# FEATURE-PRESERVING MESH SIMPLIFICATION: A VERTEX COVER APPROACH

Sajid Hussain, Hakan Grahn and Jan Persson
*Department of Systems and Software Engineering*
*Blekinge Institute of Technology, Sweden*
*{sajid.hussain, hakan.grahn,jan.persson}@bth.se*

**ABSTRACT**

In computer graphics image synthesis algorithms like ray tracing, the mesh complexity decreases the performance of these algorithms. Therefore, the need arises to reduce the complexity of these meshes and at the same time preserving the salient features of the shape. Initial selection of vertices for mesh simplification heavily relates with the quality of the simplified meshes. In this paper, we present a greedy approach to select initial vertex contraction pairs to preserve salient features in the simplified meshes. The greedy algorithm exploits the property of meshes where vertices forming small features contain less number of edges. The technique selects vertices connected with large number of edges and makes them potential candidates for contraction according to a given cost function. The purpose is to first simplify those regions which are enriched with number of triangles and preserve small details of the shape constructed with small number of triangles. Our technique preserves very small details in the shape even after considerable simplification as compared to other existing techniques. Initial experiments show promising results with preserved salient features.

**KEYWORDS**

Mesh simplification, Feature-preserving, Pair contraction, Level of details, Multiresolution modeling.

## 1. INTRODUCTION

Complex and detailed meshes are directly related to the quality and realism of synthesized images in computer graphics [1]. However, the complexity of meshes requires substantial storage and also reduces the performance of rendering algorithms like ray tracing. Although, 3D data acquisition scanners can easily generate millions of polygons for simple objects, it is always required to have simple version of these complex meshes generated automatically and at the same time preserving salient features. The most common use of the mesh simplification algorithms is in representing distant objects in a scene with low level of detail (LOD) and near objects with higher LOD. Since, the performance of the mesh simplification algorithms is greatly affected by the vertices selected for contraction or decimation [1], extra care should be taken while performing this operation. It is not always feasible to select vertices with the least contraction or decimation cost first, if we need to preserve small features in the shape. We have developed a mesh simplification algorithm which uses a vertex contraction method combined with a vertex selection method as an initial start. The vertex selection method incorporates in itself a greedy algorithm for the selection of vertices. We stop our greedy algorithm after we have found vertices equal to or greater than some threshold. The algorithm focuses on selection of vertices with maximum number of edges and determines the contraction pair which minimizes a cost function. Our algorithm first identifies a set of vertices with connected edges equal to or greater than some threshold value. It then finds a minimum cost contraction pair for each vertex in the main set (we use the heuristic that vertices do not move far from their original positions after contraction. This also serves the purpose of selecting those vertices first which are connected with triangles with small areas). Vertices in the main set are then sorted according to their contraction cost with minimum contraction cost first and iteratively removed and contracted into new vertices with new positions to simplify the mesh. The new positions are determined with the one dimensional search algorithm described in [15]. The algorithm makes use of a golden section search technique combined with parabolic interpolation to find the minimum of the contraction cost function along the edge joining the contraction vertices pair. The primary advantage of our algorithm is the quality of the produced mesh both geometrically and visually. The preservation of the

salient features of the model even after considerable simplification is another property of our technique. The algorithm preserves very small details in the shapes even after 50-55% complexity reduction as compared to the other two algorithms discussed in this paper. Hence, a suitable choice for meshes with small level of details (LODs). The remainder of this paper is organized as follows. Section 2 reviews related work in this particular area and discusses some mesh simplification techniques. In section 3, we introduce the theory behind our algorithm. Section 4, gives some implementation details of our algorithm with some comparison results. We conclude the paper in section 5 with future work.

## 2. PREVIOUS WORK

We focus here on triangulated meshes due to their generality and common use. There exist different techniques to simplify triangulated meshes. Some common techniques are Edge Collapse (Vertex Contraction), Vertex Decimation, Vertex Clustering and Face Constriction Process. In Edge Collapse Process, an edge is identified and collapsed to form a new vertex. The optimal placement of the new formed vertex combined with the edge selection determines the quality of the process. Many researchers [1] [2] [3] [4] have used the technique in their research work. Figure 1(a) presents the idea visually. In Vertex Decimation Process, a vertex is deleted along with its connecting edges and the resulting hole is re-triangulated. The algorithm is described in [5], where the method iteratively selects the vertices and performs decimation process. The technique is also used in [6]. Figure 1(b) presents the idea visually. In Vertex Clustering Process described by [7], a bounding box is placed around the model and divided into an equally spaced grid. The vertices present in each cell are contracted together and the corresponding faces are updated accordingly. The process is quite fast and the quality of the output mesh depends on the size of the grid. Figure 1(c) presents the idea visually. In Face Constriction Process, a triangular face is constricted and its adjacent faces become degenerated and therefore deleted. The technique is adopted by [8] and [9]. Figure 1(d) presents the idea visually. A more general survey on mesh simplification algorithms can be found in [10]. Recent works in mesh simplification and implementation can be found in [12], [13] and [14].

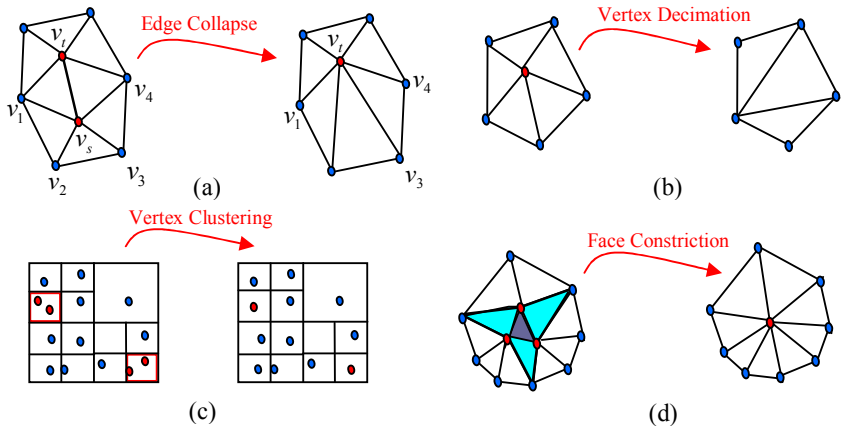


Figure 1. Mesh Simplification Methods (a) Edge Collapse (b) Vertex Decimation (c) Vertex Clustering and (d) Face Constriction

## 3. VERTEX COVER PAIR CONTRACTION

In our algorithm, we use the iterative pair contraction which is used in many simplification algorithms. First, we select the initial vertex set and we start with a greedy algorithm for minimum vertex cover selection problem. Although, we do not aim to approximate the optimal solution in minimum vertex cover problem, the idea is to select the vertices with a large number of edges. This step is important in preserving small features in the shape as the performance of the algorithm depends on the initial pair selection criteria.

Second, we find valid contraction pairs for each vertex in the main set according to some cost function (Valid contraction pairs are selected based on the assumption that points do not move far from their original locations in a good approximation [1]) and sort the main set vertices with respect to their contraction cost values. We then choose a sub-set of the vertices from the main set depending on the number of polygons to reduce. We then perform the contraction operation and update our mesh. The greedy algorithm for minimum vertex cover problem works as follows. The algorithm takes initial mesh $M_0 = G(V, E)$ and returns a set of vertex cover $V'$.

```
function V' = MinVertexCover ( M_0 )
V' ← ∅; E' ← E
      while  E' ≠ ∅ do
            select an arbitrary edge uv in E'
            add u and v to V'
            remove edges incident to u or v from E'
      endwhile
return  V'
end function
```

We have used position and curvature uncertainty functions along with quadric error metrics to select edges for contraction. These functions are described in detail in [11] and [1]. The position uncertainty function is defined as

$$\mu_{pos}(\mathbf{v}_s, \mathbf{v}_t) = \max_{\mathbf{v}_j \in V(\mathbf{v}_s)} \left\{ \left\| \|\mathbf{v}_s - \mathbf{v}_j\| - \|\mathbf{v}_t - \mathbf{v}_j\| \right\| \right\}, \tag{1}$$

and the curvature uncertainty function is defined as follows.

$$\mu_{cur}(\mathbf{v}_s, \mathbf{v}_t) = \left| \min_{f_i, f_j \in T(\mathbf{v}_s)} \left( f_{i\perp} \cdot f_{j\perp} \right) - \min_{f_i, f_j \in T(\mathbf{v}_t)} \left( f_{i\perp} \cdot f_{j\perp} \right) \right|. \tag{2}$$

Where $\mathbf{v}_s$ and $\mathbf{v}_t$ are the two vertices to be analyzed for collapse operation. The set $V(\mathbf{v}_s)$ contains vertices adjacent to $\mathbf{v}_s$ before contraction. For example in Figure 1(a)

$$V(\mathbf{v}_s) = \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_t. \tag{3}$$

The set $T(\mathbf{v}_s)$ is the set of triangles adjacent to $\mathbf{v}_s$ before contraction and $f_\perp$ is the normal to triangular face. From Figure 1(a),

$$T(\mathbf{v}_s) = \Delta\mathbf{v}_s\mathbf{v}_1\mathbf{v}_2, \Delta\mathbf{v}_s\mathbf{v}_2\mathbf{v}_3, \Delta\mathbf{v}_s\mathbf{v}_3\mathbf{v}_4, \Delta\mathbf{v}_s\mathbf{v}_4\mathbf{v}_t, \Delta\mathbf{v}_s\mathbf{v}_t\mathbf{v}_1. \tag{4}$$

$T(\mathbf{v}_t)$ contains the set of all triangles after contraction and from Figure 1(a),

$$T(\mathbf{v}_t) = \Delta\mathbf{v}_t\mathbf{v}_1\mathbf{v}_2, \Delta\mathbf{v}_t\mathbf{v}_2\mathbf{v}_3, \Delta\mathbf{v}_t\mathbf{v}_3\mathbf{v}_4. \tag{5}$$

In quadric error metrics approach, each vertex is associated with a symmetric 4x4 matrix $\mathbf{Q}$ and the errors at vertices $\mathbf{v}_s$ and $\mathbf{v}_t$ are defined as $\mathbf{v}_s^T\mathbf{Q}_s\mathbf{v}_s$ and $\mathbf{v}_t^T\mathbf{Q}_t\mathbf{v}_t$ respectively. The cost for contracting $\mathbf{v}_s \to \mathbf{v}_t$ becomes $\mathbf{v}_t^T\mathbf{Q}_s\mathbf{v}_t$ and contracting $\mathbf{v}_t \to \mathbf{v}_s$ becomes $\mathbf{v}_s^T\mathbf{Q}_t\mathbf{v}_s$. So, the total cost function for contracting a vertex pair $\mathbf{v}_s\mathbf{v}_t$ becomes

$$Cost(\mathbf{v}_s, \mathbf{v}_t) = \mu_{pos}(\mathbf{v}_s, \mathbf{v}_t) + \mu_{cur}(\mathbf{v}_s, \mathbf{v}_t) + \mathbf{v}_t^T\mathbf{Q}_s\mathbf{v}_t, \tag{6}$$

or

$$Cost(\mathbf{v}_t, \mathbf{v}_s) = \mu_{pos}(\mathbf{v}_t, \mathbf{v}_s) + \mu_{cur}(\mathbf{v}_t, \mathbf{v}_s) + \mathbf{v}_s^T\mathbf{Q}_t\mathbf{v}_s, \tag{7}$$

or

$$\overline{Cost}(\mathbf{v}_s, \mathbf{v}_t) = \mu_{pos}(\mathbf{v}_s, \mathbf{v}_t) + \mu_{cur}(\mathbf{v}_s, \mathbf{v}_t) + \bar{\mathbf{v}}^T(\mathbf{Q}_s + \mathbf{Q}_t)\bar{\mathbf{v}}. \tag{8}$$

Equations 6, 7 and 8 correspond to the contraction costs for $\mathbf{v}_s \to \mathbf{v}_t$, $\mathbf{v}_t \to \mathbf{v}_s$ and $\mathbf{v}_s, \mathbf{v}_t \to \bar{\mathbf{v}}$ respectively. Here $\bar{\mathbf{v}}$ is an optimal vertex position along the segment $\mathbf{v}_s - \mathbf{v}_t$ where the contraction cost is minimum. Hence, the final cost of contracting a vertex pair $\mathbf{v}_s, \mathbf{v}_t$ becomes

$$\overline{Cost}^{*}(\mathbf{v}_s, \mathbf{v}_t) = \min\left\{ Cost(\mathbf{v}_s, \mathbf{v}_t), Cost(\mathbf{v}_t, \mathbf{v}_s), \overline{Cost}(\mathbf{v}_s, \mathbf{v}_t) \right\}. \quad (9)$$

All the cost function values are normalized between [0 1] in the real implementation. The optimal contraction vertex position $\bar{\mathbf{v}}$ search along the segment $\mathbf{v}_s - \mathbf{v}_t$ is done with a one dimensional optimization technique, i.e., a golden section search combined with parabolic interpolation to speed up the convergence rate [15]. The technique takes advantage of the fact that a second order polynomial usually provides a good approximation to the shape of a parabolic function. As the cost function we are dealing with is parabolic in nature, parabolic interpolation can provide a good approximation of the cost function minima and hence the optimum vertex position along the contraction segment. Figure 2 shows the idea visually, where the contraction cost functions for $\mathbf{v}_s \to \mathbf{v}_t$ and $\mathbf{v}_t \to \mathbf{v}_s$ are plotted along with their summation and $\bar{\mathbf{v}}$ is the optimum contraction position along the segment $\mathbf{v}_s - \mathbf{v}_t$.

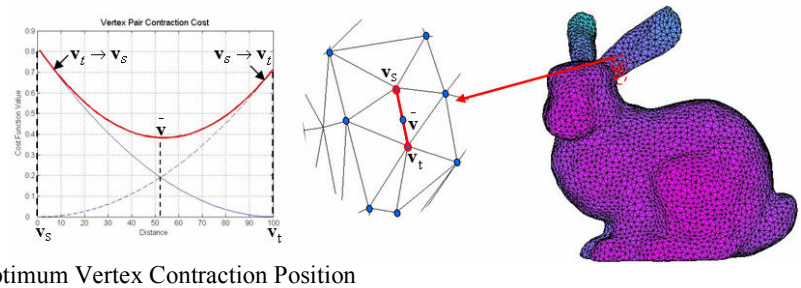


Figure 2. The Optimum Vertex Contraction Position

Our technique uses the greedy algorithm for vertex cover problem for finding suitable vertex contraction pairs. The main purpose is to preserve the salient features in the shapes, both geometrically and visually. The main steps involved in our algorithm can be summarized as follows:

(a) Start with the greedy algorithm for vertex cover $V'$ of the input mesh $M_0$.

(b) Generate the initial set $P$ and include vertices with number of edges equal to or greater than some threshold $t$.

(c) For each vertex in $P$ identify its contraction pair with minimum contraction cost according to the cost functions in Equations 6 and 7.

(d) Sort the vertices in $P$ (ascending order) with respect to their contraction costs.

(e) Pick up the sub-set $p$ from $P$ according to the number of polygons to be reduced from the mesh (as $P$ is sorted in the above step, we will always pick the vertices with small contraction cost).

(f) Update the mesh by contracting each vertex in $p$ with its counter part and return the simplified mesh $M_1$.

## 4. EXPERIMENTAL RESULTS

We have implemented our algorithm in MATLAB® and chosen some meshes with small features, especially to test our algorithm. We have also compared the quality of our algorithm with some existing mesh simplification techniques. Figure 3 shows the results and compares them with other two algorithms, Quadric Error Metrics [1] and its flavour called Quadric Weighted by Triangle Area. Notice the small level of details in Helicopter (missile fins) and Ship (Antennas and Parabolic dish) meshes (red dotted circles). After about

50% reduction, our algorithm still preserves them as compared to the other two algorithms. Table1 shows the running time of our algorithm in MATLAB® along with percentage reduction in scene complexity. Timings for Bunny scenes with variable complexities are also shown in Table1. For the evaluation purpose of our algorithm, we have used attribute deviation metric (ADM) [16]. The ADM is guided by geometrical correspondence between two meshes. It uses the difference assessment between the mesh attributes and is calculated by defining a set of points on the mesh surfaces. Figure 3 shows the results and in case of Helicopter and Military Ship, our algorithm exhibits less mean geometric deviation as compared to the two other algorithms because of small features preservation. This is not the case with the Bunny and the mean geometric deviation for our algorithm is almost the same as compared to the two other algorithms. The reason behind is that the Bunny does not contain very small features as compared to the other two meshes.

Table 1. Running time and percentage reduction: our algorithm

| Scene | Primitives | Reduced To | % age | Time |
|---|---|---|---|---|
| Helicopter | 6448 | 3534 | 55% | 25 sec |
| Mil. Ship | 10781 | 6128 | 56% | 57 sec |
| Bunny | 3851 | 2018 | 52% | 11 sec |
| Bunny | 9580 | 5148 | 53% | 48 sec |
| Bunny | 16301 | 8788 | 54% | 2 min |
| Bunny | 69451 | 39918 | 57% | 15 min |

## 5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a feature preserving mesh simplification algorithm. We have used the vertex cover problem for initially selecting vertex pairs for contraction. Results show that our algorithm successfully preserves small details in the meshes as the initial selection of vertices is very important in preserving small features in the shapes. We do use the strategy to contract minimum cost vertices first, as used by other mesh simplification techniques, but the difference is that we choose the minimum cost contraction pairs from vertices with maximum edges first. We implemented the algorithm in MATLAB® and showed the timing details for scenes with different complexities. As MATLAB® is quite slow executing "for loops" as compared to C++, the algorithm performs quite slowly. Towards implementation side, the future work is to implement the algorithm in C++ and reduce the execution time.

## REFERENCES


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

[2].H. Hoppe.: Progressive Meshes. Computer Graphics (SIGGRAPH'96 Proceedings), 99–108, (1996)

[3].H. Hoppe., T Duchamp.: Mesh Optimization. Computer Graphics (SIGGRAPH'93), 19–26, (1993)

[4].P. Lindstrom., G. Turk.: Fast and Memory Efficient Polygonal Simplification. Proceedings of IEEE Visualization'98, 279–286, (1998)

[5].W. J. Schroeder., J. A. Zarge., W. E. Lorensen.: Decimation of Triangle Meshes. Computer Graphics (SIGGRAPH'92 Proceedings), 65–70, (1992)

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

[7].J. Rossignac., P. Borrel.: Multi-resolution 3D Approximation for Rendering Complex Scenes. Modeling in Computer Graphics: Methods and Applications, 279–286, (1993)

[8].T. S. Gieng., B. Hamann., K. I. Joy.: Smooth Hierarchical Surface Triangulation. Proceedings of IEEE Visualization'97, 379-386, (1997)

[9].B. Hamann.: A Data Reduction Scheme for Triangulated Surfaces. Computer Aided Geometric Design, 197-214, (1994)

[10].D. P. Luebke.: A Developer's Survey of Polygonal Simplification Algorithms. IEEE Computer Graphics and Applications'01, 24-35, (2001)
[11].C. Chang., S. K, Yang., D. Z. Duan., M. F. Lin.: A Fuzzy Based Approach to Mesh Simplification. Journal of Information Science and Engineering 18, 459-466, (2002)
[12].Y. Wu, Y. He, H. Cai.: QEM-based Mesh Simplification with Global Geometry Features Preserved. Computer Graphics (SIGGRAPH'04 Proceedings), 50–57, (2004)
[13].S. Mata, L. Pastor, A. Rodriguez.: Attention Based Mesh Simplification using Distance Transforms. Lecture Notes in Computer Science (LNCS'06), Vol (4245), 83-294, (2006)
[14].C. DeCoro, N. Tatarchuk.: Real-time Mesh Simplification using GPU. Computer Graphics (SIGGRAPH'07 Proceedings), 161–166, (2007)
[15]. S. Hussain and H. Grahn.: Fast kd-tree construction for 3d-rendering algorithms like ray tracing. In Proc. Third International Symposium on Advances in Visual Computing, Lecture Notes in Computer Science (LNCS'07) (2007), vol. 4842, pp. 681–690.
[16]. Roy, M., Foufou S. and Truchetet F.: Mesh comparison using attribute deviation metric. International Journal of Image and Graphics 4, 1 (2004), 1–14.

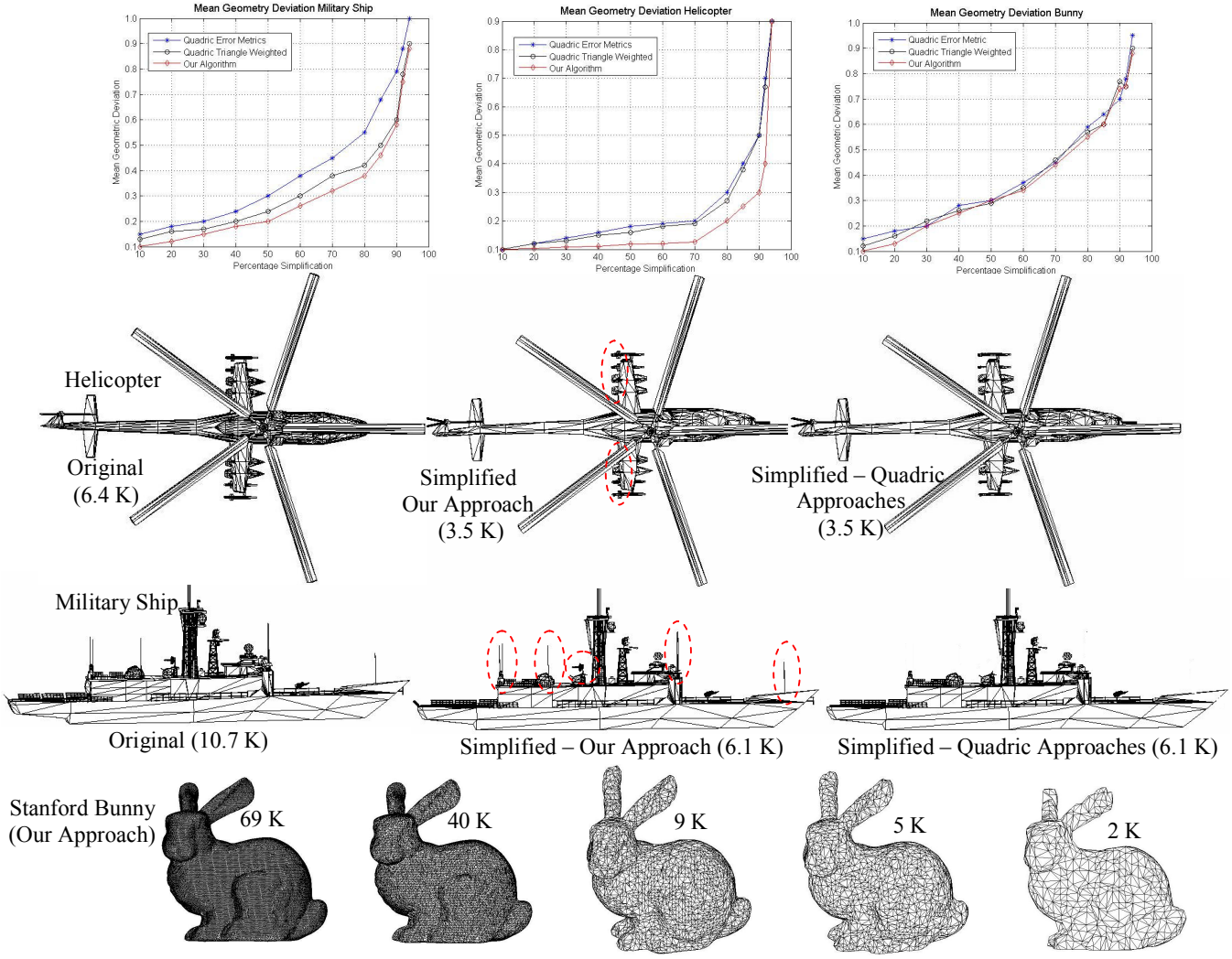


Figure 3. Examples of Geometric Deviation and Mesh Optimization