International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

Mesh Simplification Algorithms for Rendering Performance

Hongle Li[1], Seongki Kim[2]

[1] *Ph.D., Students, Dept. of Computer Engineering, Keimyung University, 1095, Dalgubeol-daero, Dalseo-gu, Daegu, Republic of Korea. ORCID: 0000-0001-7857-1020*

[2] *Assistant Professor, Division of SW Convergence, Sangmyung University, Seoul, South Korea. ORCID: 0000-0002-2664-3632*

Abstract

In virtual worlds simulated by computers, all objects consist of complex 3D meshes, and it is important to simplify the meshes for rendering performance while maximally maintaining the original geometry. This research conducts a series of studies on the simplification of the triangular mesh model and proposes four different simplified algorithms and two hybrid algorithms. All of these algorithms are based on vertex aggregation, vertex removal, and iterative edge folding, and the common principle is to reduce vertex data in the original model that is not important to the model geometry. After reducing the data, the point cloud is retriangulated. This research implements these six algorithms and conducts an experiment. With the simplification algorithms in this paper, rendering performance improves by 25% ~ 35% while maintaining the geometry of the model to the greatest extent. To the best of our knowledge, the proposed algorithms in this paper (*VT*, *FAD*, *SS*, *VTD*, *FASS*) are the world-first.

Keywords: Triangular mesh, Mesh reconstruction, Mesh simplification, Model rendering

I. INTRODUCTION

To render a 3D scene into generally used 2D devices such as monitors and HMDs (head-mounted displays), triangles have been widely used because of their high efficiency during rendering and have succeeded in games, VR (virtual reality), AR (augmented reality), simulation, and other fields. However, the detailed triangular mesh models usually require computers to deal with complex and large quantities of data, which decreases application rendering performance.

In many cases, such as rendering or simulation, it is not always necessary to use the highest complexity model to render all parts of a scene. For example, distant mountains, rivers, and other backgrounds in game scenes and nongazing objects in foveated rendering do not have to be rendered with the full details. For example, in a scene from the PUBG (PlayerUnknown's Battlegrounds) game, when the player operated game sprite advances toward a target, objects around the sprite, whether it is a house, box, or grass, should be rendered in detail, but the mountains and rivers far from the sprite do not need to be carefully rendered due to their long distances. It is enough to coarsely render these objects in scenes where the user will not notice. Similarly, in foveated rendering, the objects that the human eye looks at should be rendered in more detail. Objects that the human eye does not focus on do not need a detailed rendering, which can improve computing performance and reduce the computational burden of the calculations.

As a result, it is important to choose the appropriate complexity according to the needs of actual rendering and to effectively simplify the triangular mesh model under the premise of keeping as much of the original geometry as possible for the rendering performance [1, 2].

In the past 50 years, a series of representative algorithms and technical theories have emerged for mesh simplification [1-2, 4-6, 13-17]. Classifying many mesh simplification algorithms according to the simplified features, the mesh simplification algorithms can be largely categorized into the following: the adaptive subdivision algorithm, the geometric element removal method, and the simplified sampling algorithm. The adaptive subdivision algorithm [6] achieves the goal of simplifying the model by generating edge points and moving the original vertices, continuously approximating, and finally generating a smooth surface with a continuous tangent plane. The geometric element removal method [7] deletes the geometric elements inside the mesh model according to the topology and geometric characteristics of the model mesh and the specified geometric elements to reduce the number of vertices to achieve the effect of model simplification. The simplified sampling algorithm [8] defines the probability distribution function based on the local geometric eigenvalues of the model, determines the triangles to be addressed, and adjusts the internal triangles of the mesh model to achieve the goal of simplification. Among them, the geometric element removal method is quite mature after previous studies, and it is one of the algorithms with relatively high simplification efficiency [2], but additionally, the previous algorithms are relatively limited. Therefore, based on the premise of geometric element removal, this study proposes some new ideas.

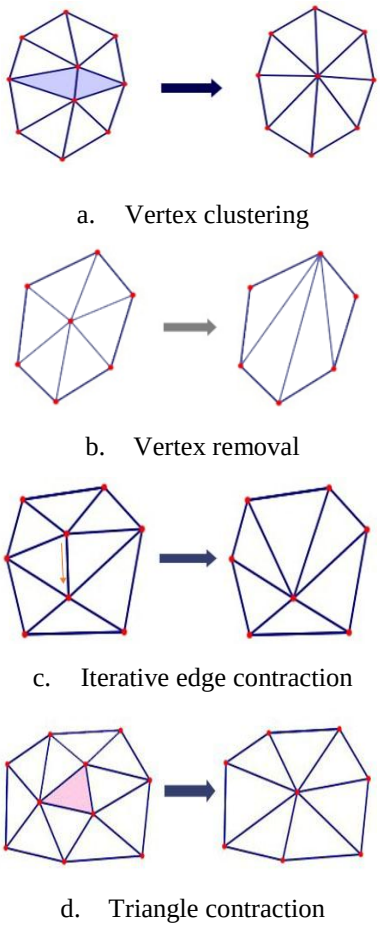
This research offers the following major contributions to 3D geometry research. First, this research proposes four different mesh simplification algorithms and two hybrid simplification algorithms based on the geometric element deletion method from three aspects. Second, this research proposes the *FAD* algorithm based on vertex removal and the *SS* algorithm based on spatial vertex clustering. Third, this paper studies and proposes a hybrid algorithm that combines different algorithms. These hybrid algorithms are merged based on the algorithms that have been proposed before. Compared to a single algorithm, the degree of grid optimization is improved, the hybrid algorithms complement each other, and the performance of these algorithms is almost the same as the performance of a single algorithm.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

This paper is organized as follows. Section 2 introduces the background of the algorithms and related works. Section 3 implements four independent algorithms and the two hybrid algorithms according to the order of algorithm proposals. Section 4 evaluates the six algorithms separately and compares the test data to prove the functions and describe the advantages and disadvantages of the algorithms. Section 5 summarizes the results of the algorithms.

II. RELATED WORK

The essence of mesh simplification is to minimize the number of triangles and vertices of the original model while maintaining as many of the features of the original model as possible. Therefore, it usually includes two principles: the principle of minimum vertices, which minimizes the number of fixed points of the simplified model given the upper error limit; the principle of minimum errors is to minimize the error between the simplified model and the original model given the number of vertices of the simplified model [9]. The minimum error refers to the error of the overall model calculated by expressing each edge of the original model with a fixed metric value, and comparing the metric value of each simplified side; the smaller the error, the closer the simplified model geometry is to the original model. Combined with the research results in the field of mesh simplification, geometric element deletion methods mainly include the vertex clustering method, vertex removal method, iterative edge contraction method, and triangle contraction method. Figure 1 shows each method.



a. Vertex clustering

b. Vertex removal

c. Iterative edge contraction

d. Triangle contraction

Fig. 1. Simplified classification of meshes based on geometric elements

Vertex clustering aggregates two or more vertices in a mesh model. In 1993, Ressignac et al. proposed a vertex clustering method in [13], which has a profound impact on the subsequent improvement of the algorithm.

[13] used a bounding box to surround the original model and then divided the bounding box into several regions; then, the vertices of the original mesh model fall into these subdivided regions. After this subdivision, on the premise of not affecting the basic geometry, the vertices in the unified region are merged to the greatest extent.

Vertex removal removes some vertices. In 1992, Schroeder et al. [14] proposed a classic deletion algorithm, which defined the standard of vertex deletion as the distance from the vertex to the plane or edge. If the distance is less than the specified threshold, the vertex is deleted; otherwise, the vertex is retained. A hole can be created at the position where the vertex is removed. After that, the hole should be recovered.

Because the removal condition is single, the original model is largely changed, especially in the sharp parts of the original model. It is difficult to maintain the original contour features because it only addresses the distance between the vertex and the surrounding planes or edge and does not consider the angle between the vertex and the surrounding planes. In addition, the degree of model simplification in this algorithm is often difficult to determine. In different positions of the same mesh model, due to different levels of detail, different control thresholds are usually required.

Most of the subsequent algorithms attempted to improve these issues. For example, a curvature-based algorithm was proposed in [10], and a new algorithm based on the half-edge structure was proposed in [11].

Iterative edge/triangle contraction iteratively reduces the edge or triangles. During the calculation of each iteration, two endpoints of the selected edge are gradually decreased to a single vertex, and the triangle that meets the requirements is decreased to a new vertex instead of the original triangle. The newly generated fixed points are combined with the vertices of the original model, and retriangulation is performed.

QEM (quadratic error metrics) [5] is the most representative algorithm among these methods. This algorithm calculates a quadratic error matrix for each vertex in the original mesh model and calculates the cost of edge folding in the mesh and the position of generating new vertices according to the error matrix; then, the folding operation is performed according to the folding cost from small to large. The new error matrix generated by the folding is the sum of the secondary error matrices of the two vertices of the folded edge. However, because only the cost of edge contraction is considered in the implementation of this method, the QEM algorithm generates many narrow and long triangular faces, so it is insufficient for maintaining the detailed features of the original model [15].

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

III. ALGORITHM DESCRIPTION

The six different mesh reduction algorithms proposed in this paper are as follows: the "*vertex triplex (VT)*" algorithm based on vertex clustering; the "*face angle dependency (FAD)*" algorithm based on vertex removal; the "*intervertex distance dependence (IVDD)*" algorithm based on iterative edge contraction; "*Spherical Scanning (SS)*" algorithm that uses vertex clustering and iterative edge contraction; the "*vertex triplex distance (VTD)*" algorithm and the "*face angle spherical scanning (FASS)*" algorithm based on the advantages and disadvantages of each algorithm. In this section, this research describes the basic ideas of these six algorithms.

III.I Vertex Triplex (VT)

The core of this algorithm is to calculate the center of the circumcircle of the triangle. The center of the triangle's circumcircle is a point with an equal distance from the three vertices of the triangle (as shown in Figure 2). Therefore, if the three vertices of each triangle in the original triangular mesh model are aggregated into their circumscribed circle centers, the resulting point will be evenly distributed on the original model.

During the implementation of the algorithm, this research divided it into three steps: first, load the triangular mesh model, and second, calculate the position of the circumscribed circle center of each triangular mesh according to the triangular mesh relationship existing in the model. The triangle mesh units are aggregated into the center of the circumscribed circle. Finally, all the triangular meshes are retriangulated.

In this research, it is assumed that the coordinates of the three points in space are $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$, and $P_3(x_3, y_3, z_3)$. The analysis shows that two constraints need to

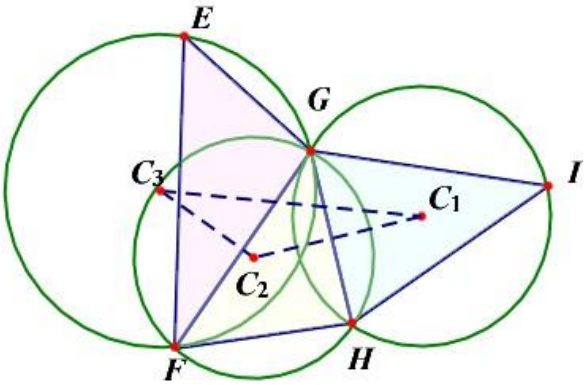


Fig. 2. Schematic diagram of a triangle outer center

be met to calculate the circumcenter of a triangle: three points are coplanar, and the distances from the three points to the space center coordinates are equal. Then, 4 free terms and 4 equations can be obtained from these two constraints, so the following four linear equations (1), (2), (3), and (4) can be listed and solved using the elimination method. This research assumes that the coordinates of the center of the circle to be calculated are (x_0, y_0, z_0), and the radius is R. For a three-point coplanar constraint, the plane equation determined by three points in space is

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

$$\Rightarrow A_1 x + B_1 y + C_1 z + D_1 = 0 \qquad (4)$$

$$A_1 = y_1 * z_2 - y_1 * z_3 - z_1 * y_2 + z_1 * y_3 + y_2 * z_3 - y_3 * z_2$$

$$B_1 = -x_1 * z_2 + x_1 * z_3 + z_1 * x_2 - z_1 * x_3 - x_2 * z_3 + x_3 * z_2$$

$$C_1 = x_1 * y_2 - x_1 * y_3 - y_1 * x_2 + y_1 * x_3 + x_2 * y_3 - x_3 * y_2$$

$$D_1 = -x_1 * y_2 * z_3 + x_1 * y_3 * x_2 + x_2 * y_1 * z_3 - x_3 * y_1 * z_2 - x_2 * y_3 * z_1 + x_3 * y_2 * z_1$$

The constraints of equal distance from the three points to the center of the circle can also obtain the following equations:

$$\begin{cases} R^2 = (x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 & (1) \\ R^2 = (x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2 & (2) \\ R^2 = (x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2 & (3) \end{cases}$$

The linear equations about the coordinates of the center of the circle can be obtained by eliminating the above four linear equations:

$$\begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} = 0$$

The calculated center coordinates are:

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = -\begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix}^{-1} \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

The radius is:

$$R = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$$

Using the equations, the eccentric coordinates of each triangle in the model can be efficiently obtained, and finally, a point composed of eccentric coordinates can be generated. These points together constitute the point cloud data of the model, but the newly generated point cloud data do not include triangulated index data. Compared with other triangulation algorithms, Delaunay's triangulation algorithm has the following advantages: these triangles are as equiangular as possible, thus eliminating the potential numerical accuracy problems caused by slender triangles; ensure that any point on the surface is as close to the node as possible; triangulation is not related to the order of processing points; and faster processing speed [12]. Therefore, this research study retriangulates the points using Delaunay triangulation.

III.II FAD (Face Angle Dependency)

The *FAD* algorithm is based on the idea of vertex removal. It is implemented by calculating the angle between the triangle

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

surfaces associated with the vertices. This research divides the implementation of the algorithm into three steps:

1) Query and calculate the angle between all triangle faces in the model related to the current vertex
2) Evaluate and adjust the current vertex based on the specified threshold conditions
3) Triangularize the holes generated after data adjustment

Querying and calculating the angle between triangular faces is the first step in this algorithm. When calculating the angle between the triangle faces, the index data of other vertices related to the currently selected vertex are extracted into the memory container by traversing the vertex number as a unit. After finding the corresponding 3D coordinates through the index data, the normal information of the triangle is calculated. Finally, the normal information is calculated in pairs to obtain the size of the included angle, and the included angle data are stored separately.

There are many methods for calculating the included angle, and in [16, 17], similar research was conducted on the *FAD* algorithm. However, in [16,17], the research only determined the included angle of the plane related to the current vertex, and all the included angles were larger than the threshold, and then the next operation was performed. The new *FAD* algorithm proposed in this paper can control the simplified process according to the two conditions of the area and the angle of the plane related to the current vertex. In this paper, this research uses the same algorithm, and it is also the simplest and most understandable method for calculating the included angle. That is, by using two intersecting vectors on a plane, you can find normal vector information (normal) perpendicular to the two vectors. Taking three points in the space as $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$, $P_3(x_3, y_3, z_3)$ and setting the normal to (d_x, d_y, d_z), the normal meets the following equation:

$$(x_2 - x_1) * d_x + (y_2 - y_1) * d_y + (z_2 - z_1) * d_z = 0$$

$$(x_3 - x_1) * d_x + (y_3 - y_1) * d_y + (z_3 - z_1) * d_z = 0$$

$$(x_3 - x_2) * d_x + (y_3 - y_2) * d_y + (z_3 - z_2) * d_z = 0$$

After solving the system of equations to obtain the plane normal vector from the above equation, the angle from the angle of the vector can be finally calculated:

$$\cos\theta = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2+y_1^2+z_1^2} * \sqrt{x_2^2+y_2^2+z_2^2}}$$

From the above equation, the magnitude of the normal angle (as shown in Figure 3) can be obtained. Finally, the number of included angles between adjacent triangular faces obtained by this method is stored in the corresponding memory container.

Through the calculation of the first step, a container is obtained to store information on the normal angles of all triangular faces. The smaller the normal angle of the face, the larger the included angle between the faces. Therefore, the main judgment basis of this algorithm is also based on the size relationship between the current included angle and a given threshold to determine whether the currently processed vertices should be retained or removed. In this process, this research uses the method of traversing the storage angle container for evaluation. When the

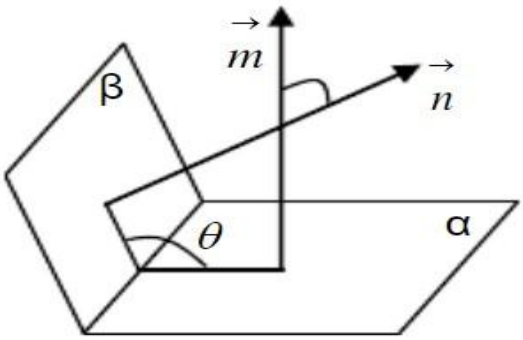


Fig. 3. Normal angle calculation and normal angle

included angle of all faces related to the selected vertex is less than the threshold, the vertex is marked as removable, and the vertex coordinates of the point are set to a specific value. Then, after all vertex traversals are completed, the specific value is deleted in the container. At this point, because the vertex is deleted, a hole is formed in this part. Finally, this algorithm uses Delaunay triangulation to retriangulate [12].

III.III IVDD (Intervertex Distance Dependence)

The basic idea of the iterative edge folding simplification algorithm is to fold the edges in the grid according to certain criteria. As shown in Figure 4, the dotted line is the edge to be folded, and the two related endpoints (P_1, P_2) are folded into a new vertex P_0 according to the criteria specified to modify the topology in the mesh model. To complete the simplification of the number of vertices, the model simplification operation is completed. One simplification in this process can reduce one edge and two triangular faces compared to the original model. The third algorithm proposed in this research is the *IVDD* algorithm, which is a new algorithm based on iterative edge folding. Therefore, it also preserves its high-efficiency characteristics, and at the same time, folds the adjacent and nearer vertices in the original model, so that the size of each triangular surface in the model tends to be similar, the effect of balancing the triangles is achieved, which makes the display of the model smoother (see test content in Section 3 for details).

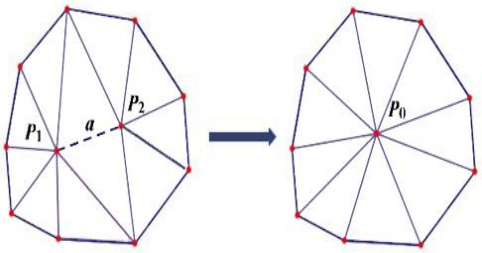


Fig. 4. Illustration of edge folding

In the specific implementation of the *IVDD* algorithm, the implementation of this algorithm is also divided into three stages. First, select the vertices and traverse the mesh model to obtain all other vertices related to the selected vertex. Second, fold the edge formed by the two closer vertices in space and calculate the center point coordinates from the two vertices to

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

replace the original two vertices. Finally, retriangulate the current part with the new vertex as the center.

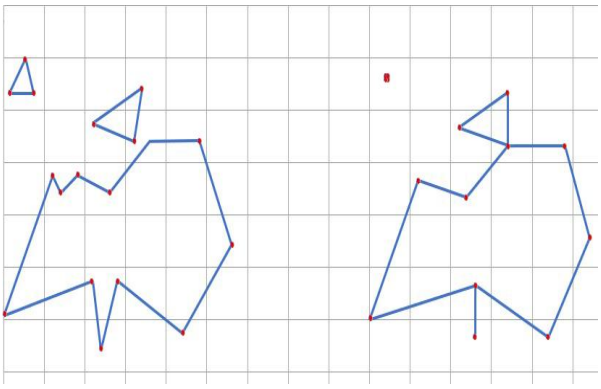
In this algorithm, the most important aspect is the edge folding process in the second step. In this process, this research first selects the vertices in the same way as the previous algorithm and uses the two points in the same triangle as the reference to calculate the distance between the two points. If the distance between one vertex and the other two vertices is less than the threshold, all three sides are folded, and the newly generated vertex is the circumcircle center of the original triangle. Then, the circumscribed circle center coordinates are used to replace the coordinate values of all three vertices that appear in the subsequent data. If the distance between a vertex and one of the triangles is less than the specified threshold, then this research calculates the coordinates of the center point of the two vertices. The center coordinates are used to replace the original values of these two vertices to fold the edges between the two vertices.

For the calculation of the distance between two points in space and the coordinates of the center point of two points, only a simple mathematical calculation is required. This research directly applies the distance formula between two points in space:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The vertex distance d can be calculated, and this d value is the judgment basis of this algorithm. For the coordinates of the center points of the two vertices in the space, the x-axis component, the y-axis component, and the z-axis component of the two vertices are added, and then the average value on the respective coordinate axes is used to obtain the center point coordinate value.

Finally, in this algorithm, all original data are also modified to obtain point cloud coordinate data composed of the modified data. Finally, this algorithm also retriangulates the points using Delaunay triangulation [12].



(a) Before simplification (b) After simplification

Fig. 5. Vertex clustering based on a 2D plane

III.IV SS (Spherical Scanning)

The *SS* algorithm is a vertex clustering method from the 2D plane to 3D space. In 2D plane vertex clustering, a mesh of a specified size is usually used to map the original vertex

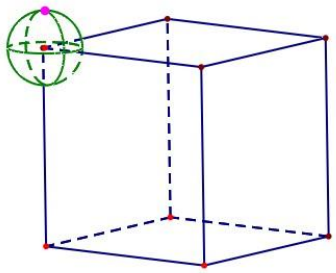


Fig. 6. Vertex clustering of the space sphere scanning

information to this mesh map. If multiple vertices fall into the same vertex at the same time as shown in Figure 5, the vertices falling into the same grid are aggregated into one vertex. Continue iteratively in the same way, detecting and querying all vertices included in the model until all points are not in the same grid.

Similarly, this algorithm extends the vertex clustering method of the 2D plane to the 3D space. Imagine that there is now model A. In 3D space, this algorithm does not use meshes for mapping but uses a sphere with a certain radius to perform a "movement scan" in the order of vertices on the original mesh model; at the same time, after moving to a vertex, the distances between the vertex and the surrounding vertices are calculated. If the surrounding vertices and the currently scanned vertices are in the "sphere", two or more vertices that fall into the same sphere are merged into a new vertex, as shown in Figure 6. The generation of new vertices is not performed in one way, and multiple methods are combined to generate the optimal new vertex coordinates. Finally, this algorithm retriangulates the changes in the geometry or mesh structure of the vertices.

III.V Hybrid Algorithm

In the design of the hybrid algorithm, this research mainly complements and combines the advantages and disadvantages of the algorithm. The *VT* algorithm has proven that although the vertex distribution of the simplified triangle model is more uniform, it does not have the ability to handle some special edge processing separately. The *IVDD* algorithm removes similar vertices or replaces them with center point coordinates. Therefore, the size of the triangles inside the geometry also tends to be similar, which can balance the triangles for special edges. Based on this, the research of hybrid algorithms is implemented by combining these two algorithms.

Similarly, the *FAD* algorithm proves that it can avoid the feature of extremely irregular triangles in the simplified model. In contrast, the *SS* algorithm based on vertex clustering in 3D space has a simple, clear, and fast simplification. However, after it folds the straight line or triangle in the mesh model, that is, the degradation of the triangle into line segments/vertices, the "protruding/tip" part of the mesh topology cannot be maintained well (for details, please refer to the content of the experimental part in Section 4). Therefore, the two algorithms are complementary and combined, and the result will be better than a single simplified algorithm.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

III.V.I VTD (Vertex Triplex Distance)

To implement this hybrid algorithm, the core ideas of the *VT* algorithm and *IVDD* algorithm and the important steps in the algorithm construction process are introduced in the previous section, so in this section, we mainly introduce how to merge the algorithms to implement the hybrid algorithm. These two different algorithms have different approaches.

After loading the mesh model into the program and storing it in a memory container, the vertex coordinates of the triangle are read according to the order of the vertex index data of the model. The previous method simplified the three vertices into one vertex (coordinates of the outer point of the triangle), but the method merged and folded the triangle vertices in different cases.

Assume that the three vertices of the triangle relationship in the space are P_1, P_2, and P_3, as shown in Figure 7. After reading the triangle data, first, use the distance formula between the two points in the space:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Calculate the distances d_{12}, d_{13}, and d_{23} between the three vertices. If the values of the three distances are all smaller than the specified threshold, then the three vertices P_1, P_2, and P_3 are merged with the original three vertices into the circumscribed circle center coordinate C_0 of the triangle according to the *VT* algorithm. If the distances d_{12}, d_{13} and d_{23} of the three vertices are not all smaller than the specified threshold, edge folding is performed according to the following procedure.

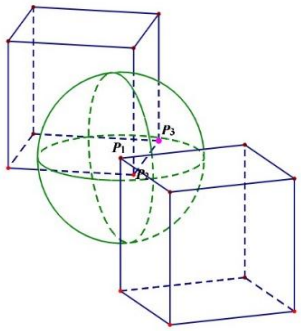


Fig. 7. Vertex simplification between different vertex spacings

First, if the distance d_{12} between the vertices P_1 and P_2 is less than or equal to the threshold, then further compare the sizes of the distances d_{13} and d_{23} between the vertices P_1, P_3 and vertices P_2, P_3. When d_{13} is larger than d_{23}, the newly generated vertex C_0 is taken from the coordinates of the center point between vertices P_1 and P_3. Conversely, if d_{13} is smaller than d_{23}, the coordinates of the newly generated vertex C_0 are replaced by the coordinates of the center point between the vertices P_2 and P_3. Using the same principle, if d_{13} is less than the specified threshold, the distance between the other two vertices is determined, and the two vertices with the larger vertices are merged into their center point coordinates according to the distance. The same applies to the calculation and data replacement process of d_{23}.

III.V.II FASS (Face Angle Spherical Scanning)

For the *FASS* algorithm, since the *FAD* algorithm is based on vertex removal and the *SS* algorithm is based on vertex aggregation, its main purpose is to reduce the number of vertices in the model. Therefore, in the fusion process of the two algorithms, this algorithm first uses *FAD* to determine whether any vertices should be removed. Then *SS* is used to perform vertex clustering on all vertices, which can reduce the number of vertices in the triangle model to a greater extent.

In the implementation of the specific algorithm, after the model is loaded and the data are read, all relevant vertex indexes of the current reference vertex are determined according to the process of the *FAD* algorithm, and all indices are stored in the memory container. Then, the included angle of the triangular surface formed by these vertices is calculated and the relationship between these included angles and the specified angle threshold are compared. If all the included angles are greater than the specified included angle threshold, then in the original mesh model, the "bulge" of the transition where the current vertex sits is relatively small, or the part is not a "tip" part, then the vertex at that position is directly deleted. Additionally, the *SS* algorithm is used to determine the remaining vertices. When there are vertices in the remaining vertices that fall into the same ball, all the vertices inside the ball are merged into one vertex to replace the original vertex in the ball.

IV. IMPLEMENTATION AND RESULT

The suggested algorithms were tested for their rendering performance. To obtain results, this research designed 14 sets of controlled trials and simplified and rendered performance tests were performed using a complex Stanford's bunny model. Each set of experiments was performed 10 times on two different machines.

IV.I Test Environment

First, before the test started, this research selected two computers with different configurations as the test environment; the configurations are shown in Table 1.

Table 1. Configuration of the test PCs

PC 1	CPU	Intel Core i7-8750H CPU @ 2.20 GHz
	Memory	16 GB
	OS	Windows 10 Enterprise Edition X64
	GPU	NVIDIA GeForce GTX 1060
PC 2	CPU	Intel Core i3-4170 CPU @ 3.70 GHz
	Memory	4 GB
	OS	Windows 10 Education X64
	GPU	Intel HD Graphics 4400

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

IV.II Test Models

This research used *Stanford's bunny* as the test model. *Stanford's bunny* mesh model has the characteristics of complex details, small morphology, and dense spatial distribution. The model's vertex and triangulations are summarized in Table 2 below.

Table 2. Raw model information

Item	Stanford's Bunny
Vertex	34,032
Vertex index	196,890
Triangles	65,630

IV.III Test Results

If the model is rendered directly on the screen, the rendering speed of the model is limited by the refresh rate of the computer screen. Therefore, this paper uses off-screen rendering (FBO rendering) for all tests to reduce the interference of external factors on the rendering efficiency measurement. After the test started, this research first performed off-screen rendering of the original models of the two test models and then wrote the off-screen rendered images to the local hard disk (the overall and locally rendered images are shown in Figure 8 below). Then, we recorded the number of vertices and triangles remaining after reduction by different algorithms in this research and also recorded the total time of 1,000 renderings from the 100th to the 1,100th in the rendering process and calculated the average rendering time, as shown in Table 5.

For the off-screen rendering of the original model, after testing on two differently configured PCs, to ensure the accuracy of the data obtained from the test, this research performed 10 tests on the same parameters. In the test, the average time consumption of the 100th to 1,100th rendering was calculated and recorded. The results are summarized in Table 3 below.

Table 3. The average time taken to render 1,000 primitives on different PCs

Item	Stanford's Bunny
PC 1	27.29
PC 2	81.29

unit is msec

In Table 3, on PC1, using Stanford's bunny mesh model as the basic off-screen rendering, 1,000 renderings required an average of 27.29 msec, and the frame rate reached 36,743 FPS; on PC2, rendering 1,000 frames, the average time was 81.29 msec, reaching 12,302 FPS.

IV.IV Performance Results

In terms of rendering efficiency, from the perspective of rendering FPS, the improvement of the *VT* algorithm is the most obvious on PC1 or PC2 (see Figures 9 and 10 for details). Figure 11 and Figure 12 show the percentage increase in the rendering rate of each algorithm under different parameter conditions and on different PCs.

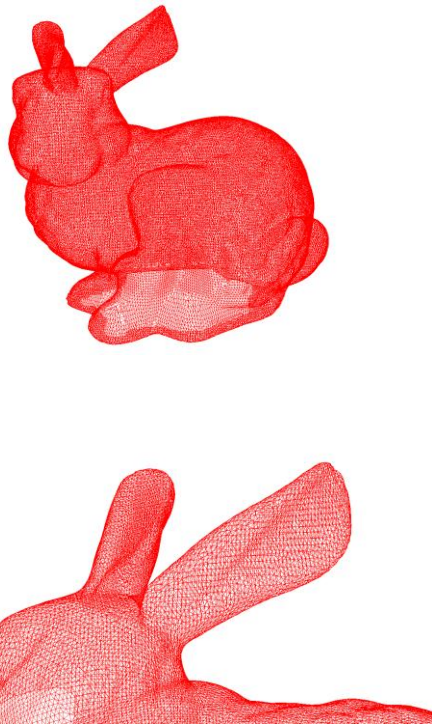


Fig. 8. Panorama effect (top) and local details (bottom) rendered by the original model

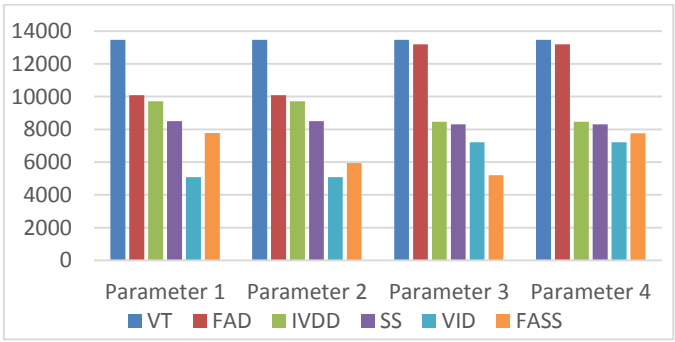


Fig. 9. Performance increase FPS of Stanford's bunny model on PC 1

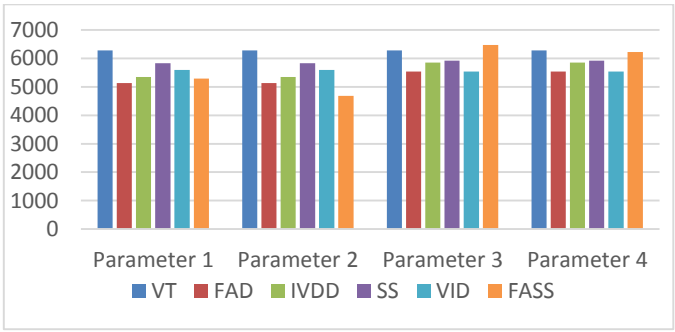


Fig. 10. Performance increase FPS of Stanford's bunny model on PC 2

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

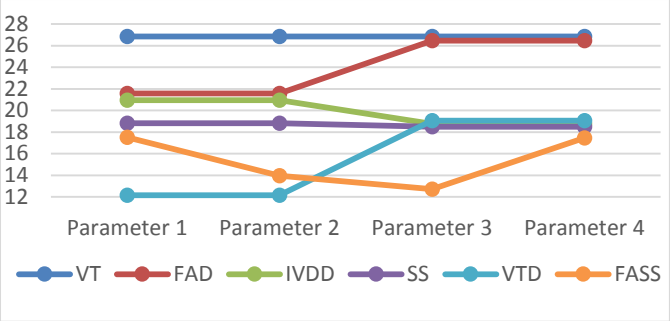


Fig. 11. Efficiency improvement when rendering Stanford's bunny model on PC 1

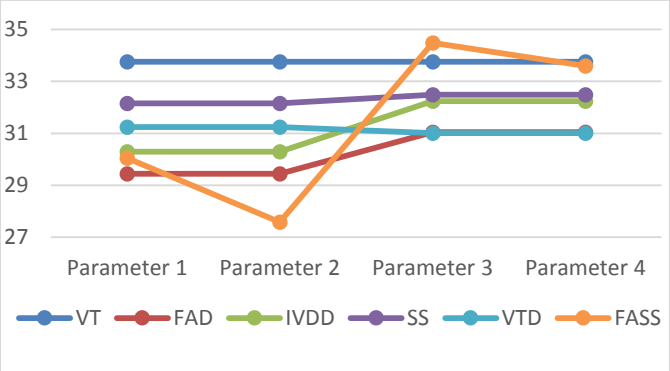
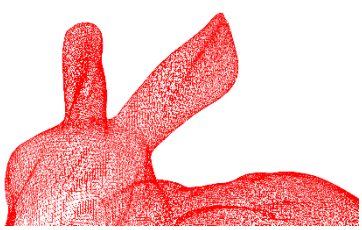


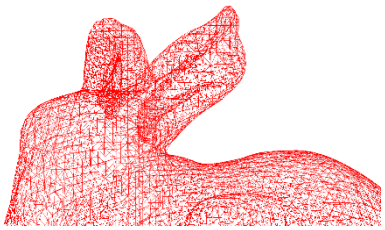
Fig. 12. Efficiency improvement when rendering Stanford's bunny model on PC 2

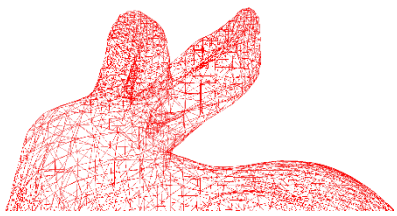
From the test data in the chart, the algorithm proposed in this paper greatly contributes to rendering efficiency and can effectively reduce the rendering time of the model on the computer. In general, the average efficiency of the six algorithms overall improved by approximately 26%. (See Figure 9 for details).

IV.V Rendering Results

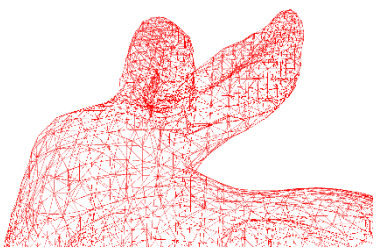
Whether it is a single simplified algorithm or a final hybrid algorithm, the algorithm can quickly simplify and improve rendering speed while preserving the geometry to the greatest extent. Taking the *FASS* algorithm as an example and comparing Figure 13, it also has good detail processing ability. The mesh topology of the "protruding/tip" part of the mesh can well preserve the original geometry. The bunny's ears, chin, tail and other parts are very clear after simplification, as shown in Figure 13. In rendering, the performance is significantly improved.

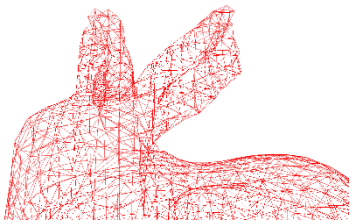

Original


VT


FAD (Included angle <= 30°)


FAD (Included angle <= 150°)


IVDD (Distance = 0.0011)


IVDD (Distance = 0.0013)

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 6 (2020), pp. 1110-1119
 https://dx.doi.org/10.37624/IJERT/13.6.2020.1110-1119

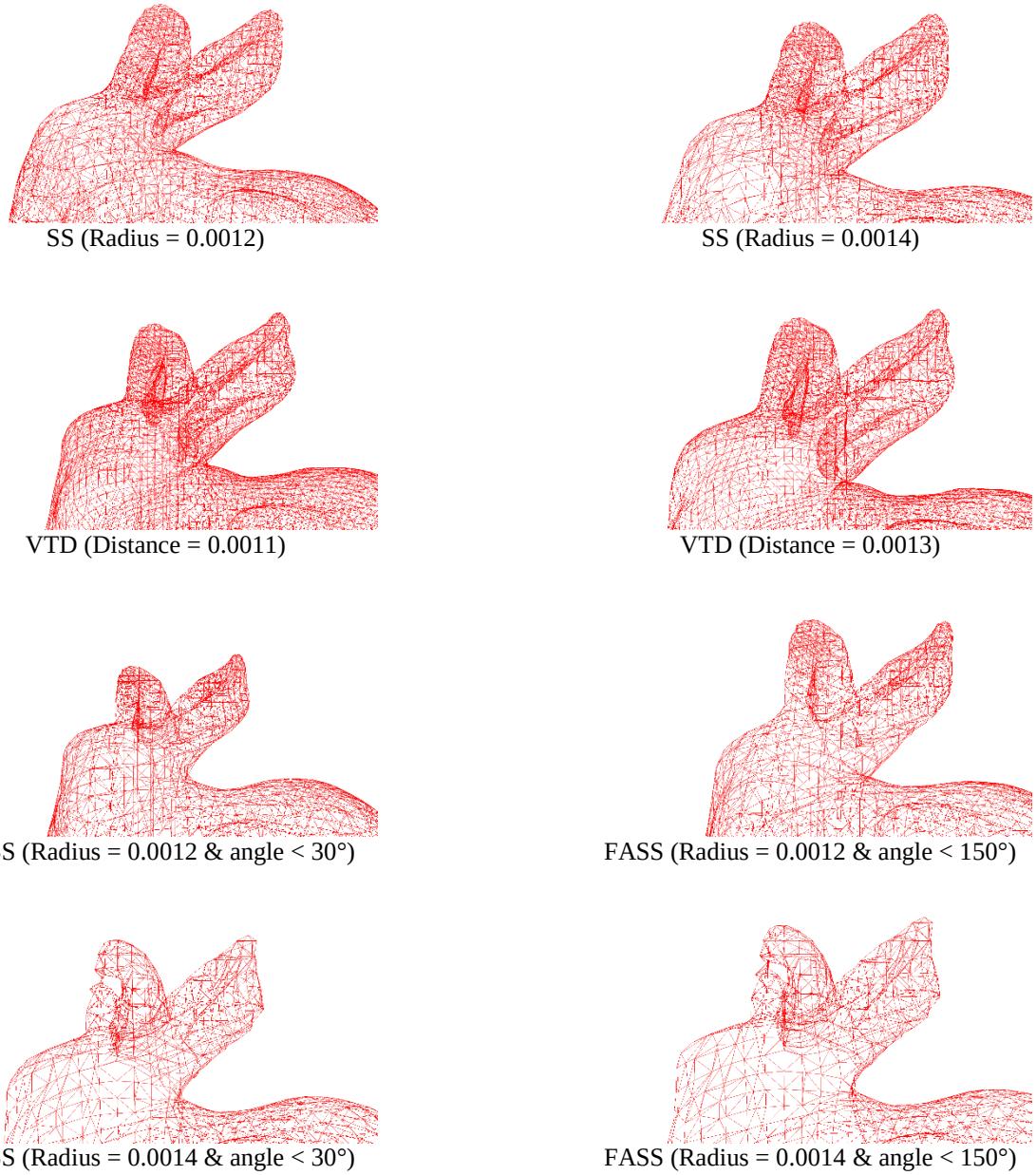


Fig. 13. Rendered details of bunny ears

For the rendering of Stanford's bunny models, from the above comparative data, the overall effect of the hybrid algorithm is close to that of a single algorithm. When rendering a bunny model, the hybrid algorithm still shows the oversimplification problem shown in Figure 13 only in the bunny's ears. In Figure 13, it is clear that the rendering level of the bunny ears is not very clear, but it is not difficult to compare Figure 13 and Figure 8 to find that the visual effect is improved compared with the simplification of a single algorithm. When using the *IVDD* algorithm to simplify, if the distance between the vertices is set to 0.0013, the simplified model also shows the problem of oversimplification in the bunny ears, which seems to be different from the original model, and this problem also has a certain impact on the *VTD* algorithm.

V. CONCLUSION

In this paper, this research described various algorithms for mesh simplification and to control the simplification process by setting different thresholds. From the results of a control group test on two computers with different configurations, it can be seen that in the *VT* algorithm, the average model rendering speed improved by 30.32%; in the *FAD* algorithm, the average rendering speed improved by 25.76%. Similarly, using the simplified model of the *IVDD* algorithm, rendering efficiency improved by 25.56%. Finally, in the *SS* algorithm, the simplified model rendering efficiency improved by an average of 23.97%. It can be seen that the improvement ratio of these four single algorithms on rendering efficiency is basically stable between 23% and 30%. In the last two hybrid algorithms,

rendering efficiency also improved by 22.71% and 23.38%. Although these two hybrid algorithms decrease in terms of rendering efficiency, the rendered results are smoother, and the details are preserved well. These several algorithms have met the original expectation that rendering efficiency will increase by nearly 20%, and the grid of the model is effectively simplified while the original geometry of the grid model is maximally maintained.

Since there is still room for improvement in these algorithms, subsequent research will make certain improvements to eliminate defects. Additionally, these algorithms will be applied to research on gaze point rendering technology of VR devices in subsequent research to expand the research scope.

ACKNOWLEDGEMENTS

This research was supported by NRF in Korea (NRF-2017R1A1A1A05069806). SeongKi Kim is the corresponding author.

REFERENCES

[1] Michael J. DeHaemer, Michael J. Zyda. Simplification of objects rendered by polygonal approximations. Computer Graphics. 1991; 25(2): 175-184.

[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Mesh optimization. in Procceedings of the SIGGRAPH' 93, 1993; 27(c):19-26.

[3] M. Garland, P. S. Heckbert. Surface Simplification Using quardric Error Metrics. in Procceedings of the SIGGRAPH' 97. 1997; PP(c):209-216.

[4] Z. G. Pan, X. H. Ma, J. Y. Shi. Overview of Multiple Level of Detail Creation. Journal of Image and Graphics. 1998; 23(c):104-106.

[5] L. B. Feng, D. Y. Luo. Research on 3D Model LOD Algorithm. Computer Technology and Development. 2010; 20(c):97-100.

[6] J. B. Pan. Adaptive subdivision research for triangular mesh. Computer Engineering and Applications. 2011; 47(c):186-187.

[7] R. Zhang. Research of Triangle Mesh Simplification Algorithm Based on Quadric Error Metrics. Xihua University Master Dedree Thesis. 2018.

[8] Y. Zhao, C Zhou, C Wang. Feature Preserved Mesh Simplification Algorithm Based on Stochastic Sampling. Computer Science. 2011; 38(c):249-251.

[9] J. Cohen, A. Varshney, D. Manocha, D. Manocha. Simplification Envelopes. in Procceedings of the SIGGRAPH '96. 1996; PP(c):119-128.

[10] W. Li, Y. F. Chen, Z. C. Wang, W. D. Zhao, L. Chen. An Improved Decimation of Triangle Meshes Based on Curvature. Rough Sets and Knowledge Technology. 2014; PP(c):260-271.

[11] J. Han, Q. J. Zhao, Z. G. Sun. Decimation of triangle meshes based on half-edge structure. Journal of System Simulation. 2006.

[12] C. Moenning, N. A. Dodgson. A new point cloud simplification algorithm. International conference on information, Imaging, and Image Processing(VIIP 2003). 2003.

[13] J. Rossignac, P. Borrel. Multi-resolution 3D approximations for rendering complex scenes. Geometric Modeling in Computer Graphics. 1993; PP(c):455-465.

[14] W. J. Schroeder, J. A. Zarge, W. E. Lorensen. Decimation of Triangle Meshes. ACM SIGGRAPH Computer Graphics. 1997; 26(2):65-70.

[15] S. J. Li, X. T. Jiang, H. Tang. High-quality simplified algorithm of texture model for detailed features preserving. in Application Research of Computers. 2018; 37(c).

[16] H. L. Li, S. K. Kim. Triangular Mesh Simplification based on Surface Angle. MITA2019. 2019; PP(c):143-146.

[17] H. L. Li, S. K. Kim. A Novel Mesh Simplification Method Based on Vertex Removal Using Surface Angle. IJERT2019. 2019; 12(c)1313-1320.