View-dependent Simplification for Web3D Triangular Mesh Based on Voxelization and Saliency

Wen Zhou, Jinyuan Jia, Shuang Liang
School of Software Engineering, Tongji University
Shanghai, China
zhouwen327@163.com  jyjia@tongji.edu.cn  shuangliang@tongji.edu.cn

Abstract—View-dependent simplification method can preserve the mesh contour feature, and be low computation, low time consuming in handling dense mesh. Besides, it totally realize simplification operation in webpage. In this paper, a framework for a simplification method based on saliency and viewpoint-based voxelization is presented. Moreover, by computing the related mesh principal curvatures, the edges that under some condition we met are simplified. Besides, we make use of voxelization for models, by computing the relationship between viewpoints and bounding box of voxelization, in order to judge which parts of the mesh is seen. Moreover, we use new data structures, the Orthogonal List and max heap, which are different from others. The Orthogonal List mainly stores the edge-point index of mesh, and the max heap does the simplification operation and sort, by computing the weight of surface, in order to ensure the complex triangle mesh always firstly being simplified. Finally, the proposed algorithms is implemented to test robustness and efficiency of our approach. The result shows that proposed methods are higher running efficiency than others.

Keywords: Web3D, View-Dependent Simplification, Saliency, Voxelization, Orthogonal List, Max Heap.

I. INTRODUCTION

For the mobile Internet, subject to restrictions on bandwidth and other conditions, it is unrealistic that showing a model on the webpage like a picture in real time. This bottleneck problem, is one of the three Web3D issues, and has hampered the development of web3D technology.

Many research scholars present their own methods. Tsang Ooi[1] etc. proposed the model loading strategy according to adaptive bandwidth based on LoD. WEN [2] et.al presented lightweight loading mechanism by computing repeatable component of model. Wang [3] etc. presented a simplification method of view-dependent, by computing the visible parts of model in view frustum.

In this paper, we incorporate view-based mesh voxel and saliency into the viewpoint-driven simplification scheme. We present three algorithms, with the first one, we make use of quaternion to compute the weight of edge of mesh, in order that the edge in complex surface can be always collapsed firstly. By this way, it will be better to preserve the structure of model. With the second method, we used voxelization of model, by computing the relationship between surrounding box of voxel and viewpoint, confirming which parts of mesh can be seen. Last but not least, we got mesh saliency by computing the principal curvature, then confirm which parts of mesh are in the contour of model.

The rest of the paper is organized as follows. In Section II , we briefly review the related work on simplification and voxelization & saliency, quaternion. In Section III, we show our proposed framework. In Section IV, we apply both viewpoint-based voxelization and saliency into simplification approach to select edges. Section V show the results of our experiments and analysis. Finally, in Section VI , we present our conclusions and future work.

II. RELATED WORK

In this section, we review the related research, including mesh simplification, voxelization, saliency, and quaternion.

A. Mesh Simplification

Mesh simplification can be classified into three methods, namely, vertex clustering algorithm, re-sampling algorithm, incremental algorithm. Kettner [4] etc. proposed the Mesh simplification algorithm based on half edge data structure of mesh model. Garland [5] proposed the incremental elimination algorithm based on quadric surface. Rossignac [6] et.al presented mesh vertex clustering algorithm that the three-dimensional space is divided into many small cubes, and each cube is called a cluster set. This method often had high efficiency and robustness, but the quality of the mesh is not high. Hoppe [7] etc. proposed the incremental method for mesh simplification, which is based on energy function.
optimization and embeds the idea of alternately smoothing then decimate. Incremental decimation algorithms used in most cases led to higher quality meshes, but suffered from large computational complexity overhead especially when a global error threshold is to be respected. The substantial improvement over classical decimation methods is the generation of progressive meshes (PM) during the decimation proposed by Hoppe [8], which is the first algorithm that employed the edge collapse operator. But it had difficulties distinguishing important shape features such as high-curvature regions. Bae [9] etc. presented a resampling method to compress triangulated surfaces, which is based on the PDE geometric diffusion equation where the finite element method is directly applied on the triangular mesh. Morigi [10] etc. presented a new approach to multilevel surface simplification based on the evolution of surfaces under p-Laplacian regularization.

B. Voxelization & Saliency

Voxelization is the process from a set of continuous geometric primitives to an array of voxels in the 3D discrete space that approximated the shape of the models as closely as possible. This concept is first introduced by Kaufman [11] et.al. Dachille [12] et.al proposed an efficient approach for incremental triangle voxelization. Zhao [13] et.al presented an efficient voxelization algorithm for complex polygonal models by exploiting newest GPU hardware.

Saliency research is firstly proposed by Lee [14] et.al. They incorporated saliency in graphics applications, such as mesh simplification and viewpoint selection. However, they used a center-surround operator on Gaussian-weighted mean curvatures. Feixas et.al [15] proposed an improved new approach that is the approach of viewpoint-driven mesh saliency. Recently, Casteloa [16] et.al made further efforts and proposed a new method, incorporating viewpoint-based mesh saliency into the viewpoint-driven simplification scheme, moreover, they made use of polygonal saliency to propose a new simplification error metric.

C. Quaternion

Quaternion had been widely applied in 3D computer graphics. It mainly played a great role in spatial rotation. When used to represent rotation, unit quaternions are also called rotation quaternions. The representation of a rotation as a quaternion (4 numbers) is more compact than the representation as an orthogonal matrix (9 numbers). Furthermore, for a given axis and angle, one can easily constructed the corresponding quaternion, and conversely, for a given quaternion one can easily read off the axis and the angle [17].

III PROPOSED FRAMEWORK

In this section, we will propose our framework, including our data structure, viewpoint-based voxelization and principal curvature based saliency. We incorporated these into mesh simplification.

Our framework (see fig. 1) mainly included server port and client port. Client port mainly sent the request with asynchronous method. Then, the server port received the request, and computed the related work, such as voxelization and mesh saliency. We use orthogonal list to save the index of edge and compute the weight of surface in order to push the edge into heap to finish the operation of simplification according the weight. Furthermore, we always firstly simplify when the weight of surface is bigger, in order to preferably preserve the contours.

A. Quaternion Notations

If Ω demotes the one of triangle mesh of model., then, \( \Omega = \begin{pmatrix} A x \ y \ z \ 1 \end{pmatrix} \). This equation expresses a general form of plane. Then, \( \begin{pmatrix} A \ B \ C \ D \end{pmatrix} \) expresses the normal vector of plane. Because we have known the value of the point of plane, that is, \( P_i(x_1,y_1,z_1) \), \( P_i(x_2,y_2,z_2) \), \( P_i(x_3,y_3,z_3) \). According to the Cramer’s Rule, we will compute the values of the parameters \( (A, B, C, D) \).

We define a 4*4 matrix, \( Q \), which is a quaternion, and every face and point in mesh had own Q matrix. It can be understand that the values of matrix are normal vector casted to the four-dimension space.

Therefore, the Q matrix of a triangle face will expressed.

\[
Q = \begin{bmatrix}
A \\
B \\
C \\
1
\end{bmatrix}
\]

Because a point belongs to many faces, the quaternion of point should be the sum of the quaternion of faces. \( Q_p = \sum_{i} Q_{i} \), then the parameter of N denotes the summary value of the face which the point belonged to. As is known to all, a surface consisted of many planes, therefore, we denote a surface, \( \Psi(x,y,z) \). Then, we use the matrix representation this equation. \( \Psi(x,y,z) = P^T Q P \). Then, \( P \) is the row vector \( (x, y, z, 1) \). As is known to all, we compute the value of \( P^T Q P \) to judge the matrix positivity. Therefore, we believe that the two point quaternion of an edge can decide the complex situation of their own surface.
IV Framework Description

A. Building data structure
In fig. 2, we briefly introduce the structure of the Orthogonal List, and in fig. 4, the max heap structure will be seen.

The mainly steps of our data structure as follows:
- New X axis, Y axis. Where X, Y axis on the node represented the edge information, the X axis node contains two pointers, down, up, respectively, which pointed to the less and greater than its value. Likewise, the Y axis also contained two pointers, left, right, respectively, which pointed to the smaller and greater than its value, the edge of the first vertex index. The storage size of two axes is the count of mesh points.
- Traverse the entire mesh, in accordance with the relationship, then establish an orderly point of the Orthogonal List structure.
- Build the quaternion of point and face of mesh.
- Along the X axis of the Orthogonal List, the edge of mesh is pushed into max heap, and the heap also established a pointer, the pointer to the edges in the index table position. At the same time, the max heap will be adjusted according to the weight value.

B. Mesh Simplification
Our data structure includes Orthogonal List, max heap. The Orthogonal List will save the edges of mesh. In last section, and we have introduced the computing process of weight of edge, we always adjust and Pop the element of heap top according the weight value.

C. Mesh Saliency
The principal curvature can better measured the steep degree of the surface. When the principal curvature is larger, the surface will be greatly changed.
Fig. 4. The process of simplification (we can compute the weight of every edge)

From fig. 5, it is not hard to find that the value of the normal vector along the direction of tangent vector, will be changed, then, the maximum and the minimum value are the principal curvature of point \( p \). In \( P \), we defined Jacobian matrix as follows:

\[
J = \begin{bmatrix}
\frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} \\
\frac{\partial g}{\partial x} & \frac{\partial g}{\partial y}
\end{bmatrix}
\]

Likewise, we defined the cosine angle of two vector as follows:

\[
\text{Cosine } (\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{||\overrightarrow{a}|| \cdot ||\overrightarrow{b}||}
\]

**Definition 1:** Inner product operation,

\[
\overrightarrow{a} \cdot \overrightarrow{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3, \quad I \text{ is called inner product operation.}
\]

Then \( ||\overrightarrow{a}||^2 = \overrightarrow{a} \cdot \overrightarrow{a} \) denoted the square of the length of the tangent vector.

**Definition 2:** Outer product operation:

\[
\overrightarrow{a} \times \overrightarrow{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}, \quad \text{\( \otimes \) is called outer product operation.}
\]

According to above defines, we can compute the curvature of a normal vector \( \overrightarrow{n} \), along the direction of tangent vector \( \overrightarrow{t} \):

\[
\kappa_n(t) = \frac{\overrightarrow{n} \cdot \overrightarrow{t}}{||\overrightarrow{t}||^2}
\]

According formulas 1, in order to obtain the extreme value, we did the differential operation, then got the formulas 2.

\[
\frac{d\kappa_n(t)}{dt} = 0
\]

We computed the result of formulas 2, then took the related value into formulas 1, finally, we can compute the two values \( (k_1, k_2) \). Surely, in some condition, \( k_1 = k_2 \).

- **HSV.** We can compute the curvatures of all points. Obviously, each principal curvature had two directions, which we called \( pdir1, pdir2 \), then the mean and Gaussian curvature the \( k^{th} \) point of mesh will be represented.

Next, we will be converted to the value of HSV. As is known to all, The V of HSV color table represented value. Therefore, we set that the value of V equaled one.

\[
r_k = a \times \tan^{-1} \left( \frac{M_1 + M_2 - M_k}{M_k + M_1 + M_2 - 2M_k} \right)
\]

Where the parameter is an experimental value, we set \( a = 0.75 \).

- **Edge Selection.** According to the color table, we acquire the related edges with high curvature.

**D. Voxelization**

From the fig. 6, we will acquire the seen surface by computing the relationship between surrounding box and viewpoint. Voxelization of mesh will generate many
boxes, which is surrounded by many the triangle mesh of model. Each box includes at least one triangle mesh.

We don’t use new method to create voxel of model. As is known to all, it is an efficient way to represent three-dimensional objects by octree. Therefore, we adopt this traditional method to generate voxelization.

Without loss of generality, we suppose the viewpoint, \( \mathbf{v} \). The center vector of the \( k \)th and \( j \)th bounding box, \( \mathbf{v}_k \) and \( \mathbf{v}_j \). We define three kinds of relationship among \( \mathbf{v}_k, \mathbf{v}_j \) and \( \mathbf{v} \). Box A and Box B are two bounding box of voxelization of model, whose center vectors are \( \mathbf{v}_k \) and \( \mathbf{v}_j \) respectively.

- **Relationship 1:** A covers B.
- **Relationship 2:** A is covered B
- **Relationship 3:** A and B don’t affect each other.

In the relation of one and two, \( K \) and \( J \) affect each other.

\[
\alpha_k = (\mathbf{v} - \mathbf{v}_k) \cdot (\mathbf{v} - \mathbf{v}_k) \quad (5)
\]

\[
\beta = (\mathbf{v}_j - \mathbf{v}_k) \cdot (\mathbf{v}_j - \mathbf{v}_j) \quad (6)
\]

Then \( \mathbf{v} \) is the vector of viewpoint.

**E. Time complexity analysis**

In this section, we will discuss time complexity of our algorithm.

Firstly, our algorithms mainly adopt the orthogonal List and max heap data structure. The time complexity of the construct of orthogonal list is \( O(t^*m) \), where \( t \) denotes the count of edge, \( m \) represents the count of point. In fact, the time complexity of the Orthogonal List will descended, along with the mesh simplification. As is known to all, orthogonal list usually is used to represent sparse matrix. The simplified mesh is just a sparse matrix. Therefore, orthogonal list will be better than others’ data structure, such as half-edge structure. Because of this, it will obviously excel in time consuming facet. Besides, it’s believed that Heap sort is better than the others’ sort algorithms, such as bubble sort. The time complexity of heap sort is \( O(n\log n) \). Therefore, our algorithms have a great advantage in time consuming facet.

In this section, we present the results of our experiments with several models simplified using voxelization and saliency.

**A. Environment**

The method presents in this paper has been implemented using C++ program language, and is executed on PC under Windows 7 OS, Intel core I5-M580 processor, 4G memory size. In this section, all results are got in this machine. Moreover, it has been applied to several models successfully.

**B. Evaluation criteria**

Typical criteria to judge the results of the simplification method are based on Hausdorff distance. We compute the above three hausdorff distance (see fig. 7), then we find that our method didn’t have obvious advantage compared with others. We analyze it then got the reason is that ours maybe enlarged hausforff distance, because our method often simplify the local region of model. However, there is a great advantage in time elapse with our method.

**C. Results**

We do some experiment on girl models (see fig. 8). It’s obvious that our method is better in time elapse. In fig. 9, our methods can send request in asynchronous mode, then the model is simplify in server port, then send the simplified model to client port. In this way, at first, the browser tool can display the simplified model in progressive mode, load the original model in last. It is so-called LoD.

Next, we will test our saliency and voxelization algorithms.

In fig. 10, we select 6 different viewpoints to test our saliency algorithm. It is not hard to find, the red part is our need part. Generally Speaking, it is the contour of model in current viewpoint. Therefore, saliency is very important, which can better preserve the feature of model in current view point.

The table1 shows the result of our algorithm computing the seen surfaces. The smaller the voxel size is, the better the result became, and however, it will add the time consuming.

![Fig. 7. Two different methods compared figures in max, mean, RMS criteria.](image)
Fig. 8. The compared experiment of girl model

Fig. 9. Our method applied in LoD display

Fig. 10. The saliency of model

### TABLE I. Viewpoint-based voxelization acquired points set

<table>
<thead>
<tr>
<th>Model</th>
<th>Viewpoint</th>
<th>Original nPoints</th>
<th>Acquired nPoints</th>
<th>Fov</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>(0, 0, 4)</td>
<td>2503</td>
<td>2449</td>
<td>45</td>
</tr>
<tr>
<td>Dragon</td>
<td>(0, 0, 4)</td>
<td>50000</td>
<td>26954</td>
<td>45</td>
</tr>
</tbody>
</table>

### VI. CONCLUSIONS AND FUTURE WORK

We have represented an efficient framework of reducing complexity of triangle mesh, using saliency and view-based voxel. Besides, we adopted new data structure, the Orthogonal List, max heap, which can lower the time complexity of simplify algorithms. Above all, in web browser-based application, it is very important to lower the complexity.

### ACKNOWLEDGMENT

This research work is supported by the National Science Foundation of China (No.61272276, No. 61305091) and The Fundamental Research Funds for the Central Universities (No.2100219054).

### REFERENCES


