An adaptive hierarchical transmission mechanism of 3D models for mobile terminals

Chen Wei and Yanlei Shang*
Institute of network technology, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: shangyl@bupt.edu.cn

Abstract. An adaptive hierarchical transmission mechanism of 3D mesh models for mobile terminals based on WebAR technology is introduced. By proposing a vertex merging algorithm based on vertex merging degree, the original model is decomposed into multiple groups of data consisting of a simplified model and a layer file for reconstruction according to the specific compression ratio. The experiments show that the initialization time of the AR scene using the simplified model as the base model is effectively reduced and the simplified model can be reconstructed to the original model losslessly with the layer file. The proposed mechanism can be used for adaptive transmission, lossless reconstruction and real-time rendering on mobile terminals.

1. Introduction

With the improvement of computing power of electronic products, AR technology is applied to more and more scenes. However, the mainstream native AR technology still has certain drawbacks: Android and iOS dual platform development costs, and the download cost of the user App is high, which makes the promotion difficult. WebAR technology based on intelligent mobile terminals has been rapidly developed because it realizes the cross-platform, easy-to-propagation and low-cost characteristics of AR, which makes the browsers become a popular carrier of AR technology.

In 2017, Apple and Google released ARKit and ARCore to give space recognition capabilities for smart mobile terminals, enabling mobile terminal users to experience the powerful ability of WebCore and Webkit to interact with the environment with only a web page. WebAR solves many pain points in the promotion of AR technology, but there are still problems: WebAR depends on virtual objects, while virtual objects are 3D mesh models, and file size is related to the fineness of the model. When fineness of the model is increased, the vertices and patches of the model are increased and the size of file is rapidly increased, which makes the amount of downloading and calculation required for rendering the WebAR scene large. However, the limited computing power of single-threaded JavaScript and the poor computing power of the mobile terminal devices lead to a large difference in loading time of WebAR scene for mobile terminal devices in different conditions. For mobile terminals whose performances are not high, when they are in poor 4G condition, there are cases where the loading time is long and the rendering frame rate is low. In view of the above problems, we propose a solution: adaptive hierarchical transmission mechanism of 3D models for mobile terminal. This mechanism can effectively reduce the delay of loading and rendering WebAR scene to improve user experience.
2. Related work

The key to the adaptive hierarchical transmission of 3D models is how to layer and compress the 3D models on the server side. There have been many related research results on the compression of 3D models. In general, the 3D models involved in WebAR are mostly mesh models, which is mainly discussed in this paper. The compression methods are divided into single resolution compression [1] and progressive compression [2, 3]. Single resolution compression can only start model rendering after all model file data have been downloaded, so this method is not suitable for solving the problem of slow initialization. Progressive compression provides a more flexible loading method: when initializing the WebAR scene, quickly transmit and render simple models, and then download additional information to improve the fineness of models. Since the resolution of mobile terminals is not high and web viewport is small, this multi-resolution [4] progressive mesh [5] can quickly construct the contour of the model and the fineness of model can be improved as needed, so that the time required for initialization is greatly reduced, which makes it more suitable for rendering 3D models on mobile terminal browsers.

The progressive mesh uses continuous edge collapse operations to eventually reduce an original model $M$ to a base mesh model $M_0$. The result of an edge collapse operation is shown in Figure 1.

![Edge Collapse Operation Diagram](image)

**Figure 1.** Edge collapse operation diagram.

For the selection criteria of the collapsed edge, the most representative method is MongKolnam’s compression method [2], which uses the quadratic error measurement method [6] to simplify the original model. Then the simplified mesh model uses the subdivision method [7-9] to fit the original model and obtains the semi-regular mesh, and the inverse transformation can generate the progressive mesh. However, to achieve the inverse transformations, the Gauss-Seidel method is needed to solve complex linear equations. A large number of complex calculations are not suitable for mobile terminals. Aiming at the progressive transmission of mesh in wireless network environment, [10] proposes a progressive mesh generation algorithm based on inverse butterfly mode, which solves the problem of network transmission and display of 3D models on mobile terminals to some extent, but the number of vertices involved in the outer ring of butterfly mode is large, that affects the speed of the model recovery. In [11], a method for 3D model compression on mobile terminals is proposed. Combining the advantages of loop mode and interpolation butterfly mode, the revised loop mode is defined. Based on reverse revised loop mode, a 3D mesh model is reduced to a sparse base mesh, meanwhile, a series of offset sequences are generated to reconstruct the model. But for higher compression ratio, the embedded zerotree wavelet coding compression is applied to the offset sequences, and the lossless reconstruction of the original 3D mesh model cannot be performed under the condition of high compression ratio.
3. System architecture

![Adaptive hierarchical transmission system of 3D models for mobile terminals architecture diagram.](image)

The adaptive hierarchical transmission system of 3D models for mobile terminals based on the proposed mechanism aims to do three things mainly: Firstly, progressively layer and compress the 3D model on server side. Secondly, adaptively transmit and load the mesh model of different levels for different mobile terminals in different conditions. Thirdly, realize the lossless recovery and reconstruction of the model. The system architecture is shown in Figure 2.

3.1. Design concept

The edge-based collapse needs to find a suitable vertex to replace two vertices involved in the collapsed edge and it costs a lot. To reduce the amount of calculation of lossless recovery, use point-based point merging operation: find suitable vertex to be merged \( v_0 \) and merging target vertex \( v' \). For how to find suitable vertices to be merged, we propose a standard based on normal vector dot product, which is called vertex merging degree: the higher the degree of coplanarity of the triangular patch where the selected vertex is located and its adjacent patches, the smaller the influence of overall model structure after the selected vertex is merged. If selected vertices are not suitable, the simplified model will produce a large deformation. To ensure that the model deformation is minimized after merging, the average normal vector dot product sum criterion is used to find set of the most suitable vertices to be merged.

To simplify the work of the recovering model, batch of point merge operation is used: one model simplification step integrates several point merging operations to merge multiple vertices one time. The merging operation of level \( i \) will merge \( i \times 10\% \) vertices of the original model \( M \) to generate a simplified model \( M_i \). The higher the compression level \( i \), the higher the compression ratio of the model and the less the number of vertices and patches of the simplified model. In the process of compression, the merged vertices and the deleted topology are recorded to form a layer file \( d_i \) for recovering \( M_i \) to \( M \). That is, for \( M \), n different levels of batch vertices merging operation are respectively performed to generate multi-level data \( \{ M_i, d_i \} \). When initializing WebAR scene, different levels of model will be transmitted and loaded according to the different computing powers and network environments of mobile terminals.

With above design concept, to realize the adaptive transmission and lossless recovery, three main modules are required. Vertices calculating module, responsible for calculating and finding the optimal set of vertices to be merged for model compression. The layering and compression module is responsible for merging all vertices in the optimal set calculated by vertices calculating module, recalculating the topological relationship of the simplified model and generating a base mesh and layer file. The recovery and reconstruction module is responsible for the recovery of the model, reconstructing the model \( M_i \) to original model \( M \) losslessly according to the downloaded layer file \( D_i \).
3.2. Module implementation

3.2.1. Vertices calculating module. For the vertex merging degree proposed in above design concept, the calculating Formula 1 and 2 are as follows:

\[ f(v) = \sum_{i=0}^{N_{v}} g(v, N_{v_i}) \]

\[ g(v, u) = \sum_{i \in T_{vu}} n_{p} \cdot n_{l} \]

\[ N_{v} \] is a set of neighbor vertices of vertex \( v \), \( |N_{v}| \) is the number of elements in \( N_{v} \). \( N_{v_i} \) is the \( i \)-th element in \( N_{v} \). \( T_{vu} \) is a set of triangular patches using vertex \( v \) and \( u \) as two vertices, \( T_{v} \) is a set of triangular patches using vertex \( v \) as a vertex and \( N_{f} \) is the positive normal vector of triangular patch \( f \).

Based on the above formula, for the model compression, a vertex index set \( S \) is calculated, whose elements represent the index position of the vertex that needs to be merged in the vertex array data of the model. The general representation and storage form of vertices and patches of 3D mesh model is introduced in [12]. By calculating the vertex merging degree of all vertices and sorting them, a set of optimal vertex index set \( S \) can be found so that the simplified model with the least distortion can be obtained after merging all the vertices in \( S \).

To facilitate calculations, define vertex and triangle structures to store information about vertices and triangular patches of the model. In the structure of vertex, information about the coordinates, index value, merging degree, index of the merging target vertex, array of the triangular patches using this vertex as its vertex, array of adjacent vertices of this vertex are stored. Meanwhile, in the structure of triangle, the index of three vertices of this patch, the normal vector and the level of this patch should be stored.

According to formula and preprocessed data of the 3D mesh model, the pseudocode of the process of calculating vertices to be merged at level \( i \) is as follow:

\[ Set \text{calVerticesToBeMerged}(V) \{ \]
\[ \text{vs} = \text{initVSet}(V); \quad // \text{generate the set of vertex} \]
\[ \text{calcuMergingDegree(vs);} \quad // \text{calculate the merging degree of vertices} \]
\[ \text{sortSet(vs);} \quad // \text{sort the set by the merging degree value} \]
\[ \text{for}(k=0; k<i*0.1*V.length;i++)\}\]
\[ v = \text{vs}[k]; \]
\[ \text{while}(\text{vs[v.mergeTarget].isMerged}) \{
\]
\[ v.\text{mergeTarget} = \text{vs[v.mergeTarget].mergeTarget}; \quad \}
\[ // \text{if the target vertex has been merged, update it} \]
\[ \text{v.isMerged} = \text{true}; \]
\[ \text{S.push(v);} \quad // \text{push v into optimal set S} \]
\[ \text{return S; \}
\]

3.2.2. Layering and compression module. In order to compress the dense mesh model \( M \) into a sparse mesh model \( M_i \) and a layer file \( D_i \) for recovery and reconstruction, two steps are required:

Firstly, vertices merging. According to whether the vertex index is in \( S \) calculated by vertices calculating module, the set \( V_i \) of all the vertices of the mesh model is divided into the reserved vertices set \( V_{r} \) and the merged vertices set \( V_{m} \), and the set \( F_i \) of all the triangular patches of the mesh model is divided into the reserved patches set \( F_{r} \) and the deleted patches set \( F_{d} \). That means, for \( M = (M_i, D_i) \), \( M_i = (V_{r}, F_{r}) \), \( D_i = (V_{m}, F_{d}) \). The vertex merging algorithm is described as: for each vertex \( V \) represented by the index in set \( S \), all the triangular patches in set \( F_i \) containing \( V \) as its vertex are
classified into the set \( F_d \). Then, correct their vertex \( V \) to the merging target vertex \( v' \) of \( V \) and delete the neighbor relationship of \( V \). After correction, if the patch can form a new triangle \( F' \), classified it into \( F_s \), otherwise, discard it. According to the table of deleted patches generated by \( F_d \) and the table of merged vertices generated by \( V_m \), layer file \( D_i \) is generated. The pseudo code of the level \( i \) layering and compression algorithm is:

```java
Void MergeVertices(S) {
    for (each index of S) {
        for (each face of \( V_{index}[S_{index}].faces \)) {
            \( F_d \).push(deepClone(updateFace(face)));
            calcuNormal(face); // update normal vector
            face.level = i;  // record level of face
        }
        for (each vertex of \( V_{index}[S_{index}].neighbors \)) {
            removeNeighbor(vertex, \( V_{index}[S_{index}] \));
        }
        \( V_{index}[S_{index}] \).isMerged = true;
    }
}
```

Secondly, topology reconstruction. Based on the reserved vertices set and the reserved patches set, the new topological relationship is obtained by re-triangulation. And the vertex table and the patch table of the new simplified mesh model are formed to describe a simplified mesh model \( M_i \) which maintains the contour feature of original model \( M \). Since some of the triangular patches in \( F_s \) are new generated patches after vertex correction, so they will be discarded during the process of recovery. To reduce the recovery work, a new column of attribute is added to the patch table, which represents the level of the patch. When there is no data in the column, it means that the patch is not corrected, so keep the patch during the process of recovery. Otherwise, it means the patch has been corrected, so discard it directly.

By repeating the operation of the two modules, the original mesh model \( M \) can be compressed into multiple groups of \( \{M_i, D_i\} \), \( i \) is the level of compression.

3.2.3. Recovery and reconstruction module. The recovery and reconstruction of the mesh model is the inverse process of layering and compression. Firstly, confirm the level \( i \) according to the downloading rate and rendering efficiency of the mobile terminal. Then, the base model \( M_i \) will be transmitted and rendered. When initialization of the WebAR scene is completed, the layer file \( D_i \) will be downloaded and the recovery work will start to reconstructed \( M_i \) to the original mesh model \( M \) losslessly.

The idea of recovery and reconstruction of the mesh model is shown in Figure 3. The black patches are the new generated patches with their vertices corrected, and they will be discarded directly during the process of recovery. The patches with the star mark are the patches deleted during the process of layering and compression, they need to be re-generated during the process of recovery, so the data of them need to be recorded in the layer file. The white patches are the reserved patches which will be retained when reconstructing. During reconstruction, the triangular patches of current mesh model \( M_i \) whose level number is not 0 will be firstly deleted. Then, according to each index, insert all vertices of the vertices set \( V_i \) in the downloaded layer file \( D_i \) into the array of the vertices of \( M_i \). Then, combine the reserved patches with all the patches in the patch table stored in \( F_i \) in \( D_i \), the new topology will be generated. After re-triangulation, \( M_i \) will be recovered and reconstructed to \( M \) losslessly.

![Figure 3. Mesh model recovery and reconstruction idea diagram.](image)
4. Experimental results

The models of flamingo, horse and sculpture are tested for compression. The rendering results of several models and the simplified models of level 5 generated after layering and compression are shown in Figure 4. Table 1 records the relevant information of several simplified models of level 1, 3 and 5: the number of vertices and patches and the size of the simplified model files can be compared.

![Figure 4](image)

**Figure 4.** Original (a)-(c) and simplified (d)-(f) mesh models diagram.

Use a mobile terminal (CPU: 2.8GHz GPU: Qualcomm Adreno630 RAM:6GB) to test in the case of a complex 3D character mesh model (60MB) is divided into five groups data of base model and layer file and the mobile terminal is in the mobile 4G network environment, the initialization time of the WebAR scene, the adaptive transmission time of layer file, the recovery and reconstruction time and the average frame rate during whole process. The relevant test result data are shown in Table 2.

**Table 1.** The relevant information about original models and simplified models of level 5.

| Level | Bird V/F  | Horse V/F  | Sculpture V/F | Size (KB)  |
|-------|-----------|------------|---------------|------------|
| Original | 273/542  | 494/984  | 10000/19482  | 32/56/1578 |
| 1      | 246/488  | 445/886  | 9000/17690  | 28/50/1408 |
| 3      | 192/380  | 347/690  | 7000/13726  | 22/39/1095 |
| 5      | 136/270  | 247/490  | 5000/9755   | 15/27/774  |

**Table 2.** The test results of adaptively transmission and losslessly recovery of complex model.

| Level | Current size (MB) | Initialization (ms) | Transmission (ms) | Reconstruction (ms) | Frame Rate (fps) |
|-------|-------------------|---------------------|------------------|--------------------|------------------|
| Original | 60                | 58328               | -                | -                  | 30               |
| 1      | 53                | 54173               | 9364             | 1800               | 37               |
| 2      | 47                | 48387               | 18613            | 2500               | 47               |
| 3      | 40                | 39752               | 26842            | 3100               | 54               |
| 4      | 32                | 33125               | 34951            | 3900               | 60               |
| 5      | 28                | 28870               | 38052            | 4400               | 60               |

5. Conclusion

Adaptive hierarchical transmission of 3D model for mobile terminals mechanism proposed in this paper can effectively reduce the transmission amount of data required for the initialization of the AR scene on the mobile terminal by 50% to 60% and improve the rendering speed of the 3D mesh model by layering and compressing the original 3D mesh model into a simplified mesh model with the number of vertices and patches about half of the original mesh model and a layer file for reconstruction. Through the recovery and reconstruction method, combined with the layer file, the on-demand progressive reconstruction of models can be realized for mobile terminals with different performances, and finally realize the lossless reconstruction. Although the total time of transmission of the simplified model, downloading of layer file and the recovery of the model is higher than the delay of one-time transmission and loading of the original model, the cost is acceptable for reducing the delay of initialization of AR scene. With the development of WebAR technology, more real-time interactive applications on the mobile terminal will use web as main carrier, which makes adaptive hierarchical transmission mechanism of 3D models for mobile terminals have a good prospect.
6. Acknowledgments

The work presented in this study is supported by the National Natural Science Foundation of China “Research on Web-based Dynamic Service Delivery Mechanism and Method in Information Center Network” (61671081) and the Natural Science Foundation of Beijing “Research on Dynamic Service Provision Mechanism of Information Center Network Based on Edge Computing” (4172042).

References
[1] Bajaj C, Pascucci V, Zhuang G, Single resolution compression of arbitrary triangular meshes with properties, Storer JA, Cohn M, eds. Proc. of the Data Compression Conf, 1999.
[2] Khodakovsky A, Schroder P, Sweldens W, Progressive geometry compression, Akeley K, ed. Proc. of the ACM SIGGRAPH, 2000, New York: ACM Press, 2000, 271-278.
[3] Mongkolnam P, Razdam A, Farin G, Lossy 3D mesh compression using loop scheme, Hamza MH, ed. Proc. of the Computers Graphics and Imaging, 2003.
[4] Lounsbery M, Derose T, Warren J, Multiresolution analysis for surfaces of arbitrary topological type, ACM Trans, On Graphics, 1997, 16(1) 34-73.
[5] Hoppe H, Progressive meshes, Rushmeier H, ed. Proc. of the ACM SIGGRAPH, 1996.
[6] Garland M, Heckbert P, Surface simplification using quadric error metrics, Turner W, ed. Proc. of the ACM SIGGRAPH, 1997, Los Angeles: ACM Press, 1997, 209-216.
[7] Dyn N, Levin D, Gregory JA, A butterfly subdivision scheme for surface interpolatory with tension control, ACM Trans. On Graphics, 1990, 9(1) 160-169.
[8] Loop C. Smooth subdivision surface based on triangles, Salt Lake City: University of Utah, 1987.
[9] Lee A, Moreton H, Hoppe H. Displaced subdivision surfaces, ACM SIGGRAPH 2000.
[10] Luo XN, Progressive meshes transmission over a wired-to-wireless network, WINET, 2006.
[11] Ma JP, Luo XN, Chen Bo, Triangle Mesh Compression Based on Reverse Subdivision for Mobile Terminals, Journal of Software, 2009, 9(20) 2609-2610.
[12] Li HS, Liu Cheng, Research on Mesh Data Compression of 3D model, Journal of System Simulation, 2013, 9(25) 2150-2152.