. Cignoni, P., Rocchini, C., Scopigno, R.: Measuring Error on Simplified Surfaces. Computer Graphics Forum 17(2), 167–174 (1998)
15. Kalvin, A.D., Taylor, R.H.: Superfaces: Polygonal Mesh Simplification with Bounded Error. IEEE Computer Graphics and Applications 16(3), 64–77 (1996)
16. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical Face Clustering on Polygonal Surfaces. In: Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, pp. 49–58 (2001)

17. Klein, R., Krämer, J.: Multiresolution Representations for Surface Meshes. In: Proceedings of Spring Conference on Computer Graphics 1997, pp. 57–66 (1997)
18. Hamann, B.: A Data Reduction Scheme for Triangulated Surfaces. Computer Aided Geometric Design 11, 197–214 (1994)
19. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh Optimization. In: Proceddings of SIGGRAPH 1993, pp. 19–26 (1993)
20. Kim, S.J., Kim, S.K., Kim, C.H.: Discrete Differential Error Metric for Surface Simplification. In: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, pp. 276–283 (2002)
21. De Floriani, L., Magillo, P., Puppo, E., Sobrero, D.: A multi-resolution topological representation for non-manifold meshes. Computer Aided Design 36(2), 141–159 (2004)
22. De Floriani, L., Hui, A.: A Dimension-Independent Representation for Multiresolution Nonmanifold Meshes. Journal of Computing and Information Science in Engineering 6(4), 397–404 (2006)
23. Cignoni, P., Montani, C., Rocchini, C., Scopigno, R.: Interactive Multiresolution Surface Viewing. IEEE Transactions on Visualization and Computer Graphics 9(4), 525–537 (2003)
24. Rigiroli, P., Campadelli, P., Pedotti, A., Borghese, N.A.: Mesh refinement with color attributes. Computers & Graphics 25, 449–461 (2001)
25. Fahn, C.-.S., Chen, H.-.K., Shiau, Y.-.H.: Polygonal Mesh Simplification with Face Color and Boundary Edge Preservation Using Quadric Error Metric. In: Proceedings of IEEE Fourth International Symposium on Multimedia Software Engineering, pp. 174–181 (2002)