Journal of Electronic Research and Application, 2025, Volume 9, Issue 2
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Web Visualization Application of Large Mesh Models Based on Simplification Algorithms

Shengtai Shi*

Cranfield University, Cranfield MK43 0AL, United Kingdom

***Corresponding author*: Shengtai Shi, shengtai.shi63@gmail.com

Abstract: This paper studies polygon simplification algorithms for 3D models, focuses on the optimization algorithm of quadratic error metric (QEM), explores the impacts of different methods on the simplification of different models, and develops a web-based visualization application. Metrics such as the Hausdorff distance are used to evaluate the balance between the degree of simplification and the retention of model details.

Keywords: QEM algorithm; Mesh simplification; WebGL rendering; Java web development

Online publication: April 3, 2025

1. Introduction

In the field of industrial design, 3D modeling enables designers to simulate and inspect product designs in detail before manufacturing. This improves the accuracy of designs, helps to detect design flaws in advance, and reduces subsequent costs [1]. High-precision 3D models can also be used in engineering design for precise load and stress analysis and to verify the integrity of structures. Although elaborate models can bring more accurate simulations, the performance degradation, high memory consumption, reduced interactivity, and increased costs caused by a large number of polygons cannot be ignored. This paper aims to balance the simplification rate and the retention of model details according to actual needs while maintaining visual quality. The simplification algorithm is combined with web applications, and the rationality of the final product is evaluated through data processing, calculation, and visualization.

1.1. Theoretical basis

Vertex clustering: The mesh is evenly divided into multiple partitions, and all vertices within a partition are clustered into a single point, usually the centroid [2,3]. However, uniform cell division sometimes cannot fit the original mesh well. In some cases, a vertex tree system is used to divide more strictly to conform to the original mesh [4].

Vertex deletion: It is an incremental reduction method. After deleting a vertex, triangles are regenerated to patch the resulting holes [5], aiming to gradually reduce the complexity.

Based on traditional simplification methods, there are also some innovative solutions:

(1) Mesh reconstruction and resurfacing: New vertices are evenly distributed on a coarser mesh to approximate the original topology and replace the original mesh [6].

(2) Symmetry-aware algorithms [7].

1.2. Quadratic error metric method

The quadratic error metric method studies the impact of geometric errors caused by vertex movement by defining a quadratic error matrix for each vertex and describing the deviation of the vertex from the plane it lies on, thereby guiding the simplification work [8].

Representing the plane equation with a matrix, we can get:

$$P = [abcd]^T \tag{1}$$

In the formula, a, b, and c are the components of the plane normal vector, determining the direction of the plane in 3D space, and d is the offset of the plane, representing the distance of the plane from the origin in the direction of the normal vector.

For a vertex, its initial error matrix is a zero-matrix. In a 3D model, this vertex may be connected to multiple triangular faces. By summing the contribution values of all the faces passing through this vertex, we can calculate the required error matrix Q_i, its form is as follows:

$$Q_i = \sum_{F \in \mathcal{F}(v_i)} \mathbf{pp}^T \tag{2}$$

In the formula,$F(v_i)$ represents all the faces passing through the vertex v_i.

To simplify the mesh, it is necessary to identify and select all possible vertex pairs for merging. The position of the merged vertex needs to be calculated to ensure the minimum error, and this metric can be represented by the error matrix. The algorithm traverses all vertex pairs, calculates the error of each merging pair, and then selects the pair with the minimum error for merging. During this process, common priority queues such as Min-Heap can be used to improve the efficiency of selecting vertex pairs. After selecting the vertex pair and determining the merging position, the mesh structure is updated, the redundant vertex is deleted, all the faces related to it are reconnected, and the error matrix of the new vertex is calculated again [9].

1.3. Hausdorff distance

The Hausdorff distance is commonly used in fields such as geometry, image processing, and computer vision. It defines the similarity or dissimilarity between two point sets and is a measure of the maximum distance between two point sets [10].

The shortest distance from any point a_i in set A to any point in set B is d_i. The longest di among these distances is selected as the Hausdorff distance between sets A and B [11].

$$\mathrm{h(A,B)} = \max_{a \in A} \left\{ \min_{b \in B} \{d(a,b)\} \right\} \tag{3}$$

The Hausdorff distance is also applicable when A and B are polygons instead of discrete point sets [12]. The conventional Euclidean shortest distance only applies to one vertex of each polygon, while the Hausdorff distance takes into account all other vertices of the polygon. The Hausdorff distance is sensitive to the distance between the positions of two polygons but not sensitive to the shortest distance between the positions of polygons [13]. In this paper, it is used to show the local maximum mismatch between the two models.

2. Research plan

2.1. Simplification algorithm

The algorithm separately defines the processing of triangles in 3D space and stores the information of their vertices in a pointer array. By defining sine and cosine values, a series of rotations are performed on the triangle so that one of its vertices falls at the origin of the coordinate system and the other two vertices are located on the corresponding coordinate axes. This process flattens the triangle in 3D space to simplify subsequent geometric calculations and can also determine whether the target is a valid triangle.

The simplification process is divided into three main steps: loading the mesh, simplifying the mesh, and updating the mesh. By default, the quadratic error metric is used as a guide to calculate the error of each edge, then the optimal position of the vertex is calculated, and a new vertex is generated. If two vertices have the same boundary attributes, they will be merged. A variant merging method that does not generate new vertices is derived from this method. It selects the vertex with a smaller error from the original two vertices as the merged position. On this basis, after calculating the errors, the errors are sorted, and the edges with the smallest errors are processed first, which improves the efficiency and accuracy when the simplification rate is fixed. Four variants of the quadratic error metric can be obtained according to whether the errors are sorted and whether new vertex positions are generated.

Shorter edges often have less impact on the appearance of the model, which can also be used as one of the criteria for merging. Calculate and save the Euclidean distances of the three sides of each triangle, and save the index of the shortest side to a new triangle array by comparison to obtain the shortest-edge algorithm.

The last type of method is the Melax threshold method. It uses Melax's error-calculation method to combine the length and curvature of the edge. The calculation method of the edge length is as described above, and the curvature is obtained by calculating the minimum angle between the normal of each triangle adjacent to the two vertices of the edge. Finally, the calculated edge length and curvature are multiplied as the error [14].

In the mesh update, the first iteration identifies and marks all the points on the boundary to prevent damage to the model. If the current iteration is not the first iteration, the triangles marked for deletion are first removed, the remaining triangles are moved forward in the array, then the references to vertices and triangles are initialized, the starting position of the triangle references for each vertex in the array is calculated based on the reference count, and the reference information of each triangle vertex is written. In addition, to make the model simplification more stable, some auxiliary functions are added to the algorithm, such as checking whether the removal of an edge will cause the triangle to flip.

To quantify the error, it is necessary to calculate the bounding box of the model and the length of its diagonal, then derive the error coefficient according to the simplification level, and finally divide the diagonal length by this error coefficient to obtain the scaled error range, which is used to control the accuracy of error calculation. Any distance between a vertex and a triangle that is less than this scaled error range is considered

to be within an acceptable error range [15]. The Hausdorff distance calculated using the vertex set of one model and the triangle-face set of another model in the common bounding box of the two models can reflect the large deviations in the local area.

2.2. Web visualization application

The main content area contains the structure and layout of the entire page and initializes the application when the page is loaded. It contains two canvas areas and a model information display area. The models are rendered through WebGL and can be interacted with separately. The model information mainly includes the number of vertices and triangles and the file size to visually compare the differences before and after simplification from a data perspective.

The head of the web page is responsible for linking and referencing external resources and initializing default settings. Meta tags are used to adjust the display character encoding and viewport configuration to ensure that the web page works properly on different types of mobile devices. Structurally, the rendering and interaction functions are achieved by embedding the WebGL renderer and JS functions in the head section so that they can be called on the web page. The fragment and vertex shader information required for each of the two models is also stored in this section. The body section only considers the layout of the web page and makes the user interface more reasonable and friendly by adding elements and components.

The JS code section covers the main functions such as managing user interactions, buffering and loading models, event handling, and interface updates. It processes a series of logical operations and feeds back the results to the interface for viewing. For example, it reads the file content and parses it into JSON data, loads and compiles shaders, creates WebGL program objects, and links them to the corresponding WebGL context.

3. Result output and analysis

When the simplification rate is low, the overall appearance and local details of the model are well-preserved, with almost no obvious differences. When the simplification rate is increased to 50%, the edges of the model become sharper. This change is particularly evident on surfaces with large curvatures, such as the edges of the rabbit's ears, which change from smooth curves to distinct straight-line segments, while details such as dents are still well-preserved as shown in **Figure 1**.

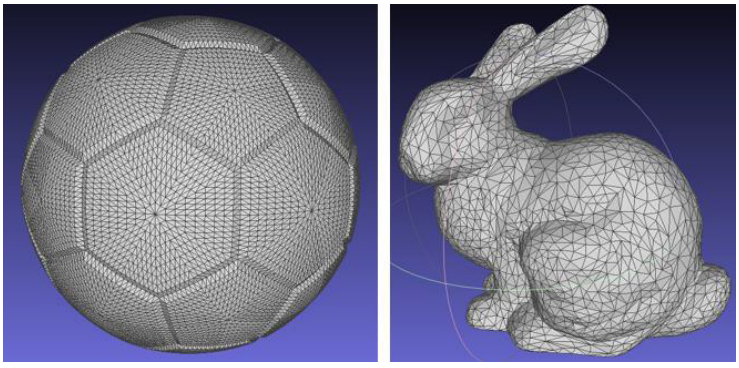

Figure 1. Grid comparison football – rabbit

Under high-level simplification, the appearance of the model becomes further blurred, and some

deformation occurs. The number of line segments forming the curves is further reduced, resulting in a further decrease in smoothness. In addition, most geometric details, such as the eyes, ears, and dents on various parts of the body, will be lost. For simple models, due to the small number of triangles themselves, the remaining triangles after high-level simplification are not enough to support all the detailed features. To ensure the overall appearance, the algorithm preferentially discards these details. Based on observations, the simplification results often visually affect the surfaces with large curvatures first, causing obvious defects. To analyze this feature, a sphere model composed of uniform continuous surfaces was selected for comparative research.

For a sphere model with a continuous and uniform surface, even under high-level simplification, the visual performance remains good. The mesh shows that such graphics often have a high degree of symmetry and consistency, and the numerical error distribution of the uniform mesh is more consistent, which helps to improve stability and overall accuracy. The topological errors caused by the removal of triangles are also evenly distributed throughout the model, so the local and overall appearance changes are also mitigated. In contrast, the mesh triangles of the rabbit model vary greatly. The triangles in the gentle slope area on the back are relatively uniform. Similarly, in the three simplification levels, this part is the least visually affected, while areas with significant triangle changes, such as the ears and concave surfaces, are the most deformed parts in the simplified model, which is consistent with the above-derived and observed phenomena.

In addition to the influence of the model structure and mesh characteristics on the visualization of the simplified model, when the original model is fine-grained enough, the number of remaining triangles after simplification will be larger, which may make the simplified model more similar to the original model. Therefore, a Buddha statue model with both high-frequency details and large, flat surfaces was selected for subsequent experiments as shown in **Table 1**.

Table 1. Simplified data of Buddha statues

	Original model	**20% simplification**	**50% simplification**	**90% simplification**
Number of vertices	149970	120006	75036	15000
Number of triangles	100000	80000	50000	10000
File size	3062.54 kb	2436.33 kb	1497.79 kb	277.46 kb

In actual results, QEM performs similarly when processing complex and simple models. It does not improve the visual effect due to the increase in the number of triangles and exposes the problem of poor handling of high-frequency details under high-level simplification. Although it is very effective in maintaining the overall shape, it often erases almost all small details. For example, bumps and dents blend in with the adjacent large planes and are difficult to distinguish. QEM does not distinguish between areas with rich and sparse details. Therefore, even considering error minimization, vertex merging will still lead to loss of details.

Analysis of the data shows that the Hausdorff error increases with the increase in the degree of simplification, which means that the gap between the output model and the original model gradually widens. When the degree of simplification is not large (less than 60%), the error increase is relatively gentle. When the degree of simplification is high (more than 60%), the error will increase significantly. Although the file size is compressed, the visual effect may become unacceptable. The positive impact of the uniformity of the model mesh on the simplification results is also well-reflected in the line chart. The error change of the football model is more controllable. Without considering model specificity, QEM has a superior simplification

performance. Sorting the triangles can slightly improve the calculation efficiency, and the selection of whether to generate a new vertex merging position can be made according to requirements. To balance the convenience of transmission and model quality, in most cases, a simplification rate of no more than 60% is appropriate. In cases where there are fewer details or only the appearance of the model is emphasized, a higher simplification rate can be selected as needed.

4. Review and summary

This paper analyzes the ideas and processes of model simplification around the QEM algorithm and compares it with other common methods to show their respective applicable scenarios, advantages, and disadvantages. To make model simplification more user-friendly, a series of model processing processes from simplification, transformation, to visualization are proposed. Through experiments, the developed algorithm can simplify well according to the set compression ratio.

In the future, it is expected to further integrate the three processing processes into the web end to achieve high-level integration and automation. Based on the characteristics of various algorithms and their performances at different simplification degrees, users can be provided with references when choosing the simplification ratio.

Disclosure statement

The author declares no conflict of interest.

References

[1] Durk J, 2019, Model Simplification in Manufacturing Simulation – Review and Framework. *Computers & Industrial Engineering,* 127: 1056–1067.

[2] Arvo J, Euranto A, Jarvenpaa L, et al., 2015, 3D Mesh Simplification: A Survey of Algorithms and CAD Model Simplification Tests, University of Turku, Technology Research Center, Finland.

[3] Rossignac J, Borrel P, 1993, Multi-resolution 3D Approximations for Rendering Complex Scenes, Springer Berlin Heidelberg, 455–465.

[4] Erikson C, Luebke D, 1997, View-dependent Simplification of Arbitrary Polygonal Environments. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’97), New York, NY, USA, 199–208.

[5] Schroeder W, 1997, A Topology Modifying Progressive Decimation Algorithm, *IEEE Visualization ’97*, Phoenix Arizona USA, 205–212.

[6] Voutchkov I, Keane A, Shahpar S, et al., 2018, (Re-)Meshing Using Interpolative Mapping and Control Point Optimization. Journal of Computational Design and Engineering, 5(3): 305–318.

[7] Podolak J, Funkhouser T, Golovinskiy A, 2009, Symmetry-aware Mesh Processing, Mathematics of Surfaces XIII*, Springer Berlin Heidelberg*, 170–188.

[8] Heckbert P, Garland M, 1997, View-dependent Simplification of Arbitrary Polygonal Environments. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’97), New York, NY, USA, 209–216.

[9] Hugues H, 1996, Progressive meshes. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96), Association for Computing Machinery, New York, NY, USA, 99–108.

[10] Klanderman G, Rucklidge W, Huttenlocher D, 1993, Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 15(9): 850–863.

[11] Munkres J, 1999, *Topology*, 2nd ed, Upper Saddle River, NJ, USA, Prentice Hall.

[12] Rucklidge W, 1997, Efficiently Locating Objects Using the Hausdorff Distance. *International Journal of Computer Vision,* 24(3): 251–270.

[13] Vito D, Valery S, 1999, Distance-based Functions for Image Comparison. *Pattern Recognition Letters*, 20(3): 207–214.

[14] Stan M, 1998, A Simple, Fast, and Effective Polygon Reduction Algorithm. *Game Developer Magazine*, 5(11): 44–49.

[15] Taha A, Hanbury A, 2015, An Efficient Algorithm for Calculating the Exact Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11): 2153–2163.