Hindawi
Security and Communication Networks
Volume 2022, Article ID 4748946, 12 pages
https://doi.org/10.1155/2022/4748946

Research Article

Web-Cloud Collaborative Mobile Online 3D Rendering System

Chang Liu[1], **Huilin Song**[2], **Ting Fang**[1], **Qiaofeng Ou**[1], **Geng Yu**[1], **Tao You**[1], **and Ming Ying**[1]

[1]*School of Information Engineering, Nanchang Hangkong University, Nanchang, China*
[2]*School of International Economics and Trade, Jiangxi University of Finance and Economics, Nanchang, China*

Correspondence should be addressed to Chang Liu; lcsszz83@gmail.com

Received 12 July 2022; Accepted 13 September 2022; Published 27 September 2022

Academic Editor: Yuanlong Cao

The collaborative online 3D rendering system proposed in this paper ensures the quality of user experience and protects online rendering resources. In this system, the conditional generative adversarial network is used to calculate complex global illumination information instead of rendering them on cloud servers. The web front-end generates high-frequency direct lighting information in real-time and displays the final result which is a blend of front-end direct lighting information and back-end indirect lighting information. Experiments show that our proposed system can improve the rendering quality of the Web3D front-end, ensure Web-Cloud load balance, and protect rendering resources online.

1. Introduction

In the era of the Internet+, with the development of Web3D technology, more and more users are accustomed to the flexibility of experiencing 3D content on various portable devices, such as mobile phones, laptops, and head-mounted devices. Web3D technology, which puts 3D content on Web browsers, is supported by most mobile devices. This technology has a revolutionary impact on the new generation of Web services and produces various critical applications in the smart city, virtual tourism, virtual museums, e-commerce, etc.

The advantages of Web3D are excellent cross-platform, but its defect is limited rendering power. Web3D system rendering capabilities are mainly determined by its core graphics application programming interface (API) "WebGL" and hardware configuration. The latest WebGL 2.0 technology version is based on OpenGL ES 3.0 designed for embedded devices. Therefore, it is difficult to achieve the same rendering performance as on the personal computer with high-level graphics API "OpenGL." Besides, the loading latency of three-dimensional (3D) model data on the Web also poses a significant challenge, resulting in a long waiting time, which significantly reduces the user's quality of experience (QoE). In addition, the 3D model resources in the Web3D application are directly transmitted to the Web front-end, which brings the risk of resource leakage.

For this, the collaborative rendering system has emerged to split the complex task of rendering the scene between the cloud server and the Web client. As a collaborative rendering system, our CloudBaking [1, 2] is a dedicated Web3D application-oriented dynamic scene lighting and shadow rendering system intended to compensate for the inadequate rendering capability of the Web3D application. Therefore, we design this system to perform collaborative lighting and shadow rendering at both the client and server for the Web3D scene. This system assigns the high-complexity lighting and shadow rendering to the cloud server, including soft shadow, global illumination, and so on. The web client performs the task of low-complexity renderings, such as direct lighting and screen-space ambient occlusion. Therefore, the high-precision 3D scene model is safely placed in the cloud, while the Web front-end only needs a lightweight and encrypted low-precision scene model, which reduces the risk of resource leakage.

2. Related Work

2.1. Web3D Technology. In 1997, the virtual reality markup language (VRML) was officially released as an international standard for Web3D, making it possible for 3D model files [3] to be transferred over the Internet. In August 2004, the X3D specification was released as an international standard for Web3D. X3D integrated technologies such as XML, Java, and streaming at that time in the hopes of increasing processing power, rendering quality, and speed of transmission. The Khronos Group released the WebGL 1.0 specification in March 2011. WebGL 1.0 is based on OpenGL ES 2.0 and provides APIs for 3D graphics. The WebGL 2.0 specification was released in 2013, and, based on OpenGL ES 3.0, was supported for the first time in major Web browsers such as Firefox, Chrome, and Opera [4].

In recent years, WebGL is used as a graphics engine in many Web3D applications. Furthermore, many companies have developed their own advanced rendering engines based on WebGL, such as three. js and Babylon. [4]. Among these, three. js is a 3D graphics real-time rendering library based on JavaScript and WebGL. It has become gradually favored by the majority of users because of its efficient and plug-in-free Web-side rendering capabilities. Many Web3D rendering systems use three. js, such as the Web page visualization system proposed by Marion and Jomier [5], and the real-time visualization and segmentation system of real-time visual medical images developed by Jacinto et al. [6]. Similarly, our system also uses the three. js engine.

2.2. Web3D System. Web3D systems have been widely used in people's daily life for their excellent cross-platform and easy deployment. In a Web3D system, data are transformed into visual 3D models and presented on the Web, stimulating people's interest. Virtual heritage (VH), which displays the digitization of culturally historical artifacts for display on the Web browser, is a very typical use of Web3D. Currently, website presentations cannot satisfy the extensive and intensive experience of VH users. The VH websites provide narrative knowledge, annotation experience, and mobile environment experience to adapt to the changes [7]. Building information models (BIMs) have recently become the mainstream visualizing data in the building field. To display such data with high volume, variety, velocity, and value attributes on Web browsers, researchers have created an online Web3D system based on semantic analysis and light-weighting technology [8].

Web3D technology has been widely used to build many educational virtual environments (EVEs). In the beginning, EVEs were used to create visual immersion-based virtual scene display cases, such as the Webtop system [9]. Besides, the researchers of EVEs are also concerned about user engagement, interaction, and collaboration, such as Web3D-based surgical training simulators for the treatment of trigeminal neuralgia [10], virtual space-time environments online [11], and virtual war online [12]. Users of the Web3D system can experience immersion in education, training, and tourism without leaving home at a low cost.

2.3. Web3D Rendering System. A real-time rendering system is the critical subsystem of a Web3D system, responsible for the 3D scene's loading and rendering. We classify Web3D rendering systems into three categories based on which "side" the rendering task occurs.

2.3.1. Local Rendering System. This system puts the main rendering tasks on the Web browser. The server is responsible for storing and transmitting the 3D scene's data without participating in rendering tasks. The VH system [7], virtual building system [8], and Webtop systems [9] mentioned above employ this kind of rendering system.

2.3.2. Remote Rendering System. This system puts the main rendering task on the servers and delivers the rendered results to the Web browser in the form of image steam. The web client is only used to display the rendering results without participating in rendering. This system is very suited for the Web3D application with high-quality rendering performance demand [4]. By avoiding the direct loading of 3D models on web browsers, the system can protect 3D models from illegal downloading by users [13].

2.3.3. Hybrid Rendering System. This system combines the two systems described above and allows the Web-Client side and cloud-server side to render collaboratively. Such a system avoids the waste of front-end rendering resources and guarantees the execution of expensive rendering tasks. This kind of system is widely used to study lighting rendering in dynamic scenes.

The Cloud Light system first proposed allocating lighting rendering tasks [14]. Remote Asynchronous Indirect Lighting (RADL) implements viewpoint-independent collaborative lighting rendering [15]. Shading Atlas Streaming (SAS) [16] puts all shading calculations in the cloud to complete and uses the Shading Atlas mechanism of Virtual Texture for storage. Also, the Cloud Baking system (CB system) proposed by ourselves is a typical hybrid rendering system [1, 2].

2.4. Generative Adversarial Networks. The GAN represents a deep learning model based on two sets of the neural network, generator, and discriminator, for game-based mutual learning to generate the desired output. Since the suggestion of Goodfellow et al. was proposed [17], GAN has achieved remarkable results in the areas of "image to image" translation [18, 19], style transfer [20], super-resolution [21], and 3D model generation [22, 23], etc. The research into the image-to-image translation field provides us with a direct motivation to replace the image-based pre-rendering mechanism and the real-time rendering at the cloud server with GAN. In doing so, the images generated by the cloud server will be saved in the cloud server for training the relevant GAN model, and the generative model ends up being sent to the web client to generate the global illumination map (GI map).

3. System Architecture

Based on our CloudBaking (CB) [2], we present an intelligent remote rendering system called the Intelligent CloudBaking (ICB). For completeness, we offer the whole pipeline and workflow of the ICB system and focus on the ICB system's advancement compared to the CB system. As the CB system, the ICB system also consists of two separate rendering systems: the Web-Client system and the Cloud-Server system. The protocol for connecting the two systems is also WebSocket, as shown in Figure 1.

Similar to the CB system, our system contains two modules: cloud server and client. But we newly proposed a GAN-based pre-rendering module, which transfers the pre-rendering task from the CRT-buffers manager to the GAN. This solves the problem of cloud storage space limitation, and the images stored in CRT-buffers provide a large amount of input data for GAN training. The 3D Warping technology based on the prediction mechanism is used to eliminate the hole artifacts of the generated GI map. The original 3D scene is preprocessed into LMP and then progressively streamed to the client to generate DI-map. Finally, the mixed image of DI-map and GI map is presented to the user on the client. In addition, the resources obtained by the cloud of this system, such as Light-Weight 3D Scene Steam, encoded GI-map steam, and GAN for GI maps, have been lightweight or encoded, which further improves the protection of rendering resources.

3.1. Cloud-Server Subsystem. Like the CB system, ICB lightweight the original 3D scene for reducing the initial loading time [24] and progressively streams the lightweight version to the Web-Client for rendering. To reduce the initial loading time, each 3D scene is preprocessed into LPM [24] and stored on the rendering server. The rendering server progressively streams the LPM version to the client renderer for rendering upon request from the client.

However, the cloud-sever no longer generates the GI maps only like the CB but renders a cloud rendering texture buffer (CRT-buffer) for the current scene. As shown in Table 1, the CRT-buffer contains a group of images rendered under the current viewing frustum, which can be divided into two categories, including: (1) images whose pixel values store the rendered scene with lighting and shadow information (e.g., GI map, albedo map, and direct lighting map), called L&S-images and (2) images whose pixel values record the rendered scene geometry information (e.g., the depth map and normal map), called G-images. When the CRT-buffer has been rendered in Cloud-Server, we design an octree-based CRT-buffers manager to store them rather than discard them as in the earlier system. These images stored in Cloud-Sever can be used as input data to pre-render the GI map (e.g., when the CRT-buffers store the GI maps under a similar view), or to train the GAN for GI maps offline (e.g., when the CRT-buffer manager store the input data enough). After storing many images in the CRT-buffer manager of Cloud-Server, our system pre-renders the GI map by CRT-buffers manager searching first instead of rendering it.

3.2. Web-Client Subsystem. The web client with good cross-platform and extensibility is the primary interaction medium. However, despite the continued improvement to external equipment performance that carries the Web3D applications, a compromise on the rendering capability remains necessary for the Web3D technique to embed on the Web. Therefore, to enhance the web client's interactivity while reducing the pressure from rendering, we restricted the task of direct lighting rendering to the web client and branded the results after rendering it as a direct-lighting map (DL map). Many GI maps either derived from rendering at the cloud server or generated by the neural network at the web client will be sent here at the time of scene editing by the user at the web client. This is achieved in the following steps: (1) The web client checks out whether there is a generative network locally. (2) If the generative network does not exist, then send GI map request to the cloud server and await rendered GI map transmitted from the cloud server. (3) Otherwise, GI map of the current viewpoint should be generated by the generative model at the web client. Meanwhile, the input images of the generated model need to be rendered on the Web client, and we call these maps the WRT-buffer (as shown in Table 2).

Both approaches to GI map generation will contribute to interactive latency, with the only difference being that the latency would occur at different phases. For the former one, it would occur during the stages, including the cloud server rendering stage, GI map encoding stage, and transmission stage. The latter one would happen when the GI map is generated by the generative model at the web client. Limited by the interaction delay, the rendering system with viewpoint correlation is incapable of ensuring consistency between the viewpoint in the rendered DL map and that in the generated GI map, distortion is bound to arise from blending the two maps at the web client. In the process of prediction, a camera with a larger range of field-of-view (FOV) than the web client was chosen for the cloud server, for which the GI map information rendered in the cloud server could have a wider range than the DL map information rendered at the web client, and thus the predicted probability of artificial hole generated to GI map would be reduced. As revealed in the experiment, the higher the camera's FOV value for the training set of rendering, the better the quality of a map generated by the neural network. Finally, the blending GI map and DL map at the web client (weighed averaging for pixels) could derive the rendered frame as output for ultimate display.

4. Prefetch Strategy for 3D Warping

The 3D warping method is a technique of warping a reference image to an arbitrary viewpoint by projecting pixels on the reference image plane into the 3D space and reprojecting them to the target image plane (see the Resources [2] for more details). All 3D warping methods produce hole artifacts, and our method is no exception. The hole artifacts appear in our method because 3D models' vertex cannot find the texture coordinates on a pixel-missing reference image (GI map), as shown in Figure 2. Our method

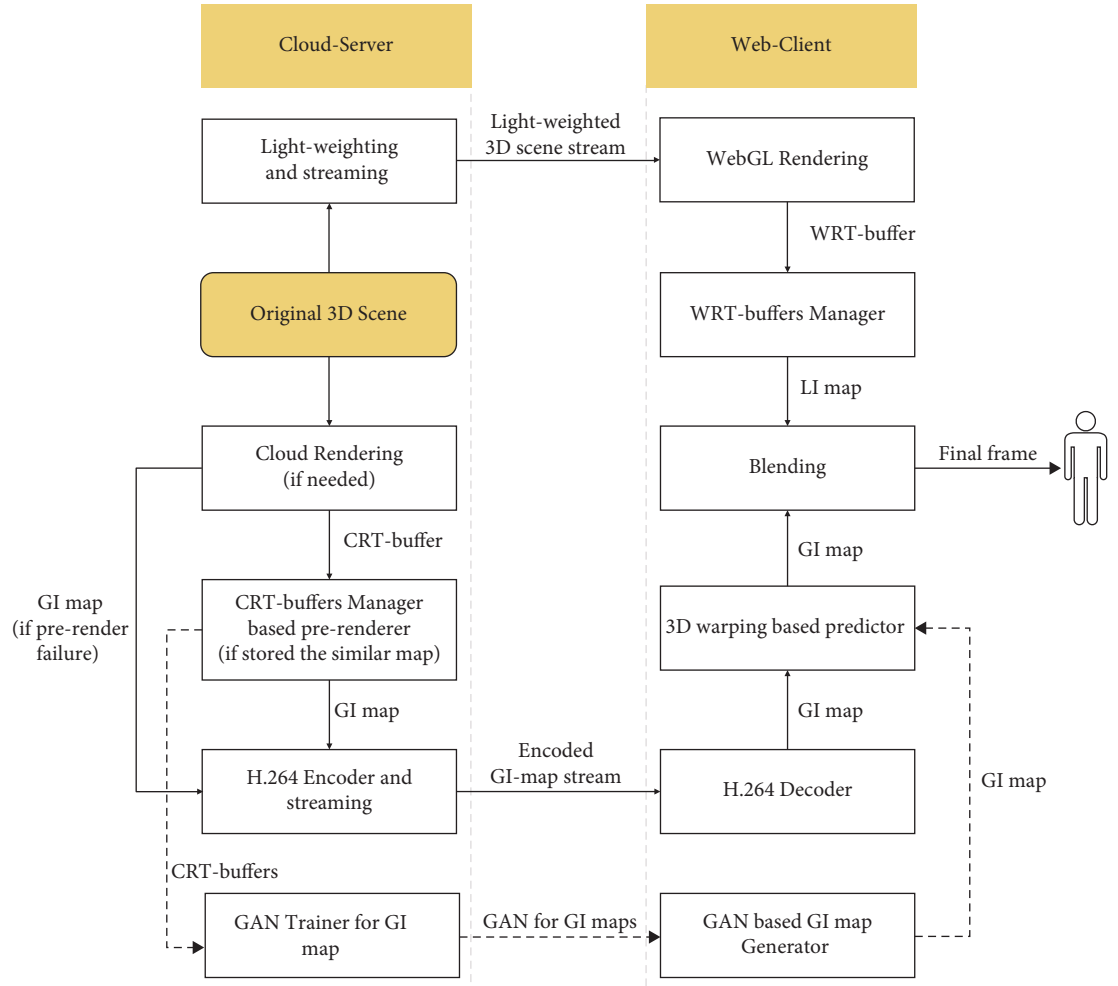

Figure 1: Intelligent Cloud Baking system architecture. Compared with the CB system, our system still contains two modules: cloud rendering and web front-end rendering, but the difference is that we propose a pre-rendering module based on GAN. The new module transfers the pre-rendering task from the CRT-buffers manager to GAN, which solves the problem of cloud storage space limitation and makes full use of the image data stored in CRT-buffers for training GAN.

Table 1: CRT-buffer struct.

Attribute names	Category	Data type
Direct lighting map	L&S-images	Texture
Albedo map		Texture
GI map		Texture
Normal map	G-images	Texture
Depth map		Texture
Direction	Camera information	Vector
Position		Vector

Table 2: WRT-buffer struct.

Attribute names	Category	Data type
Direct lighting map	L&S-images	Texture
Albedo map		Texture
Normal map	G-images	Texture
Depth map		Texture

of eliminating hole artifacts, the Prefetch method, supplements missing pixels by prefetching the new GI map generated by the predicted viewpoint.

In the field of Web3D technology, the camera simulates the human eyes and becomes the primary source of interactive data in the Web3D system. Our Prefetch method predicts the view frustum of the reference camera on the cloud-server side after a change in TIL (TIL is the length of interactive latency in our system). Assuming the network environment is stable, TIL is relatively fixed. We take six general directions, including forwarding, backward, left, right, up, and down, as the basic predicted camera movement directions. Within the TIL, our system calculates the camera view frustums after the movements. Then we set up a new camera whose frustum includes these view frustums, as shown in Figure 3.

First, after the web front-end camera is shifted, the cloud camera needs to include this range, and the cloud camera's view angle θ_t at any time t is shown as follows:

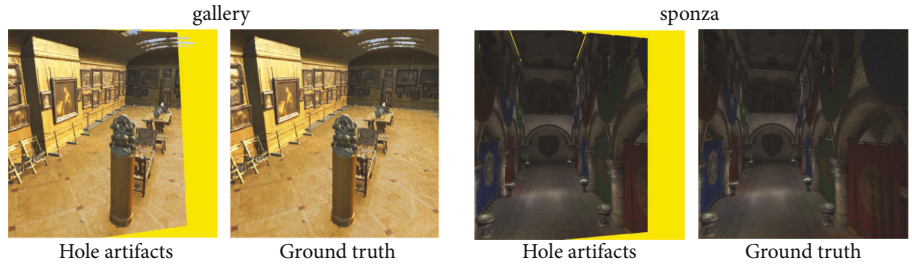

Figure 2: GI map with hole artifact VS. The ground truth. The image processed by 3D Warping has many hole artifacts due to the lack of pixel information, which is obviously different from the ground truth.

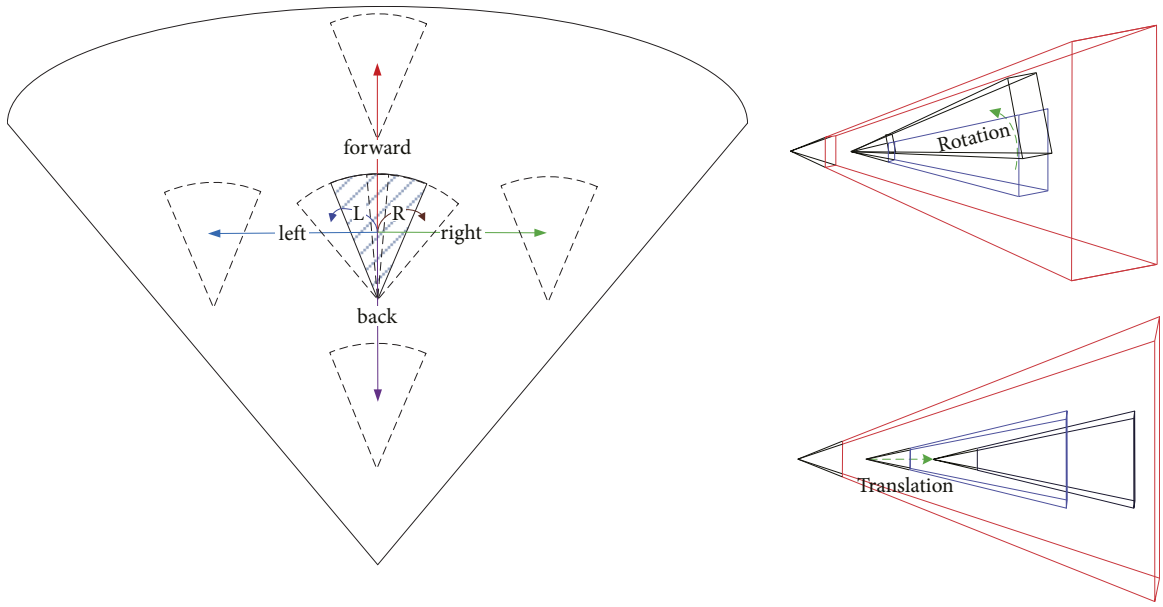

Figure 3: Optimization strategy for 3D warping. This strategy uses the image of the rendered frame to infer the motion vector of the next frame according to the corresponding position of the object in the scene in the pixel and predicts it according to the four directions of movement and the two directions of rotation in the motion vector, which can effectively improve the accuracy of prediction.

$$\theta_t = 2\pi \cot\left(\frac{\max\left\{\left|T_{\mathrm{IL}} * V_t.x\right|, \left|asp * T_{IL} * V_t.y\right|\right\}}{V_t}.z\right) + \alpha. \quad (1)$$

Vt (x, y, z) refers to the moving speed vector of the front-end viewpoint at any time t, asp refers to the aspect ratio of the cross-sectional view of the cloud camera. This formula first compares the moving distance's influence in the horizontal direction (x-direction) and the vertical direction (y-direction) on the view angle, then gets the most significant value and converts it into such an angle. In the same way, after the web front-end camera is rotated, the new view angle of the cloud camera is shown as follows:

$$\theta_r = \max\left\{\alpha + T_{\mathrm{IL}} * V_r.x,\ \arctan\left(\tan\left(T_{\mathrm{IL}} * V_r.y + \alpha\right) * asp\right)\right\}. \quad (2)$$

Vr (x, y, z) is the Web front-end viewpoint's rotation speed. However, under the influence of translation and rotation, the new view angle generated by the camera view volume at the back end of the cloud does not need to be superimposed simultaneously. It is only necessary to use the largest of the two as the new viewing volume angle. The maximum cannot exceed π, as shown in the following formula:

$$\theta = \min\left\{\pi, \max\left\{\theta_t, \theta_r\right\}\right\}. \quad (3)$$

The orientation of the viewpoint remains the same, and the position of the viewpoint moves in the opposite direction of the orientation, mainly to place the cloud viewpoint behind the front viewpoint, as shown in the following formula:

$$\begin{cases} P_i.x = P_o.x, \\ P_i.y = P_o.y, \\ P_i.z = P_o.z + V_t.z. \end{cases} \quad (4)$$

Pi (x, y, z) refers to the position of the cloud viewpoint, and Po (x, y, z) refers to the position of the front-end viewpoint.

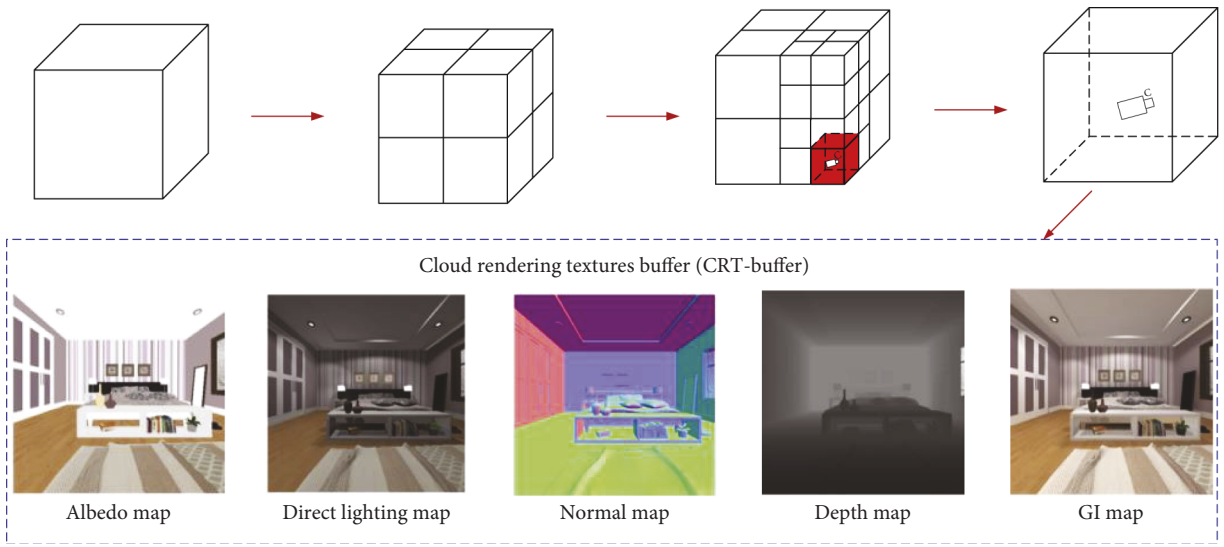

Figure 4: Octree-based CRT-buffers manager. This management method has a positive effect on data access in large-scale scene datasets, using the octagonal tree structure to manage the data in the entire scene, divide the scene data, and store the CRT-buffer data (Albedo maps, Direct lighting map, Normal map, Depth map, and Gl map) in the scene to each leaf node, which can effectively improve the system operation efficiency.

5. GAN-based Pre-renderer

For GAN, the management of large-scale data sets for training is very critical. Therefore, we built the CRT-buffer manager, which is an octree-based cloud server data manager, as shown in Figure 4. The nodes of this tree contain a set of viewpoint information in the rendered scene and the image information rendered on these viewpoints, as shown in Table 3. The depth of our octree is determined by the scale of the scene, and we store information in leaf nodes.

Before the construction of GAN, the pre-rendering task mainly relied on the CRT-buffer manager, which was realized on the assumption that the light source information in the rendered scene did not change. The pre-rendering steps are as follows: (1) Under the assumption that the conditions are established, the system checks whether there is a leaf node in the octree of the manager through the position of the camera. (2) If the leaf node does not exist, our system will directly request the cloud to render the GI map. (3) Otherwise, the system will compare the current camera information with all the camera information in the leaf node one by one. (4) During the comparison process, if it is found that there is a camera "similar" to the current camera's position and orientation in the leaf node, then directly take out the GI map corresponding to the camera's position and send it to the Web front-end. Otherwise, the system will still request the cloud to render the GI map. If the dot product of the positions and directions in the two cameras are all below threshold α ($\alpha < 0.1$), we judge that the two cameras are "similar".

With the accumulation of data in the CRT-buffer manager, the limitation of cloud storage space has become a bottleneck problem for pre-rendering based on the CRT-buffer manager. Therefore, we built a GAN-based Pre-Renderer mechanism, which uses the image generation capabilities of GAN to pre-render, instead of pre-rendering based on the CRT-buffer manager. Meanwhile, a large number of images stored in the CRT-buffer manager provide input data for the construction of GAN.

Table 3: Node struct of octree in CRT-buffer manager.

Attribute names	Category	Data type
CRT-buffers	L&S-images and camera information	CRT-buffer
Adjacent		CRT-buffer
Position	Node information	Vector
Index		Int

Our GAN-based Pre-Renderer is derived from our proposed GIGAN system for the rendering of human organs [25]. As shown in Figure 5, the GAN-based Pre-Renderer consists of a Web client and a cloud server and employs WebSocket for network communication. The cloud server trains a conditional generative adversarial network (GAN) to generate a global illumination map (GI map) using a CRT-buffer rendered by the cloud renderer. After training the network, our pre-rendering system sends the generated model to the Web client and then uses it to generate GI maps on the Web client in real-time. Like the GIGAN system, the GAN generator network in our pre-rendering system uses the traditional U-net-based encoder-decoder framework [26]. The discriminator network uses the Markovian patch GAN structure [20]. Their input data is the image dataset in our CRT-buffer, and we prove the validity of these data in the subsequent chapter. In addition, multipath TCP (MPTCP) is considered to be the most potential transmission mechanism to meet the specific requirements of

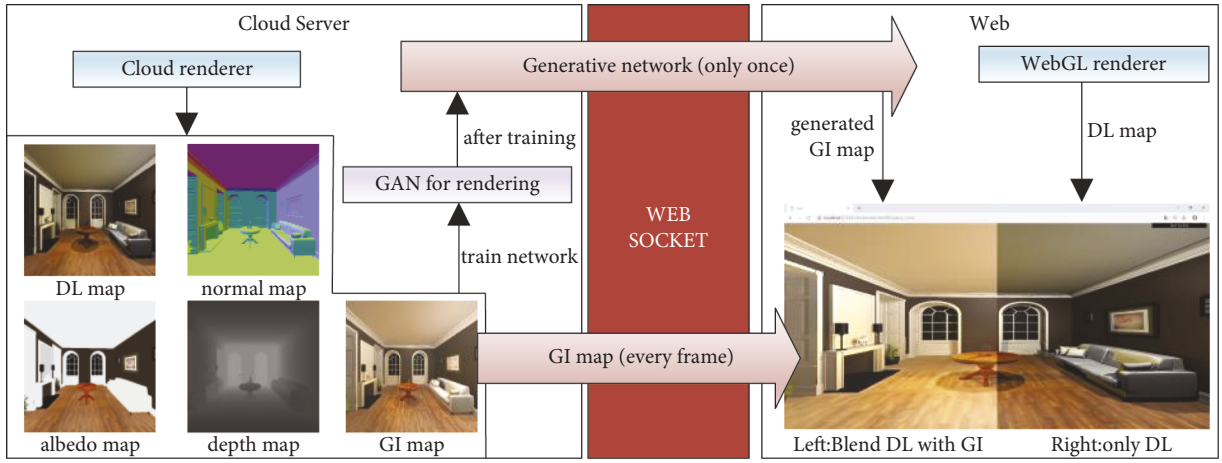

Figure 5: GAN-based Pre-Renderer mechanism. The cloud server transmits the generative model to the web client only once, which changes the previous model in which the cloud server needs to render the GI map and transmit it to the web client every frame. Then the web client can generate the GI map locally in real-time.

multimedia transmission in a multi-homed wireless network environment, which is the main reference for the future transmission mechanism of our system [27].

However, the scene rendered by our system is universal, and the complexity and scale of the 3D models in the scene exceed that of the human organ models rendered in the GIGAN system. Therefore, we optimized GAN to improve the quality of generated images and shorten the training time. The loss function of our GAN is shown as follows:

$$L = L_a + L_c + L_p. \tag{5}$$

Similar to the loss function of GIGAN, we employ the adversarial loss function (La) based on the conditional Wasserstein GAN [28] with gradient penalty to measure the basic information's difference between generated image and ground truth in adversarial processing, as shown in the following formula:

$$\begin{aligned} L_a(G,D) = {} & E_{c,y\ p_{\text{data}}(dg,\ y)}[D(c,y)] \\ & - E_{c\,P_{\text{data}}(c),\ z\ P_z(z)}[D(c,G(z))] \\ & + \lambda_{\text{GP}} E_{c\ P_{\text{data}}(dg),\bar{y}\ p_{\text{GP}}}. \end{aligned} \tag{6}$$

In this formula, we refer to the algorithm of WGAN-GP. G is a generator, D is a discriminator, z~pz (z) is a random noise from a certain distribution (such as normal distribution and uniform distribution), c, y~pdata (c, y) are images, respectively, from the source domain and the corresponding target domain, $\bar{y}$~pgp represents the distribution after linear interpolation between the real data distribution and the generated data distribution. λ GP is a hyperparameter. K represents the expected close value of the gradient during training, and $k = 1$ here.

And we use the contention loss (Lc) to measure the difference of each pixel between the generated image and the ground truth (for details, please refer to the literature [25]), as shown in the following formula:

$$L_c(G) = E_{c,y\sim p_{\text{data}}(c,y)}[\|y - G(c,z)\|_1]. \tag{7}$$

In addition, we added a new perceptual loss function Lp to the original loss function. We employ the perceptual loss (Lp) to measure the contextual and structural information between generated image and ground truth and apply a pre-trained VGG19 network [29] to achieve this. The perceptual loss can be formulated as follows:

$$L_P(y,\hat{y}) = \frac{1}{C_i H_i W_i} E_{y\sim p_{data},\hat{y}\sim p_{gp}}\left[\left\|\theta_i(y) - \theta_i(\hat{y})\right\|_2\right]. \tag{8}$$

In formula 8, C_i, H_i, and W_i, respectively, represent the channel number, width and height of the image features. $\theta_i(y)$ indicates the i-th layer of the VGG19 network (after activation). The new loss function plays an effective role in ensuring the quality of the output GI map.

As shown in Figure 6, the image data set in the CRT-buffer is collected as the input of the generator with skip connections and then passes 5 downsampling layers and 5 upsampling layers before generating the GI map. LeakyReLu is used as the activation function in the entire downsampling process, while ReLu and tanh (the last layer only) are used as the activation function in the upsampling process. The discriminator is composed of 4 encoders and uses LeakyReLU as the activation function. During training, the network randomly reads data from the data set in batch size to 4, and G-D alternately uses mini-batch stochastic gradient descent and Adam optimizer with learning rate = 0.0001 to update the weight of the network. Compared with GIGAN, the GAN training time of our pre-rendering system is shortened by 20%–30%.

6. Experimental Results and Analysis

The test environment of our system is as follows: Our cloud server is equipped with two Intel Xeon Silver 4114 2.2 GHz CPUs, one Nvidia Quadro P5000 GPU, and 128 GB of RAM, and the server is running Windows Server 2012. For the web client, we use a laptop with an Intel Core i7-7700HQ 2.8 GHz CPU, an Nvidia GeForce GTX1060 M GPU, and

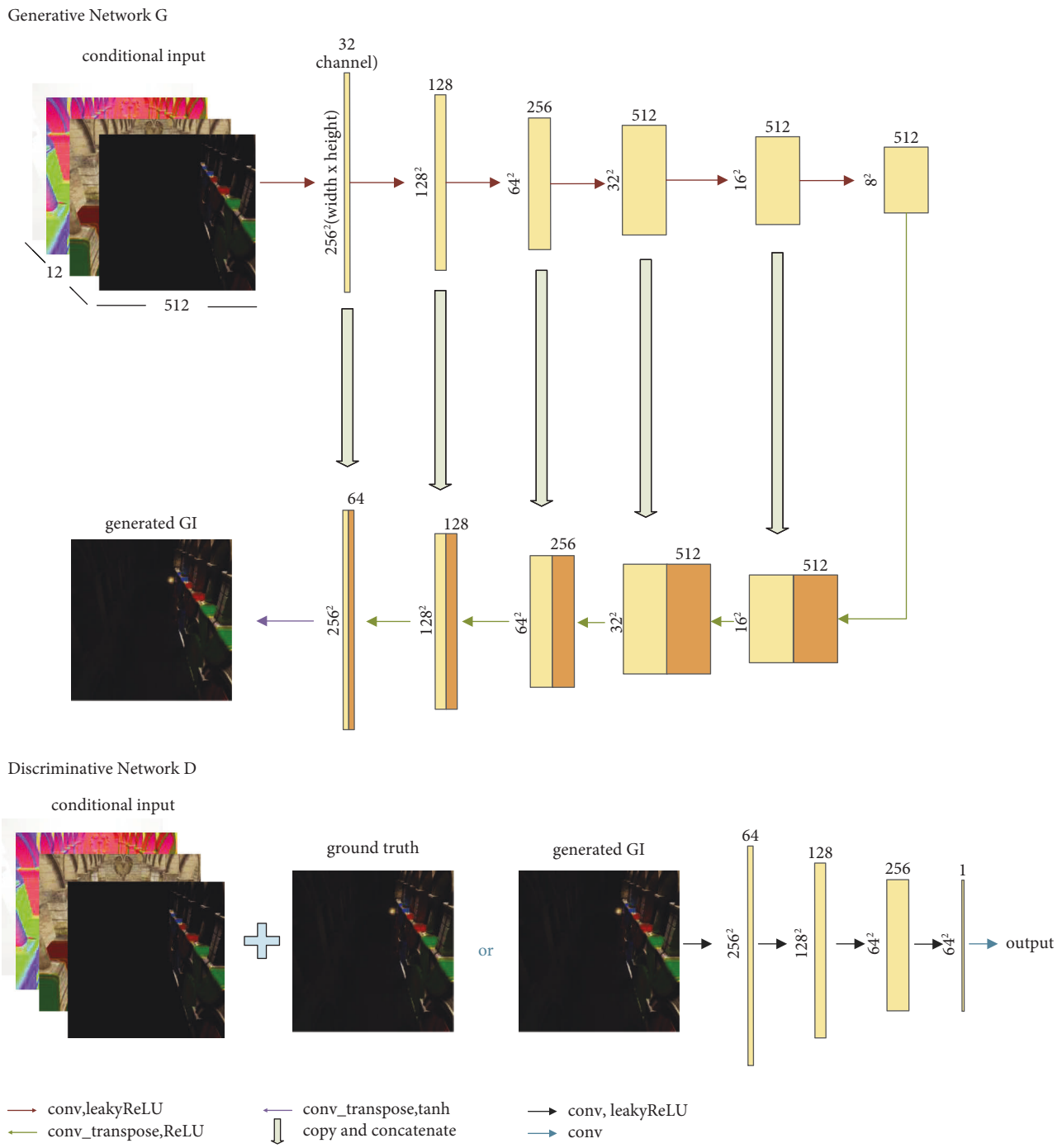

Figure 6: Overall conditional generative adversarial network architecture. The generator network adopts a 5-layer U-NET structure, in which the convolution and deconvolution operations can be regarded as the process of encoding and decoding. And the discriminant network adopts a 4-layer full convolutional network.

8 GB of RAM. The laptop runs Windows 10 and uses Google Chrome version 71 as a web browser.

For verifying the rationality of the input of our GAN and the effectiveness of this system, we conducted the following test: (1) First, we enumerate a combination of multiple types of image data as the input data of the generative confrontation network. (2) In the cloud back-end, we generate various GANs based on these different sets of input data. (3) We pass these GANs to the Web front-end and generate new GI maps based on them on this end. (4) Finally, we test the image quality of these GI maps generated by different GANs and compare them.

Direct lighting information is part of global illumination, and the albedo is directly involved in the calculation of global illumination. Therefore, we use the DL map and albedo map data storing these two information as the most

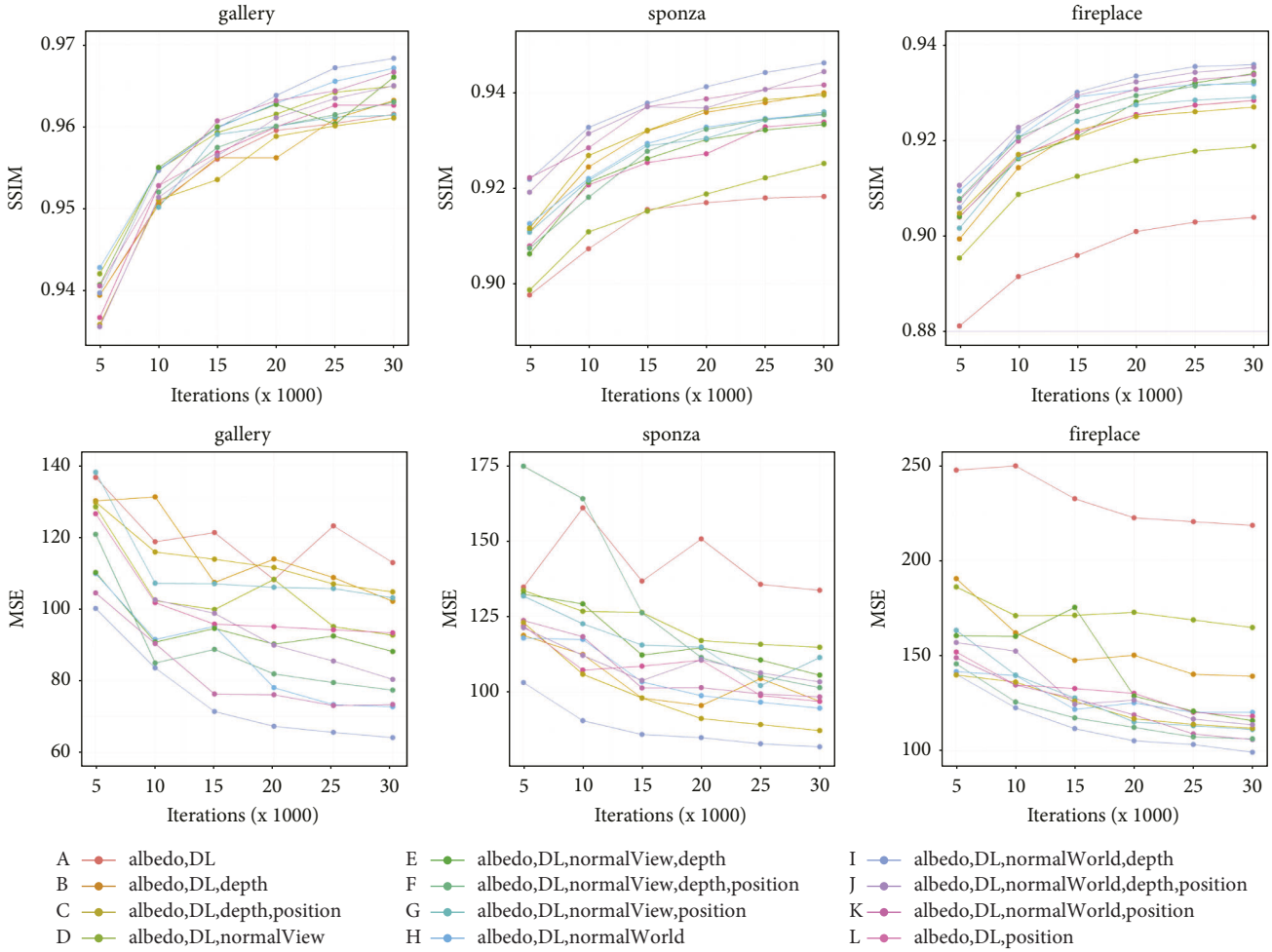

Figure 7: Network iterations VS. SSIM and MSE in different input configurations (test scene is sponza). We can compare the difference between GI map generated by different GANs and Ground truth under the same number of iterations.

important GAN input data to generate the final GI map. In addition, we also selected position map, depth map, normalWorld map (normal map in world space), and noramlView map (normal map in view space) from the G-buffers data. We take the random combination of the pictures elected in G-buffers and the previous two pictures as GAN's input configuration, as shown in the dotted box in Figure 7. In the end, we get multiple sets of generative adversarial network models.

We judge the pros and cons of our GANs' model by testing the quality of the final generated GI map. This paper uses structural similarity (SSIM) and mean square error (MSE) to evaluate the similarity between the generated image and the real image. The former measures the similarity by comparing the brightness, contrast, and structure of the two images, while the latter measures it from the error between the corresponding pixels of the two images. Zinner et al. proposed that the image quality is acceptable when SSIM is greater than 0.88 [22], so our paper uses 0.88 as the image quality threshold and sets both SSIM and MSE to be calculated in the image's RGB color space (range 0–255). In addition, we make the cameras in the test set move along a different path from the training set to test the generalization of the model. In Figure 7, different colors represent different input configurations. The horizontal axis represents the number of iterations of the network, and the vertical axis represents the SSIM (higher is better) or the MSE (lower is better) between the generated image and the real image.

As shown in Figure 7, most GANs obtained after a small number of training iterations can make the generated image's quality exceed the basic threshold. We judge the direct illumination information occupies a higher ratio in the global illumination image, which enables GAN to quickly fit and generate a high-quality GI map, and all our input configurations include this information. After 20,000 training iterations, the changes of SSIM and MSE between the real image and the image generated by most GANs have stabilized. Therefore, we consider 20,000 times is the ideal threshold for our GAN training times.

In addition, based on the viewpoint correlation of our system, we use the normal map of the view space as the input data of GAN. But the experimental results prove that using the normal map of world space as the input data of GAN can make the final generated image quality higher, which means

Figure 8: Generated GI map VS. Real GI map. The difference between the differential GI map and the Real GI map pixel-by-pixel difference operation can be more intuitively recognized. The darker the color, the smaller the difference between the corresponding pixels These results show that the generated GI maps are very close to the real GI maps.

that the normal map of world space has more effective information than the normal map of view space. Note that the data between the three pairs of different input configurations in Figure 7 illustrate the above results, including *E* and *H*, *F* and *I*, *G* and *J*.

Finally, we found that among all input configurations, the generated image obtained by the GAN obtained by the I^{th} input configuration (albedo, DL, normal, and depth) has the highest SSIM and the lowest MSE value, and the entire training process is relatively stable. Therefore, we use the I-th input configuration, the final results are shown in Figure 8. These results show that the GI maps generated by this GAN are very close to the real GI maps.

7. Conclusion

This paper attempts to combine cloud rendering technology with artificial intelligence technology. We propose a complete architecture of a smart cloud rendering system for Web3D based on the CloudBaking system and GAN. Our system uses a trained neural network to generate rendered images and eventually partially replaces the hardware's rendering capabilities. Experimental results prove that the GAN-based intelligent rendering system for Web3D can complete rendering tasks while saving and protecting rendering resources.

Although the use of GAN effectively improves the speed of real-time rendering, neural network cannot completely solve the inevitable delay. In order to reduce latency, the architectural optimization of cloud rendering systems is usually considered. At present, 5G network and edge computing technology are fully utilized to optimize the architecture of cloud rendering system, which reduces the interaction delay of the system. In addition, the training process relies on the accumulation of a large number of pre-rendered images, which is also a challenge to storage space.

And the use of different neural network structures has a crucial impact on the result. These are the areas that can be improved in our future work.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was part of the research supported by the Science and Technology Program of Educational Commission of Jiangxi Province, China (DA202104172), the Innovation and Entrepreneurship Course Program of Nanchang Hangkong University (KCPY1910), and the Teaching Reform Research Program of Nanchang Hangkong University (JY21040).

References

[1] C. Liu, J. Jia, Q. Zhang, and L. Zhao, "Lightweight websim rendering framework based on cloud-baking," in *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 221–229, Singapore, May 2017.

[2] C. Liu, W. T. Ooi, J. Jia, and L. Zhao, "Cloud baking: collaborative scene illumination for dynamic Web3D scenes," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 3, pp. 1–20, 2018.

[3] L. Chittaro and R. Ranon, "Web3D technologies in learning, education and training: Web3D technologies in learning, education and training: Motivations, issues, opportunitiesotivations, issues, opportunities," *Computers & Education*, vol. 49, no. 1, pp. 3–18, 2007.

[4] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat, "3D graphics on the web: 3D graphics on the web: A survey survey," *Computers & Graphics*, vol. 41, pp. 43–61, 2014.

[5] C. Marion and J. Jomier, "Real-time collaborative scientific WebGL visualization with WebSocket," in *Proceedings of the 17th International Conference on 3D Web Technology*, pp. 47–50, California, CL, USA, July 2012.

[6] H. Jacinto, R. Kéchichian, M. Desvignes, R. Prost, and S. Valette, "A web interface for 3D visualization and interactive segmentation of medical images," in *Proceedings of the 17th International Conference on 3D Web Technology*, pp. 51–58, California, CL, USA, Auguest 2012.

[7] H. Rahaman and B. K. Tan, "Interpreting digital heritage: a conceptual model with end-users' perspective," *International Journal of Architectural Computing*, vol. 9, no. 1, pp. 99–113, 2011.

[8] X. Liu, N. Xie, K. Tang, and J. Jia, "Lightweighting for Web3D visualization of large-scale BIM scenes in real-time," *Graphical Models*, vol. 88, pp. 40–56, 2016.

[9] T. Mzoughi, S. D. Herring, J. T. Foley, M. J. Morris, and P. J. Gilbert, "WebTOP: a 3D interactive system for teaching and learning optics," *Computers & Education*, vol. 49, no. 1, pp. 110–129, 2007.

[10] Y. Li, K. Brodlie, and N. Phillips, "Web-based VR training simulator for percutaneous rhizotomy," in *Medicine Meets Virtual Reality 2000*, pp. 175–181, IOS Press, Amsterdam, Netherlands, 2000.

[11] V. Ramasundaram, S. Grunwald, A. Mangeot, N. B. Comerford, and C. Bliss, "Development of an environmental virtual field laboratory," *Computers & Education*, vol. 45, no. 1, pp. 21–34, 2005.

[12] C. Liu, J. Jia, Y. Ge, and N. Xie, "Web3D online virtual education platform for touring huangyangjie battlefield scenario over internet," in *Proceedings of the International Conference on Technologies for E-Learning and Digital Entertainment*, pp. 63–76, Springer, 2016.

[13] D. Koller, M. Turitzin, M. Levoy et al., "Protected interactive 3D graphics via remote rendering," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 695–703, 2004.

[14] C. Crassin, D. Luebke, M. Mara et al., "CloudLight: a system for amortizing indirect lighting in real-time rendering," *Journal of Computer Graphics Techniques*, vol. 4, no. 4, pp. 1–27, Hangzhou, China, April 2015.

[15] K. Bugeja, K. Debattista, and S. Spina, "An asynchronous method for cloud-based rendering," *The Visual Computer*, vol. 35, no. 12, pp. 1827–1840, 2019.

[16] J. H. Mueller, P. Voglreiter, M. Dokter et al., "Shading atlas streaming," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–16, 2018.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[18] P. Isola, J. Y. Zhu, T. Zhou, and A. Efros, "A Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, Honolulu, HI, USA, July 2017.

[19] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2849–2857, Venice, Italy, October 2017.

[20] C. Li and M. Wand, "Precomputed real-time texture synthesis with Markovian generative adversarial networks," in *Proceedings of the European Conference on Computer Vision*, pp. 702–716, Springer, Amsterdam, The Netherlands, October 2016.

[21] C. Ledig, L. Theis, F. Huszár et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, Honolulu, HI, USA, July 2017.

[22] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[23] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y. G. Jiang, "Pixel2mesh: generating 3d mesh models from single rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67, Munich, Germany, September 2018.

[24] L. Wen, J. Jia, and S. Liang, "LPM: lightweight progressive meshes towards smooth transmission of Web3D media over internet," in *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its*

Applications in Industry, pp. 95–103, Shenzhen China, November 2014.

[25] N. Xie, Y. Lu, and C. Liu, "Web3D client-enhanced global illumination via GAN for health visualization," *IEEE Access*, vol. 8, Article ID 13281, 2019.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, Munich, Germany, October 2015.

[27] Y. Cao, L. Zeng, Q. Liu, G. Lei, M. Huang, and H. Wang, "Receiver-assisted partial-reliable multimedia multipathing over multi-homed wireless networks," *IEEE Access*, vol. 7, Article ID 177689, 2019.

[28] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the International Conference on Machine Learning*, pp. 214–223, Sydney, Australia, August 2017.

[29] C. Ledig, L. Theis, F. Huszár et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, Honolulu, HI, USA, July 2017.