IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 13, 2021, accepted January 21, 2021, date of publication January 27, 2021, date of current version April 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055016

Real-Time High Realistic Web Display Method of Complex 3D Model

CHUAN XIE[1], ZHIZHONG WANG[2], HAIBO CHEN[2], XIAOLONG MA[3], WEI XING[2], LEI ZHAO[2], AND ZHIJIE LIN[4]
[1]Hangzhou Vocational and Technical College, Hangzhou 310003, China
[2]Department of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[3]School of Management, Huzhou University, Huzhou 313000, China
[4]School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China

Corresponding authors: Xiaolong Ma (xiaolongma@zjhu.edu.cn), Wei Xing (wxing@zju.edu.cn), Lei Zhao (cszhl@zju.edu.cn), and Zhijie Lin (bytelin@qq.com)

This work was supported in part by the Construction and Verification of Digital Identity System for Folk Cultural Relics under Grant 2020YFC1523201, in part by the Research on Key Methods of Intelligent Extraction, Understanding and Generation of Elements in Large Ruins under Grant 2020YFC1523101, in part by the Collaborative Processing and Intelligent Computing Engine of Digital Cultural Heritage under Grant 2020YFC1522701, in part by the Research on Key Technology of Image Inpainting Based on Background Knowledge Constraint under Grant LY21F020005, in part by the Management Oriented Network Mechanism Design and Data Mining Method under Grant LGF19F020003, and in part by the Research on Key Techniques of Image Inpainting Based on Weak Supervised Learning under Grant LY19F020049.

ABSTRACT With the in-depth application of 3D models in many fields, the display of models, especially the model display based on B / S architecture, has become an indispensable part of the application of 3D models. The development of 3D modeling technology has led to the emergence of a large number of complex three-dimensional models with more than one hundred thousand facets and hundred megabytes of file size. Coupled with the limit of B / S architecture by the server's concurrency, network bandwidth, browser's processing capabilities, and many other objective factors, how to achieve its real-time and high-realistic display on the web pages has become one of the hot spots research in the field of graphics, which has great research significance and challenge. In order to resolve the above problems, we propose a new model display method. The method pre-renders the model into a two-dimensional image by sampling the subdivide surface of the model's circumscribed sphere and uses the view-dependent level-of-detail method to divide the pre-rendered image into different layers and tiles. Combined with the optimization, including adaptive and progressive network transmission, index-weighted user interaction prediction, and Hilbert-based image indexing, the method achieves a frame rate of more than 20 frames per second in the actual tests.

INDEX TERMS Model Pre-rendering, real-time display, high realistic rendering, Internet 3D display, B/S architecture.

I. INTRODUCTION

With the development of acquisition equipment and computer modeling software, the acquisition of 3D models is more and more convenient, and its use is more and more extensive. Three-dimensional models have been widely used in engineering and academic fields. The medical industry uses them to construct elaborate organ models; the film and game industries use them as virtual scenes or objects and characters in the scene; the construction industry uses them to display the proposed buildings and landscapes to replace the traditional physical building models; the engineering industry uses them to design new equipment, vehicles and other structures; the scientific community uses them to construct complex chemistry Molecular structure model of the complex. In addition, the 3D model is also the basis of 3D printing. In these applications, the 3D model display is one of the key components and application direction. The speed of display and the high degree of realistic display not only affect the direct sensory experience of users but also affect the feasibility of 3D models in practical engineering practice.

However, with the development and progress of acquisition and modeling technology, the precision and complexity of the 3D model are getting higher and higher. For example, for some high fidelity 3D models, the number of vertices and patches can reach several million. The size of the model file has also reached the size of GB. In the process of 3D model display based on B / S architecture, model files need to be transmitted through the network first and then rendered by WebGL through a browser. For such a complex and fine

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-hoon Kim.

3D model, it will inevitably encounter real-time and high realistic problems in the process of display based on B/S (browser/server) architecture.

According to the interactive characteristics of 3D models, 3D models can be roughly divided into two categories. The interactive feature of a kind of 3D model is that it can browse 360 degrees around it outside and observe its texture and material details closely, such as artifacts and other cultural relics. The interactive feature of another kind of 3D model is that the whole model needs to be browsed in its interior. It focuses on the internal details rather than the external structure, such as the scene cultural relic model such as caves. The difference of interaction mode also causes the difference between them. In general, the rendering viewpoint of the latter model does not need to change and only needs to zoom in and out at a fixed position. This feature makes it possible to choose the panoramic image-based rendering method for this kind of model. The whole model is pre-rendered into a cube or spherical panorama format so that the image can be directly transmitted through the network and rendered directly on the client. However, the rendering viewpoint of the former model, which is also the research object in this paper, cannot be fixed. Because we want to be able to browse around its outer surface from various angles, its viewpoint generally needs to change according to the observation angle. In this way, we cannot directly apply the panorama based scene model browsing mode to such model browsing. This paper mainly studies the first kind of 3D model.

Because the complexity is related to the number of triangles and the size of model files (including geometry files and texture files), the complex model defined in this paper is more than 10 million triangles, and the model file size is larger than 1GB. Real-time definition is to show the whole model to users in the shortest possible time delay and to ensure a certain degree of fluency in the whole interaction process. Due to the real-time and fluency will affect the rendering frame rate, considering the user's actual subjective experience, this paper defines above 20fps as meeting the requirements of real-time. However, the traditional grid-based rendering method is difficult to achieve real-time smooth rendering through the network.

1) There are limitations and bottlenecks in network bandwidth. The traditional grid-based rendering method needs to transfer the complete model file before rendering. Obviously, there is still a big gap between this level and the size of the complex 3D model file, which makes the transfer of the model file need to consume a long time. We assume that the model file is 100MB, and the web page download speed is 2Mb / s. It takes 50s to load. If the model file is larger or the server bandwidth and network speed are slightly smaller, the load time will be longer. Such a long time of network transmission caused by the waiting will make the user experience very poor, thus reducing the user's interest in browsing 3D models.

2) There are limitations and bottlenecks in computing power. Even if the network transmission delay is not considered, according to the traditional rendering method of the 3D model in C/S (client/server) architecture, for a complex model, the browser generally needs tens of seconds or more than ten minutes to render the whole model. And in the rendering process will occupy a lot of memory and CPU resources, making the browser become stuck or even unresponsive.

Although there has been a lot of academic research on this problem, many scholars have proposed a series of solutions, such as image-based rendering. As a new rendering method, image-based rendering (IBR, please refer to II-B for more details) has been proposed for more than ten years. But for more than ten years, IBR has been staying in the field of theoretical research, and the practical engineering application is very few. In addition, the IBR method still needs to solve the large amount of bandwidth traffic and the huge network delay caused by the continuous transmission of high-definition images. So up to now, 3D model real-time display based on B/S architecture is still a pain point and difficulty, and needs to be solved urgently. The contributions of this paper are as follows:

1) This paper proposes a real-time display method based on model pre-rendering. By transforming the 3D model into the 2D images and layered and partitioned based on LOD, it can quickly display complex 3D models on the web.

2) Aiming at the huge bandwidth load of image transmission, an adaptive and progressive transmission method is proposed to transmit the image to the client efficiently.

3) Aiming at the complex and fast user interaction, the interactive prediction is proposed to preload the predicted viewpoint image in advance.

4) In view of how to quickly request the specified image from the server, this paper proposes to index the image in the server by means of Hilbert curve coding and puts forward a new implementation of the Hilbert curve coding algorithm.

II. RELATED WORK

According to the different data types used in client rendering, the rendering methods are divided into mesh-based rendering methods and image-based rendering methods.

A. MESH-BASED RENDERING

Traditional mesh-based rendering methods need to transfer the whole model file before rendering. The size of the model file studied in this paper can reach the order of GB. Obviously, such a large model will exert great pressure on the network transmission, and it is obviously impossible to transfer from the server to the client in real-time. On the premise of not improving the hardware performance, the first problem to be solved is to improve the transmission capacity of the network so that the model file can be transferred from the server to the client through the network as soon as possible.

One easy way to think of is to compress the geometry and topology information in the mesh for the model file so that the vertex data and connection relationship of the model can be represented by as few bits as possible. In this way, the transmission speed is improved, the waiting time of users is reduced, and the rendering is done by the browser. The server has only bandwidth pressure, and there is not too much computing pressure. In this way, the complexity of the model and the rendering ability of the browser jointly affect the user experience.

The rendering methods based on mesh compression can be divided into single resolution mesh compression rendering and multi-resolution mesh compression rendering.

1) SINGLE RESOLUTION ORIENTED MESH COMPRESSION RENDERING

For the compression of grid data, some common data compression techniques can be used, such as blue wave LIF Markov chain coding (LZMA). However, the biggest disadvantage of these algorithms is that for web browsers, the speed of decoding data using JavaScript is several orders of magnitude slower than that of local code [1].

The most well-known grid compression technology is the WebGL loader algorithm proposed by Google based on WebGL [2]. This method relies on the bounding box coordinate quantization of vertex position and uses incremental coding to transmit the difference between the above values. It can use JavaScript for quick decoding operations. However, because it uses UTF-8 encoding, the vertex cache of this method is limited to no more than 55296 (0xd800). Recently, Google proposed a new grid compression algorithm Draco [3]. Compared with WebGL loader, this method removes the dependence on UTF-8 coding and improves the compression performance. However, its decompression performance is relatively poor.

The above algorithm can reduce the amount of data in the model to a certain extent, thus reducing the storage space and network transmission time. However, the problem is that the client still has to wait until all the model data has been downloaded before rendering. It is difficult to achieve real-time remote rendering for complex models. And after the transmission is completed, the browser will still take a long time to render the first frame.

2) MULTI-RESOLUTION ORIENTED MESH COMPRESSION RENDERING

Since the details observed by the human eye are inversely related to the distance between the viewpoint and the object, how to use level of detail (LOD) to adjust the model accuracy at different view positions becomes particularly important for complex models. The traditional discrete LOD technology usually needs to build models with different resolution levels. However, the models at different levels are independent of each other. Although the more refined hierarchical model already contains the data in the coarser hierarchical model, even if the latter has been transmitted, it is impossible to add data on the basis of it to reach the fine level. Instead, all the data of the hierarchy must be transmitted from the beginning, which undoubtedly increases the amount of data transmitted on the network as a whole.

The problem of discrete LOD technology can be solved by progressive transmission. Progressive transmission of the 3D model means that the server first transfers the simplified model to the client. As the transmission process continues, more and more details are added to the simplified model until all model data transfer is completed. In this way, it is not necessary to wait for the whole model to be transferred before operation, but the client can render at the beginning of the transfer, that is, "rendering while loading". Progressive transmission is a kind of stream coding for a 3D model, which is similar to streaming media transmission. It effectively solves the contradiction between limited network bandwidth and complex 3D models.

Progressive transmission is closely related to mesh simplification. In 1996, hopper and others first proposed the progressive mesh (PM) simplification algorithm for 3D models [4]. It is an iterative contraction mesh simplification method and is very suitable for progressive transmission. A large number of related algorithms are optimized and improved based on this algorithm. However, for the B / S-based display structure, it is difficult to effectively implement progressive grid decoding with JavaScript on the client-side because JavaScript is an explanatory language, its language execution speed is slow, and the decoding process requires a lot of time-consuming CPU operations. In this way, even if the compression transmission speed is faster, the long decoding time will offset the advantages of progressive transmission and may bring more delay. In view of the shortcomings of PM, such as relatively slow encoding, some scholars have implemented a relatively fast progressive grid decoding algorithm using the half-edge structure based on JavaScript [5]. Some scholars use spherical Fibonacci points to store normal vertex vectors with low resolution to realize a relatively fast progressive mesh decoding algorithm [6].

In addition, sometimes, the model file is very large because its related texture data accounts for a large proportion of the size. Therefore, some scholars have proposed remote rendering technology based on progressive texture [7]. The idea of this method is similar to that of progressive transmission based on the grid.

Although the multi-resolution oriented mesh compression method can effectively reduce the waiting time when users see the first frame, they only see the outline of the model. If they want to see the details, they still have to wait for a long time. In addition, the grid-based method requires high computing power of the terminal and cannot guarantee the security of the model data. Once the model data is transmitted to the client through the network, users can copy and propagate the 3D model. Although additional watermark information can be embedded in the model data, it can only prove that the use of the model belongs to piracy after the fact, and it cannot prevent other users from stealing the model and using it on other occasions.

B. IMAGE-BASED RENDERING

The traditional model rendering method is to display 3D objects on the screen through the triangular mesh, which is not the only method for 3D rendering, nor the most appropriate method.

Image-based rendering (IBR) provides a new viewpoint for 3D model rendering. The advantage of using triangulation to represent the model is that the representation of the model is independent of the position of the camera. The biggest advantage of using images to represent 3D models is that the rendering time is directly proportional to the image resolution to be drawn, rather than the number of vertices and patches of 3D models. It overcomes the defect that the rendering speed of traditional computer graphics decreases with the increase of scene model complexity. In addition, a single image can be quickly rendered to the screen, while the traditional triangular patch-based rendering method needs to go through a series of complex pipeline processes.

However, the IBR method needs to transmit a series of high-definition images continuously, which puts great pressure on network bandwidth. Therefore, most scholars improve it from two aspects: data compression and reducing the transmission frequency of images. For example, some scholars proposed to use the wavelet method to replace the traditional JPEG image compression method [8], and some scholars proposed to transform 3D scenes into panorama while transmitting panorama through progressive transmission protocol [9]].

Some scholars have proposed a rendering method based on image deformation [10]. In other words, different from the way of direct image rendering, image morphing based rendering usually uses the depth information in the image (including a color image and a depth image) to synthesize the new view image, so it does not need to transmit the image of each viewpoint. However, due to occlusion in the scene, the composite image usually contains holes. Some scholars have proposed to make adaptive compensation for the missing information in the depth map and use a Gaussian filter to reduce holes [11]. Some scholars have proposed a method of synthesizing new viewpoints from images of multiple viewpoints to minimize holes and use oversampling to compensate for the existing image missing [12]. Some scholars have proposed to use the supersize reference image to solve the whole problem [13]. Using the image morphing method, in addition to the loss of information which affects the visual effect, view synthesis calculation is a processor intensive task, and its time cost is proportional to the image resolution, which undoubtedly challenges the fast computing ability of the client.

There are also some methods that combine mesh-based rendering with image-based rendering, which are called hybrid rendering methods in this paper. There are two directions for the main joint points: one is to use different rendering methods for different interactive operations. It divides user interaction into dynamic operation and static browsing. For dynamic interaction, rough 3D models [14], sparse triangular mesh patches [15], or point cloud models [16] are used to achieve real-time interaction. The direct browsing method is based on the static image. This method has also been applied to cultural heritage [15], [16]. The other is to use different rendering methods for distant and close range.

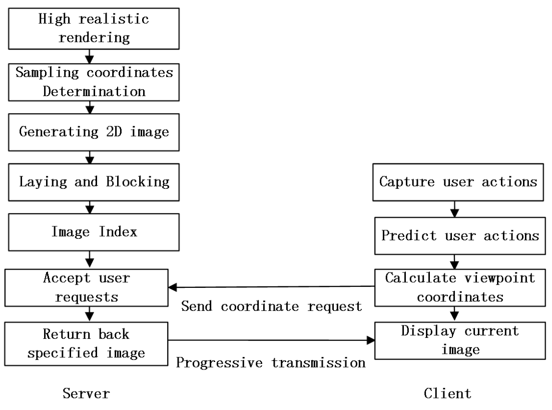


FIGURE 1. Overall display flow chart of our real-time approach based on model pre-rendering.

Because compared with the close range, the same depth change perspective brings less space to people, so we can use the image-based rendering method to display the distant view. However, the disadvantage of this method is that different data streams will cause visual differences in the interaction process, which will bring users an unnatural experience. In addition, due to the combination of grid-based rendering, there is still the risk of model data leakage.

III. METHOD

Because the client browser has limited resources and computing power, it is difficult to apply the grid-based rendering method to ensure real-time performance. The image-based rendering method directly represents the whole 3D model through a set of 2D images, which is especially suitable for remote display or roaming of complex 3D models. Compared with the grid-based rendering method, the first frame delay is smaller.

Therefore, in order to minimize the delay (including network delay and rendering delay), so that users can get the most fluent interactive experience. In order to realize the copyright protection of the 3D model (not directly exposing the model data to users on the Internet) and the interactive characteristics of the model, this paper proposes a real-time display method based on model pre-rendering. The main process is shown in Figure 1.

On the server-side, firstly, the Monte Carlo path tracing algorithm is used to render the 3D model with high fidelity, and then the sampling coordinates are determined to pre-render it into a two-dimensional image (see III-A). Then the images are layered and partitioned (see III-B), and all images are indexed and stored on disk (see III-E). On the client-side, after capturing the user's operation, it is necessary to calculate the current new view coordinates and send the coordinates and the predicted viewpoint coordinates (see III-D) to the server for processing. After receiving the user's request, the server looks up the corresponding two-dimensional image according to the viewpoint coordinates and returns it to the client through network transmission (see III-C). After receiving the returned results, the client directly displays the image to the user. Our method captures the user's interaction action in real-time on the client, analyzes the user's action, calculates the viewpoint position of the 3D model to be calculated, and then goes to the server to query and fetch, and then the corresponding image data is queried and obtained on the server.

A. MODEL PRE-RENDERING

In this paper, a method based on subdivision surface is proposed to transform a 3D model into a 2D image set. The specific ideas are as follows: According to the difference in the size and shape of the 3D model, the axis of the 3D model is calculated to align the bounding box, and the center of the bounding box is aligned with the origin of the world coordinate system. Then a sphere containing the whole 3D model is constructed, and the center of the sphere coincides with the origin. Then the outer surface of the sphere is subdivided. That is, a series of points are selected on the sphere surrounding the model, and the triangular mesh is established by these points. Then a regular polyhedron is constructed by these triangular meshes. Then, a camera is constructed at each vertex of the polyhedron. The camera faces to the origin, and the scene is rendered when the camera is at this position, and a highly realistic two-dimensional color image is obtained.

Thus, the whole model can now be represented by a binary relationship $< R, E >$. Where R is the connection between camera positions and E is the collection of all cameras. $E = \{e_i | i = 1, 2 \ldots |E|\}$, $|E|$ is the length of the camera set. $e_i = (c_i, p_i)$, c_i is the 2D image at the *ith* camera position. $p_i = (pc_i, near_i, far_i, aspect_i, fov_i)$, p_i is the parameter of the *ith* camera, where pc_i is the camera position, pl_i is the camera facing position, $near_i$ and far_i is the distance between the near plane and the far plane in the perspective projection of the camera, $aspect_i$ is the width height ratio of the near plane of the camera, and fov_i is the vertical view angle of the camera.

In theory, as long as the sampling density of the camera is large enough, we can reconstruct the original three-dimensional model based on the SFM theory, and the reconstruction result is unbiased. When the number of camera samples is fixed, uniform density sampling is adopted in order to ensure the same sampling density at each spatial position of the model. That is, the camera sampling points are evenly distributed on the surface of the whole model, and the spherical symmetry of the distribution is ensured. The mathematical definition of uniformity is to maximize the minimum distance between any two points. This is obviously an optimization problem. Taking the minimum distance of points on the sphere as the objective function will make the stability of the function poor, so it is difficult to find the optimal solution. Here, the reciprocal sum of Euclidean distances between all points is set as the objective function, and its minimum value is calculated, where $d(i, j)$ is the Euclidean distance between sampling points i and j

$$minimize : \sum_{i<j} \frac{1}{d^2(i, j)} \tag{1}$$

When there are many sampling points, it is very difficult to prove and solve the optimal solution of the objective function. In this paper, the solution of approximate distribution is solved. It is assumed that n sampling points are uniformly and symmetrically distributed on the surface of the ball, the coordinates of the spherical center are (0,0,0), and the radius of the sphere is r. If $i = 1, 2 \ldots N$, then the spherical coordinate system coordinates $pc_i(r, \theta, \phi)$ of the *ith* point is

$$\begin{aligned} k &= \frac{2i - n - 1}{n} \\ \theta &= arcsin(k) \\ \phi &= \theta\sqrt{n\pi} \end{aligned} \tag{2}$$

Transforming spherical coordinate system into Cartesian rectangular coordinate system $pc_i(x, y, z)$, then

$$\begin{aligned} x &= rsin\theta cos\phi \\ y &= rsin\theta sin\phi \\ z &= rcos\theta \end{aligned} \tag{3}$$

By introducing formula 2 into formula 3, we can get the following results:

$$\begin{aligned} x &= rcos(\sqrt{n\pi} sin^{-1}(k)\sqrt{1 - k^2}) \\ y &= sin(\sqrt{n\pi} sin^{-1}(k)\sqrt{1 - k^2}) \\ z &= rcosrsin^{-1}(k) \end{aligned} \tag{4}$$

In general, if the spherical center coordinate is x_0, y_0, z_0, then the sampling point pc_i is

$$\begin{aligned} x &= rcos(\sqrt{n\pi} sin^{-1}(k)\sqrt{1 - k^2}) + x_0 \\ y &= rsin(\sqrt{n\pi} sin^{-1}(k)\sqrt{1 - k^2}) + y_0 \\ z &= rcosrsin^{-1}(k) + z_0 \end{aligned} \tag{5}$$

B. LAYERING AND BLOCKING

The method in the previous section can only render two-dimensional images with fixed viewpoints. Considering that users need to be able to browse the model in a short distance and a long distance in the process of interaction, that is, the viewpoint of the model when rendering, that is, the position of the camera needs to change constantly. This paper introduces the view-dependent level of detail (LOD).

Because the model fineness that the camera can observe is inversely related to the distance from the camera to the model, the use of level of detail (LOD) technology enables low-resolution images to be rendered if the model details are not high, that is, when the viewpoint is far away from the model. On the other hand, if the details of the model are high, that is, when the viewpoint is close to the model, the high-resolution image will be rendered. In this way, when you only need to browse the relatively rough model, the amount of scene data that needs to be downloaded through the network can be greatly reduced.

Although the change of viewpoint is a continuous process, in order to save the amount of data from being calculated and stored, we use the discrete view change to approximate the continuous view change and sets a level for each fixed viewpoint position.

Assuming that there are m levels, the entire 3D model can now be expressed as $L = \{l_1, l_2, \ldots l_m\}$, where $l_i = \{r_i, n_i, < R_i, E_i >\}, i = 1, 2 \ldots m$. For each level l_i, we all need to construct a circumscribed sphere with a radius of r_i. The number of sampling points is n_i.

Suppose that the axis of the model is aligned and the size of the bounding box is $(bMin, bMax)$, where $bMin$ is the coordinate of the point with the smallest coordinate value in the vertex of the bounding box, and $bMax$ is the coordinate of the point with the maximum coordinate value in the vertex. For the lowest level l_i, that is to say, the outfall at the farthest point of view. Its radius r_1 should be:

$$\begin{aligned} rw &= \frac{bMax.x - bMin.x}{2 \times aspect \times tan(\frac{fov}{2})} \\ rh &= \frac{bMax.y - bMin.y}{2 \times tan(\frac{fov}{2})} \\ r1 &= max(rw, rh) + bMax.z \end{aligned} \tag{6}$$

To make the calculation simple and make the scale of each scaling the same, we divide all levels evenly, that is, to ensure that the distance between all adjacent levels is equal, then for $r_i, i = 1, 2 \ldots m$. It should satisfy:

$$r_i = \frac{(m + 1 - i) \times max(rw, rh)}{m} + bMax.z \tag{7}$$

As for the number of sampling points, this paper sets the initial value $n_1 = 256$ and satisfy $n_i = 2 \times ni - 1$. The specific layer diagram is shown in Figure 2 After the above derivation, we can calculate which level should be used for any given viewpoint position. Suppose that the distance from the current viewpoint to the origin is D, the resolution of the current browsing window is $sw \times sh$, and the resolution of the rendering window (near plane) is $w \times h$. The current level l_i that should be transferred and displayed should satisfy (where $i = 1, 2 \ldots mandj < i$):

$$l_i = args_{l_i}(l_i \leq d \times \frac{max(sw, sh)}{max(w, h)} \leq l_j) \tag{8}$$

It can also be seen from the above that the resolution of rendering (generated image) is different from that of browsing (browser window). Generally speaking, the rendering resolution should be greater than or equal to the browsing resolution. In this paper, the image rendered by each camera in each layer is cut into blocks of the same size (except when the segmentation is not enough). The size of each block is 512 × 512. In this way, when the browse resolution is less than the rendering resolution, only the currently visible blocks can be loaded, which further reduces the amount of data transmitted on the network and the rendering speed of the browser. Moreover, after blocking, the browser can also use the concurrent loading of images to further speed up the operation fluency during browsing.

For a given viewpoint, we can not only deduce the level where it should be located but also get which camera sampling point should be responsible for rendering the image of the viewpoint position. However, as with the level, the sampling points of the camera are also discrete. Therefore, the viewpoint is temporarily classified to the nearest sampling point in the viewpoint. That is to say, in fact, the user cannot see the 3D model at the viewpoint (the same for the hierarchy) but can only see the 3D model where the viewpoint coincides with the sampling point. Although in theory, users will feel a sense of jumping because of the discontinuity of sampling when browsing. However, due to the use of LOD based layering and blocking, in fact, when the viewpoint is far away from the model, such a sense of jumping will be greatly reduced. Moreover, as the viewpoint gets closer and closer to the model, the number of sampling points is also exponentially increased. As a whole, the actual sense of jump is not particularly obvious (of course, this is also very limited by the transmission speed of the network). This is also confirmed by the actual experimental results.

C. NETWORK TRANSMISSION

Even if the 3D model is converted into an image, it still needs very high network bandwidth for smooth transmission. For example, for a $32 - bit$ true-color image with a resolution of 512 × 512, assuming its compression rate is 0.1, a total bandwidth of 19.2mbps is required at the frame rate of 24fps. If the frame rate is 60fps, a bandwidth of 48mbps is required. Obviously, such a large bandwidth browsing on the user's network requirements will be very high.

So on the basis of the previous section, in order to further improve the fluency of browsing, this paper uses the progressive method to transfer the rendered image. Many image formats such as JPG, PNG, Webp support progressive encoding. Most browsers also support progressive JPG and PNG by default, while Webphas a smaller support range. In terms of compression performance, according to official experiments, the size of lossless compressed Webp is about 20% smaller than that of PNG files, and the size of lossy compressed Webp images is about 30% smaller than that of jpg images with the same quality index. In addition, when Webpof, the image is reduced by about 50%. When Webp compresses JPG to 80% of the quality of the original image, the size of the image is reduced by about 70%. Therefore, compared with JPG, Webp image can greatly reduce the size of the image, and the user will Webp notice the difference.

The main reason why the compression performance of lossy JPG is not as good as Webp is that compared with Huffman coding used by JPG, Webp uses entropy coding technology with better compression performance, namely Boolean arithmetic coding. Furthermore, the performance of the adaptive block is further improved by using Webp compression technology. However, according to the official test results, the compression speed of JPG is ten times faster than Webp, and the decompression speed of JPG is 1.5 times faster than the Webp. In practical application, the additional

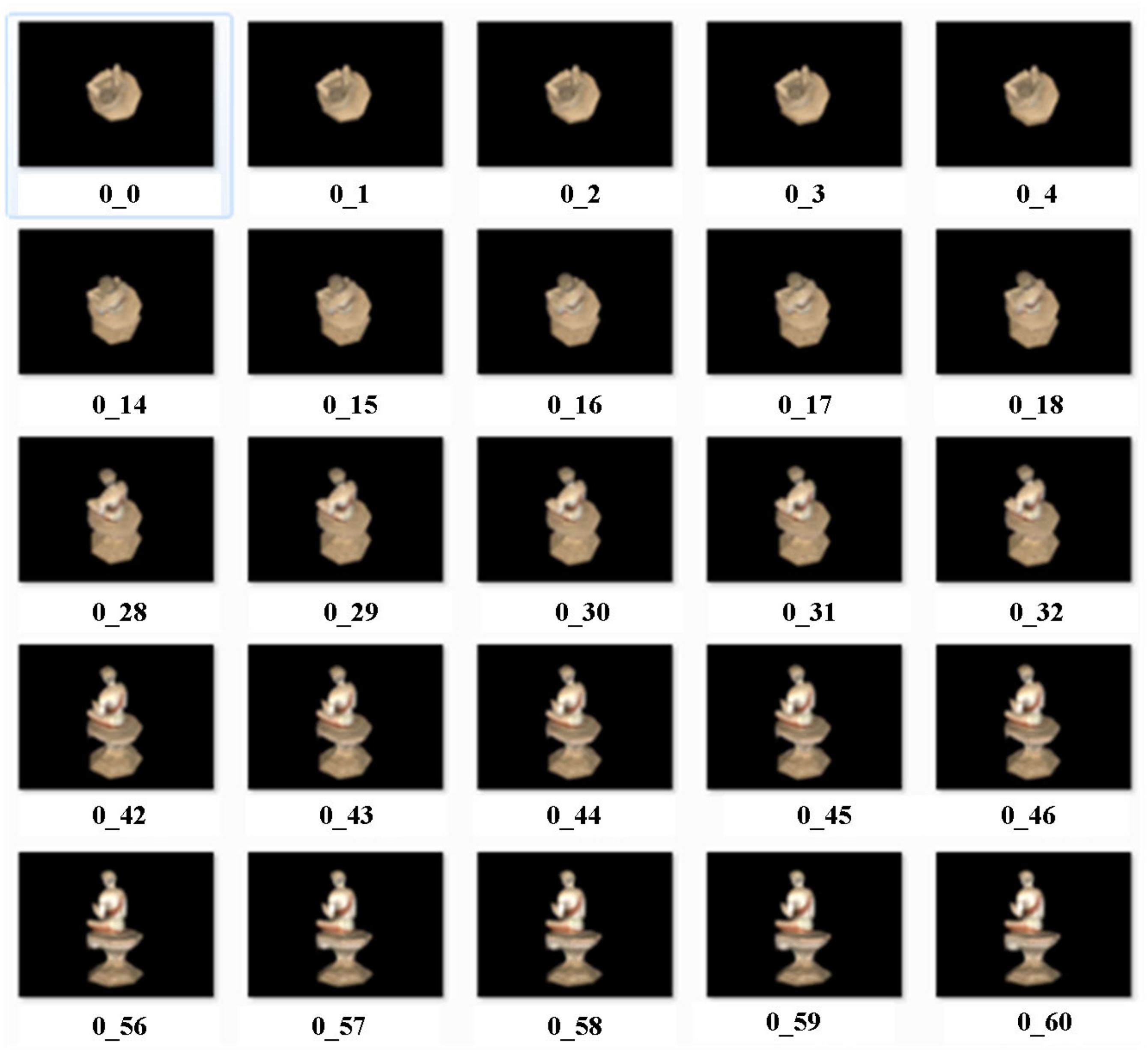


FIGURE 2. **Layered diagram used in our pre-rendering method. The number below each small image in the above figure represents the number of images rendered at different viewpoint positions. Our method numbers the images rendered at different viewpoint positions.**

performance consumption of the server is not a problem. The main problem is whether 1.5 times of decoding speed can offset the reduction of network transmission delay caused by volume reduction.

In order to compare the effect of JPG and Webp, this paper compares the loading time of JPG and Webp images with the same quality under the same network conditions. The number and size of pictures tested are shown in Table 1.

As can be seen from Figure 3, although Webp consumes a longer compression time, due to the higher compression rate, the images are relatively smaller, and the network transmission time is shorter. In this way, the loading speed of the whole page becomes faster. In addition, as the number of images increases, the loading time gap between Webp images and jpg images will become larger and larger. Therefore, using Webp can not only save bandwidth but also improve the loading efficiency.

To sum up, this paper adopts JPG as the mainstream transmission method. However, for the browser that supports Webp, the background server will convert the rendered JPG image into Webp format for transmission. This adaptive progressive transmission reduces the size of network transmission to the greatest extent, thus effectively improving the browsing experience of users.

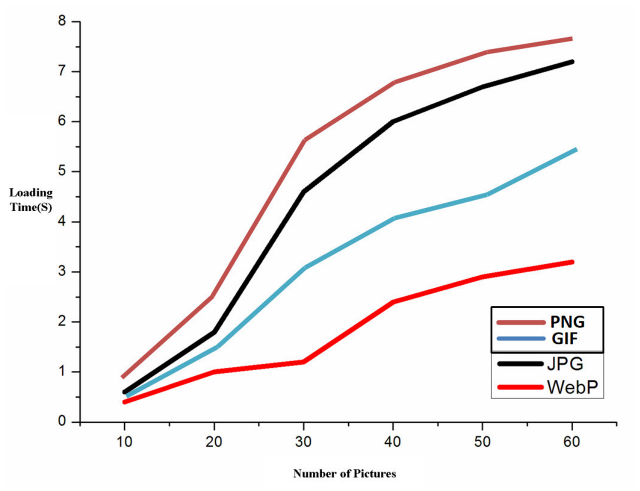


FIGURE 3. The loading time of JPG and WebP varies with the number of pictures in Table 1.

TABLE 1. Number and size of JPG and WebP images tested.

Method	10	20	30	40	50	60
JPG	112	248	550	801	942	1126
WebP	62	130	190	269	342	402

D. IMAGE PRELOADING

When browsing 3D models, users usually have a series of interactive operations, such as rotation, scaling, and so on. Suppose the user carries out a series of fast and intensive interactive operations for a period of time. Because the view point changes, the client needs to constantly request new pre-rendered images from the server. At this time, the pressure on the network load and the client will be very great. If the client cannot complete the image addition within the specified time (about 16.7ms if the frame rate is 60fps) If it is loaded and displayed, the frame will drop. That is, the user will feel stuck, which will greatly affect the fluency experience of user operation. In addition, it is necessary to wait until the end of each interactive operation before sending the operation to the server for processing, and due to the existence of network and processing delay, when the user carries out the i interaction operation, it will be sent to the server for processing. When the user receives the processing result from the server, the user may have already carried out the $i + 1$ or later interaction operation. At this time, the result of the server is obviously outdated to the user. In order to solve the lag of server-side processing, this paper predicts the user's interaction behavior through the previous interaction operations, and sends the predicted interaction operations to the server for processing, so as to realize preloading.

The operation prediction implemented in this paper mainly aims at continuous user interaction behavior. In contrast to the continuous user interaction behavior, the discrete user interaction behavior is usually irregular, which makes the prediction more complex, and the prediction accuracy is low. Because continuous interaction is usually the same type

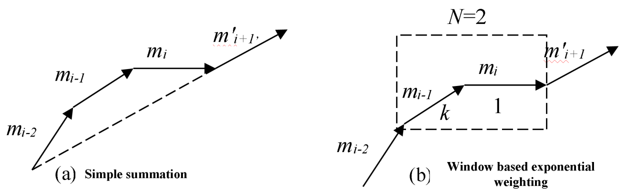


FIGURE 4. Trajectory prediction based on historical motion behavior used in our approach.

of continuous interaction, its interaction trajectory can be regarded as a curve, so linear extrapolation and other methods can be used to predict the new interaction position.

This paper presents an exponential weighted prediction algorithm based on the window. Take the continuous rotation operation of a fixed level as an example. That is, the viewpoint $pc_i = (r, \theta, \phi)$ coordinates remains unchanged, so we only need to consider the change of two-dimensional coordinates (θ, ϕ). Suppose that the viewpoint positions of the current *ith* moment and the previous n rotation operations are $p_{i-n+1}, p_{i-n+2}, \ldots .p_i$, respectively. The corresponding vector displacement of each movement is $m_{i-n+1}, m_{i-n+2} \ldots m_i$. And the predicted value of P_{i+1} is p^1_{i+1}, the predicted value of m_{i+1} is m^1_{i+1}. Because the time interval between each operation is very short, we ignore the effect of acceleration. The simplest prediction method is to sum the n vector displacements to get the displacement direction at the $i + 1$ moment. The details are shown in Fig. 4(a). In order to simplify the derivation, we first consider the case of $n = 2$

$$\begin{aligned} m^1_{i+1} &= m_i + k(p_i - p^1_i)p^1_{i+1} \\ &= p_i + m_{i+1}m^1_{i+1} = (1-k)m_i + km^1_i \end{aligned} \tag{9}$$

And so on, when $n = 5$,

$$m^1_{i+1} = \sum_{n=1}^{5}((1-k)k^(n-1)m_{i-n+1} + k^n m^1_{\text{i-N+1}}) \tag{10}$$

The advantage of an exponential weighted prediction algorithm based on the window is that it considers the influence of historical interaction behavior at different times, especially the recent interaction behavior, so its accuracy is higher.

However, in theory, there are always some errors between the predicted value and the real value, so we must deal with all kinds of possible deviations. When the deviation of the predicted value is in a certain small range, this paper will directly show the predicted image. When the deviation is large, if the image of the prediction result has not been transmitted, the request will be terminated immediately. Otherwise, the image of the prediction result needs to be discarded or cached for future use.

In addition, when the user stops interaction, we can also use the relative free time of the processor and network to preload the image. The main idea is: when the system does not monitor the occurrence of new events within a certain period of time (including mouse and keyboard), this paper will preload the images of several viewpoints around the current viewpoint (which can also be extended to higher-level viewpoints). Once the user starts a new interaction, all new and incomplete preloading operations are terminated. Of course, preloading will consume more storage and network resources, so the specific range and number of preloads can be dynamically changed according to the user's settings.

Finally, in addition to interactive prediction on the client-side, when the user interaction speed is faster, users tend to ignore the details of the middle screen. Therefore, the server can compress the image to a greater extent before transmitting the image. The combination of user prediction and dynamic server-side image compression coding can effectively reduce the network load and improve the smoothness of operation.

E. SERVER IMAGE INDEX

After prerendering operation and blocking and layering, we have stored a large number of block images. In the previous section, different aspects of optimization are used to reduce the correlation delay so as to improve the real-time interaction. In this section, we mainly consider how to reduce the time required to request partitioned images from the server. In essence, the reason for this problem is that data requests are much faster than disk reads and writes. To solve this problem, it is necessary to reduce the disk access time per request. Data partition and reorganization is a method to solve this problem. Its idea is to cross-deploy adjacent data blocks on multiple disk servers so that multiple data blocks can be read at one time in parallel.

There are three well-known methods of data partition and reorganization: disk modular, logical XOR, and Hilbert curve allocation method [17]. Based on the Hilbert curve allocation method, this paper deploys a block graph on multiple disk servers. Hilbert curve assignment method can organize a two-dimensional block grid into linear form. This order is also called the Hilbert order. In this paper, $h(x, y)$ is used to represent its value. The following figure 5 is a schematic diagram of the access sequence of the third-order Hilbert curve (for a fixed level view-point, the X and Y axes of the image actually correspond to the θ and ϕ of the spherical coordinate system coordinates of the circumscribed sphere sampling point at this time respectively).

However, the time complexity of the traditional Hilbert curve coding method to transform (x, y)) coordinates to the corresponding position on the Hilbert curve increases exponentially with the increase of data volume. Therefore, this paper proposes an improved Hilbert coding algorithm.

Because the high order Hilbert curve H_{i+1} is formed by four sub curves $H_i$hi. Except for the four basic Hilbert curve shapes in Figure 6, all sub curves can be parent curves. In addition, each sub curve is similar to the curve of the basic shape. That is, all Hilbert curves are variations of the four basic curves.

The properties of the Hilbert curve ensure that when two Hilbert curves have the same trend, they will have the same structure. In this paper, a recursive quadrant partition Hilbert curve generation method is adopted, and Hilbert code is calculated based on the invariance of parent curve and child curve. The details of the algorithm are as follows:

1) To judge the shape of a given Hilbert curve: first, calculate which quadrant the current block graph belongs to, and then look up the Hilbert sub curve code I in table 2 to get the shape of the sub curve in the quadrant.

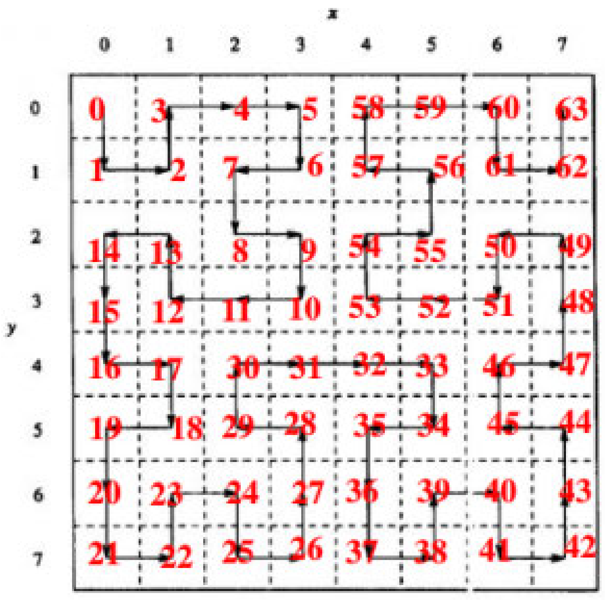


FIGURE 5. Access order of third order Hilbert curve used in our approach.

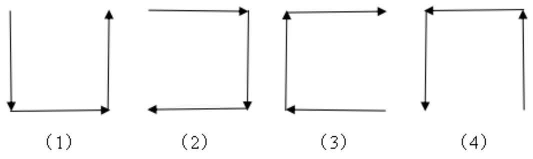


FIGURE 6. Hilbert curves of four different basic shapes used in our approach.

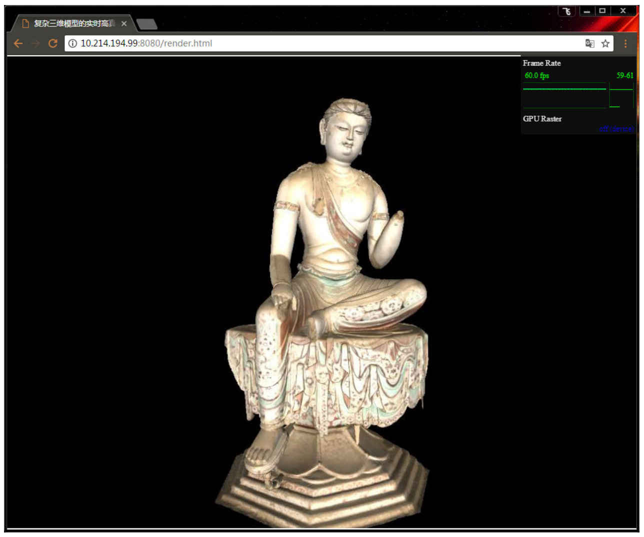


FIGURE 7. Exhibition effect of 196 Buddha statue model in Mogao Grottoes.

TABLE 2. Hilbert sub curve coding 1.

Parent curve shape	1	2	3	4
1	2	3	1	1
2	1	2	4	2
3	3	1	3	4
4	4	4	2	3

TABLE 3. Hilbert sub curve coding 2.

Parent curve shape and quadrant	Corresponding code h
11, 21, 34, 44	h+=0
13, 22, 33,42	h+=1
14, 24, 31, 41	h+=2
12, 23, 32, 43	h+=3

TABLE 4. Hilbert sub curve coding 3.

Parent curve shape and quadrant	Corresponding code h
11, 21, 34, 44	h+=1<<(2×n)× 0/4
13, 22, 33,42	h+=1<<(2×n)× 1/4
14, 24, 31, 41	h+=1<<(2×n)× 2/4
12, 23, 32, 43	h+=1<<(2×n)× 3/4

2) The quadrant is divided into four sub-quadrants. If the quadrant cannot be divided, the coding is calculated according to table 3 Hilbert sub curve code II and returned.

3) Determine which sub quadrant the block graph belongs to. According to the curve shape of the parent quadrant and the position of the sub quadrant, the curve shape of the sub quadrant is judged. Then, the initial value of the sub quadrant curve is obtained according to table 4 Hilbert sub curve code III.

Note that the upper left corner is the I quadrant, the upper right corner is the second quadrant, the lower-left corner is the third quadrant, and the lower right corner is the fourth quadrant. Table 2 shows that the shape of the sub curve is determined by the shape of the parent curve and the quadrant of the sub curve. The parent curve shape corresponds to the four basic curve shapes shown in Figure 6. Each cell in the first column of tables 3 and 4 consists of four numbers separated by commas. The first digit of each number indicates which of the four sub curves belongs to, and the second digit represents which of the four sub-quadrants. n in Table 4 represents the current level.

IV. EXPERIMENT AND ANALYSIS

In this paper, through the model of 196 Buddha statues in Mogao Grottoes, the actual browsing effect of this method in the browser is tested. The geometry file size of the 3D model is 120MB, the texture file size is 1.24GB, the number of vertices is 2482263, and the number of patches is 827421. The waiting time of the first screen in the conventional mode is about 67s, while the waiting time of the first screen based on this method is about 1s (the corresponding HTML, JS files, and corresponding configuration files need to be loaded at the same time). In this paper, the 196 Buddha statue model of Mogao Grottoes is divided into five levels ($M = 5$).

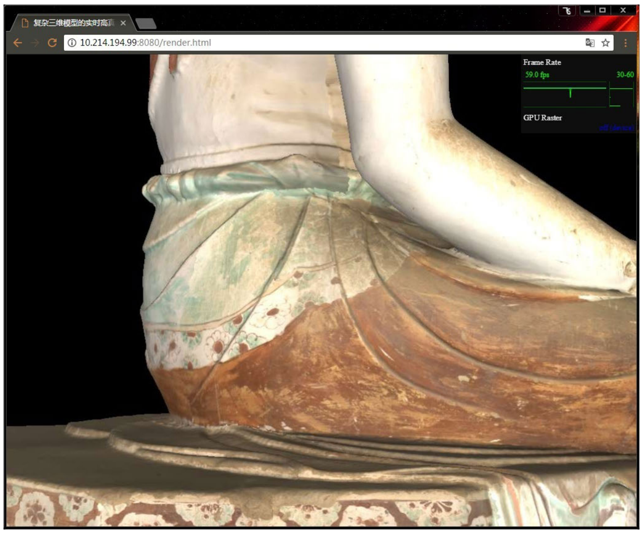


FIGURE 8. Exhibition effect of 196 Buddha statue model in Mogao Grottoes.

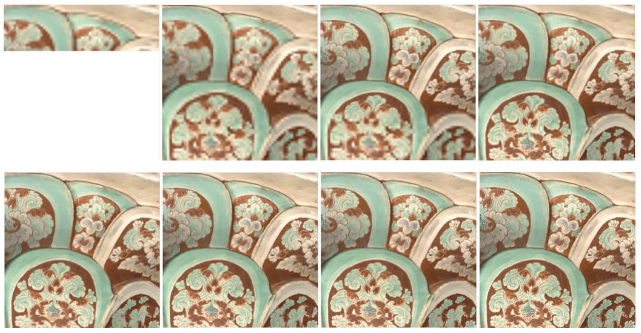


FIGURE 9. Adaptive progressive transmission display effect in different ditail layer in our approach.

TABLE 5. Browser average rendering frame rate.

Hierarchical	Frame Rate
1	47.4
2	42.7
3	36.1
4	28.6
5	23.4

The pre-rendered image resolution of each sampling point is 1024 × 768. Figure 7 below shows the display effect when it is located on the first floor and the fifth floor, respectively. Figure 9 shows the effect of adaptive progressive transmission. You can see that the texture details gradually change from blur to clarity (from top to bottom, from left to right). When the model is set to auto rotation mode, the browser's average frame rate (FPS) within 20s is shown in Table 5.

As can be seen from Table 5, as the number of corresponding sampling points increases with the increase of layers, the corresponding frame rate is also getting lower and lower. In addition, due to the automatic rotation, the accuracy of the prediction algorithm will be relatively high so that the corresponding frame rate will be improved to a certain extent.

TABLE 6. Comparison of browser rendering frame rate changes.

Hierarchical	Δ F(preloading)	Δ F(no preloading)
1	36.6	46.9
2	33.7	45.1
3	34.2	48.2
4	28.4	40.5
5	22.9	36.7

When automatic rotation is prohibited, and a fast manual rotation operation is performed around the model, with and without image preloading, the browser's change ΔF in FPS within 20s is shown in Table 6. As can be seen from Table 6, the range of frame rate variation without image preloading is much larger than that with image preloading. This result is reflected in the actual operation: in the continuous interactive operation, especially a series of intensive fast operation, when there is no image preloading, the user will feel smooth and sometimes stuck. When the image is preloaded, the user's card sense will be relatively reduced. That is, when browsing, the user's sense of fluency will be stronger.

V. CONCLUSION

In order to enable users to browse complex 3D models on the web in real-time and to ensure a high sense of reality in the process of the display, based on the existing research, this paper starts from the interactive characteristics of 3D models and considers how to essentially reduce the bandwidth load of 3D models in network transmission. In this paper, the compression method and multi-level of detail technology are used to reduce the details which are not very sensitive or cannot be observed by human eyes under the corresponding conditions; consider how to preload some data according to the user's operating habits. Finally, design experiments and verify the feasibility and effect of this idea in the 3D model display.

The technology proposed in this paper will provide new ideas for solving similar problems and promote the related research work and practical engineering application as well as the development of industry, such as animation, film, and virtual reality. In addition, taking the cultural heritage exhibition as the application field, the system implemented in this paper is applied to the digital display of museum resources, which helps to speed up the progress of digital display of museum resources, solve the inconvenience of physical, cultural relics transmission, reduce a large number of human service costs, including commentators, in the process of physical display, and alleviate the museum during peak hours, especially during holidays Congestion in the museum. For users, through this system to browse cultural relics, you can do a convenient and fast browsing of all kinds of cultural relics models you want, and bring users a new interactive operation experience.

REFERENCES

[1] F. Ponchio and M. Dellepiane, "Fast decompression for Web-based view-dependent 3D rendering," in *Proc. 20th Int. Conf. 3D Web Technol.*, Jun. 2015, pp. 199–207.

[2] A. Blume, W. Chun, D. Kogan, V. Kokkevis, N. Weber, R. W. Petterson, and R. Zeiger, "Google body: 3D human anatomy in the browser," in *Proc. ACM SIGGRAPH Talks*, 2011, p. 19.

[3] Google. (Jan. 2017). *Introducing Google Draco*. [Online]. Available: https://opensource.googleblog.com/2017/01/introducing-draco-compression-for-3d.html

[4] H. Hoppe, "Progressive meshes," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 99–108.

[5] G. Lavoué, L. Chevalier, and F. Dupont, "Streaming compressed 3D data on the Web using JavaScript and WebGL," in *Proc. 18th Int. Conf. 3D Web Technol. (Web3D)*, 2013, pp. 19–27.

[6] A. Evans, J. Agenjo, and J. Blat, "A pipeline for the creation of progressively rendered Web 3D scenes," *Multimedia Tools Appl.*, vol. 77, pp. 1–29, Dec. 2017.

[7] J. E. Marvie and K. Bouatouch, "Remote rendering of massively textured 3D scenes through progressive texture maps," in *Proc. 3rd IASTED Conf. Vis., Imag. Image Process.*, vol. 2, 2003, pp. 756–761.

[8] S. Deb, S. Bhattacharjee, S. Patidar, and P. J. Narayanan, "Real-time streaming and rendering of terrains," in *Proc. ICVGIP*, 2006, pp. 276–288.

[9] R. W. Pazzi and A. Boukerche, "PROPANE: A progressive panorama streaming protocol to support interactive 3D virtual environment exploration on graphics-constrained devices," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 11, no. 1, pp. 1–22, Aug. 2014.

[10] P.-P. Vázquez and M. Sbert, "Bandwidth reduction for remote navigation systems through view prediction and progressive transmission," *Future Gener. Comput. Syst.*, vol. 20, no. 8, pp. 1251–1262, Nov. 2004.

[11] C.-H. Hsia, "Improved depth image-based rendering using an adaptive compensation method on an autostereoscopic 3-D display for a kinect sensor," *IEEE Sensors J.*, vol. 15, no. 2, pp. 994–1002, Feb. 2015.

[12] S. Shi, K. Nahrstedt, and R. Campbell, "A real-time remote rendering system for interactive mobile graphics," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 8, no. 3s, pp. 1–20, Sep. 2012.

[13] P. Bao and D. Gourlay, "A framework for remote rendering of 3-D scenes on limited mobile devices," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 382–389, Apr. 2006.

[14] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno, "Protected interactive 3D graphics via remote rendering," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 695–703, Aug. 2004.

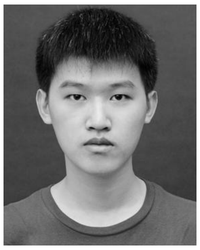
[15] Y. Okamoto, T. Oishi, and K. Ikeuchi, "Image-based network rendering of large meshes for cloud computing," *Int. J. Comput. Vis.*, vol. 94, no. 1, pp. 12–22, Aug. 2011.

[16] C. Su, J. Ping, Q. Yue, and S. Xukun, "Protected-3DMPS: Remote-rendering based 3D model publishing system in digital museum," *J. Comput. Inf. Syst.*, vol. 2, no. 1, pp. 277–283, 2006.

[17] C. Faloutsos and P. Bhagwat, "Declustering using fractals," in *Proc. 2nd Int. Conf. Parallel Distrib. Inf. Syst.*, 1993, pp. 18–25.

CHUAN XIE is currently an Associate professor with the Hangzhou Vocational and Technical College. He is engaged in the deep learning, image processing, especially the research and application of image style migration, image restoration, 3D modeling, and rendering.

ZHIZHONG WANG is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University. His research interests include computer vision and deep learning.



HAIBO CHEN is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University. His research interests include computer vision and deep learning.

XIAOLONG MA received the Ph.D. degree in management science and engineering from the Shanghai University of Finance and Economics, China, in 2016. He is currently an Associate Professor with the School of Economic Management, Huzhou University, Zhejiang, China. His research interests include image processing, network mechanism design, and data mining.

WEI XING received the Ph.D. degree from Zhejiang University, in 2009. He is currently an Associate Professor with the College of Computer Science and Technology, Zhejiang University. His research interests include computer vision, image processing, and deep learning.

LEI ZHAO received the Ph.D. degree from Zhejiang University, in 2009. Then he did two years postdoctoral training at the College of Computer Science and Technology, Zhejiang University. He is currently an Assistant Professor with Zhejiang University. His research interests include computer graphics, computer vision, image processing, and deep learning.

ZHIJIE LIN received the degree from Hangzhou University of Electronic Science and Technology, and the master's and Ph.D. degrees from Zhejiang University. He is currently working with the Hangzhou University of science and technology. He is engaged in the research and application of deep learning, machine learning, and image processing.