Sajid Hussain · Håkan Grahn · Jan Persson

Feature-Preserving Mesh Simplification: A Greedy Vertex Cover

Abstract In computer graphics rendering algorithms like ray tracing, the complexity of meshes decreases the performance of these algorithms. Therefore, the need arises for reduced mesh complexity with considerably good quality and at the same time preserving the salient features of the shape. Mesh simplification problem in some cases can be considered as an initial value problem where the quality of the simplified mesh heavily depends on the initial selection of vertices for contraction. In this short paper, we present an ongoing work in mesh simplification that uses a greedy algorithm for vertex cover problem in order to select vertex contraction pairs to preserve salient features in the simplified mesh. 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]. On the other hand, the complexity of meshes hinders the performance of rendering algorithms like ray tracing. It is then useful to have a simple version of these complex meshes generated automatically and at the same time preserving salient features. We have developed a mesh simplification algorithm which uses a greedy algorithm for vertex cover problem as a starting point for selecting contraction pairs. Although, we use the term, vertex cover, but we do not aim to solve or approximate it completely in minimum vertex cover sense. 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.

Sajid Hussain
Blekinge Institute of Technology, Sweden
Tel.: +46 457 38 58 76
E-mail: sajid.hussain@bth.se

Håkan Grahn
Blekinge Institute of Technology, Sweden
Tel.: +46 457 38 58 04
E-mail: hakan.grahn@bth.se

Jan Persson
Blekinge Institute of Technology, Sweden
Tel.: +46 457 38 58 29
E-mail: jan.persson@bth.se

Our algorithm first identifies a set of vertices with connected edges equal to or greater than some threshold value. It then selects a sub-set of vertices (with minimum contraction cost first) from the main set.

Vertices in the sub-set are then iteratively contracted and the mesh is simplified. The size of the sub-set depends on the number of polygons we want to reduce from the mesh. 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 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 Related 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 discussed below.

2.1 Edge Collapse (Vertex Contraction) Process

In this technique, 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. Fig.1(a) presents the idea visually.

2.2 Vertex Decimation Process

In this 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]. Fig.1(b) presents the idea visually.

2.3 Vertex Clustering Process

In the 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. Fig.1(c) presents the idea visually.

2.4 Face Constriction Process

In this process, a triangular face is constricted and its adjacent faces become degenerated and therefore deleted. The technique is adopted by [8] and [9]. Fig.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].

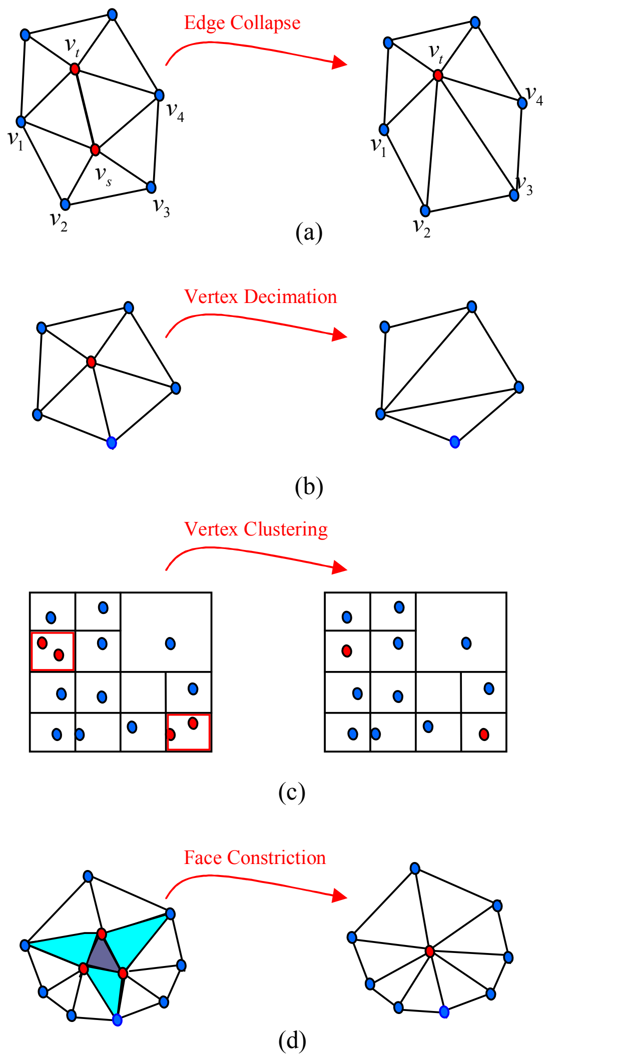

Fig. 1 Mesh Simplification Techniques

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 vertex cover selection problem. Although, we do not aim to find the optimal solution in 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.

3.1 A Greedy Algorithm for Vertex Cover

The vertex cover problem is to find the set of vertices in a graph $G(V,E)$ that covers all edges. More strictly speaking, a subset $V' \subseteq V$ is a vertex cover if every edge has at least one endpoint in V' and $V - V'$ is an independent set. Finding a maximum of the independent set is therefore equivalent to finding minimum a vertex cover.

```
function $V'$ = MinVertexCover ( $M_0$ )
$V' \leftarrow \varnothing; E' \leftarrow E$
while $E' \neq \varnothing$ 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
function $M_1$ = MeshSimplification ( $M_0$ )
// Start with Vertex Cover
$V' \leftarrow$ MinVertexCover ( $M_0$ )
// Generate Initial Set $P$
For all $V_i' \in V'$
$P \leftarrow V_i'$ if $E_i' \geq t$ // Where $t$ is some threshold and $E_i'$ are edges for $V_i'$
// Determine Minimum Cost Edge for Each Vertex in $P$
For all $V_i' \in P$
$V'_{i.cost} \leftarrow$ Contraction cost for $V_i'$ and $V_j'$ // Where $V_j'$ is surrounding vertex of $V_i'$
// Sort $P$ According to its Vertices Contraction Cost
$P = sort(P \rightarrow V'_{i.cost})$
// Pick Sub-Set $p$ with size According to Number of Polygons to Reduce where $p \leq P$
$p \leftarrow P(1 : PolygonsReduce / 2)$
// Update the mesh $M_0$ with simplified Mesh $M_1$
$M_1 \leftarrow$ UpdateMesh ( $M_0, p$ )
return $M_1$
end function
```

Fig. 2 Vertex Pair Contraction Algorithm

The greedy algorithm for finding the vertex cover works as shown in Fig.2 ($V^{'}$= MinVertexCover (M_0)). The algorithm takes initial mesh $M_0 = G(V, E)$ and returns a set of vertex cover $V^{'}$.

3.2 Cost Function to Minimize

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

$$\mu_{pos}(v_s, v_t) = \max_{v_j \in V(v_s)} \left\{ \left| \left\| v_{s.pos} - v_{j.pos} \right\| - \left\| v_{t.pos} - v_{j.pos} \right\| \right| \right\}, \qquad (1)$$

and the curvature uncertainty function is defined as follows.

$$\mu_{cur}(v_s, v_t) = \left| \min_{f_i, f_j \in T(v_s)} \left(f_{i\perp} . f_{j\perp} \right) - \min_{f_i, f_j \in T(v_t)} \left(f_{i\perp} . f_{j\perp} \right) \right|. \qquad (2)$$

Where v_s and v_t are the two vertices to be analyzed for collapse operation. The set $V(v_s)$ contains vertices adjacent to v_s before contraction and from Fig.1(a)

$$V(v_s) = v_1, v_2, v_3, v_4, v_t . \qquad (3)$$

The set $T(v_s)$ is the set of triangles adjacent to v_s before contraction and $f_\perp$ is the normal to triangular face. From Fig.1(a),

$$T(v_s) = \Delta v_s v_1 v_2, \Delta v_s v_2 v_3, \Delta v_s v_3 v_4, \Delta v_s v_4 v_t, \Delta v_s v_t v_1 . \qquad (4)$$

$T(v_t)$ contains the set of all triangles after contraction and from Fig.1(a),

$$T(v_s) = \Delta v_t v_1 v_2, \Delta v_t v_2 v_3, \Delta v_t v_3 v_4 . \qquad (5)$$

So, the total cost function for contracting a vertex pair v_s, v_t becomes

$$Cost(v_s, v_t) = \mu_{pos}(v_s, v_t) + \mu_{cur}(v_s, v_t) . \qquad (6)$$

3.3 Algorithm Summary

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 pseudo code of the algorithm is shown in Fig.2, where the main function of the algorithm `MeshSimplification` takes M_0 as an input mesh and returns M_1 as simplified mesh. 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 function in section 3.2.

(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 and Comparisons

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. Fig.3 shows the results and compares them with other two algorithms, Quadric Error Metrics [1] and its flavor 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 %age reduction in scene complexity. Timings for Bunny scenes with variable complexities are also shown in Table1.

Table 1 Running time and %age 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 short 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.

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.

Towards algorithmic side, we plan to combine the above mentioned implementation with Genetic Algorithms (GA) and Neuro-Fuzzy approaches for mesh simplification. We would like to generate random sets of vertices and apply GA to minimize the cost function.

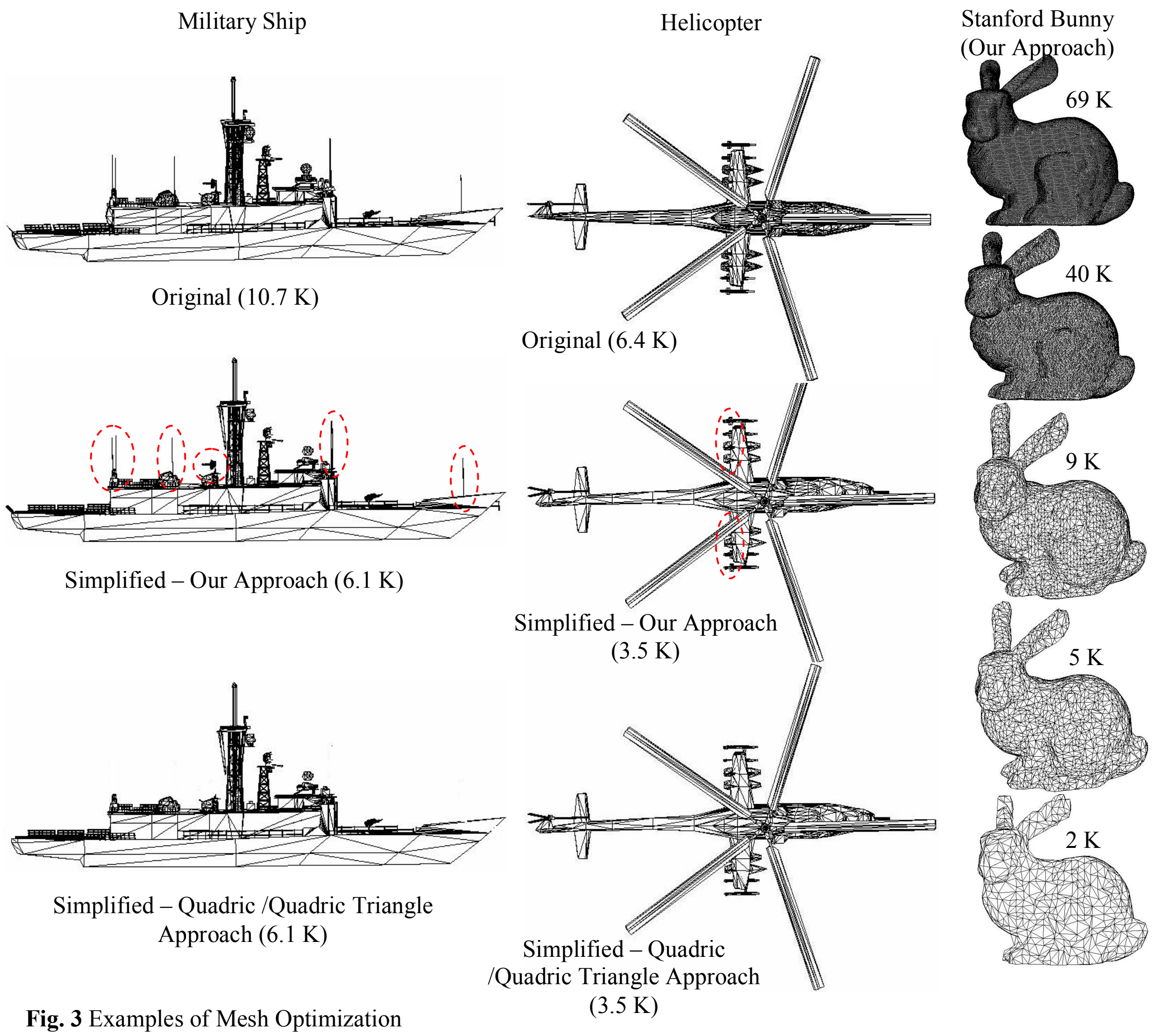
Military Ship
Original (10.7 K)
Simplified – Our Approach (6.1 K)
Simplified – Quadric /Quadric Triangle Approach (6.1 K)
Helicopter
Original (6.4 K)
Simplified – Our Approach (3.5 K)
Simplified – Quadric /Quadric Triangle Approach (3.5 K)
Stanford Bunny (Our Approach)
69 K
40 K
9 K
5 K
2 K

Fig. 3 Examples of Mesh Optimization

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 Proceedings), 19–26, (1993)
4. P. Lindstrom., G. Turk.: Fats 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. Modelling 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)