Received November 13, 2019, accepted December 9, 2019, date of publication December 24, 2019,
date of current version January 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962108

Web3D Client-Enhanced Global Illumination via GAN for Health Visualization

NING XIE[1], YIFAN LU[1], AND CHANG LIU[2]
[1]Center for Future Media, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]College of Information Engineering, Nanchang Hangkong University, Nanchang 330063, China

Corresponding author: Chang Liu (70202@nchu.edu.cn)

This work was supported in part by the National Nature Science Foundation of China under Grant 61602088, Grant 61572108, and Grant 61632007, and in part by the Fundamental Research Funds for the Central Universities under Grant Y03019023601008011.

ABSTRACT 3D visualization of digital human becomes a key tool for the medical visualization, especially for medical education. Web3D technology has been commonly applied in this field. However, the quality of rendering is not expected for the medical purpose. Nowadays, global illumination (GI) map is an efficient tool for real-time lighting and shadow rendering. On the cloud baking server, a large number of rendered GI maps are generated under variety of configuration in the scene on the Web3D interface end. GI tree works on organizing these baked maps for reusing in the Web3D client. Meanwhile, it dispatches the existing baked maps directly in the case that the viewpoint appears in the duplicate positions in the Web3D client. This is the main stream solution of the cloud pre-rendering. However, it is a challenge to store and manage excessive rendered maps. In this paper, we propose a light-weight collaborative machine learning method for lighting and shadow rendering in medical applications. In this system, the conditional generative adversarial networks (GAN) works for generating the GI map instead of finding out the similar from number of stored maps, and we propose structure-aware 3D image warping method to improve the system performance. The experiments demonstrated that our proposed system not only guarantees the resolution of the GI map in the Web3D client, but also significantly reduces the rendering computational needs so as to improve the system performance.

INDEX TERMS Generative adversarial network, Web3D, global illumination, medical data visualization, 3D image warping.

I. INTRODUCTION

Nowadays, 3D visualization of digital human becomes the key tool for the medical visualization and education. E-learning systems are increasingly supporting medical education and medical health. This system allows users to learn knowledge in an autonomous, time and location independent style [1]. Medical education is undergoing significant changes due to technological developments in the areas of 3D model rendering and computer-aided design.

The web client as a universal multimedia platform can be accessed easily. It exists several online E-learning systems for showing digital health by rendering 3D models, such as virtual liver [2] and anatomy teaching [3]. However, all these systems do not produce high quality visual effects. The 3D rendering task of the web client is mainly completed by Web3D. With the development of Web3D technology, the applications are emerging in various fields, such as medical education and training [4], video games [5], virtual reality [6] and so on. However, the rendering capability of Web3D client is rather limited, that is to say, the complex rendering task (i.e., global illumination) cannot work in the real time. For Web3D-based visualization of digital human, the users always have high rendering quality requirements. To address this issue, the cloud baking [7], [8] works as a collaborative system with dynamic scene illumination and shadow rendering. It divides the complex task of rendering within the scene between the cloud server and the Web3D client. The server renders global illumination (GI) map including soft shadow, indirect illumination, and ambient occlusion in real time. As the cloud baking runs, the server is in charge of GI maps rendering. Then, the generated GI maps will be transmitted to Web3D clients. These maps are organized by the sprite tree [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Wenbing Zhao.

It eliminates the need to repeat rendering by deploying the relevant images directly from the sprite tree when the Web3D client viewpoint pass to the same location repeatedly. It will definitely lead to a large occupation of storage space (see Table 2).

In C/S cloud rendering system, the interaction latency consists of the request transmission, the GI map rendering, GI map encoding, GI map decoding, 3D image warping, and blending phases. It causes poor performance on the Web3D client. Theoretically, the real-time global illumination rendering is practically a mapping from the buffer of the screen-space, for instance, the position map, normal map, reflection map, and other attributes in a virtual 3D scene, to various frames of screen effect. It can be treated as a typical task of image-to-image translation. GAN has achieved significant results in image-to-image translation field [10]. In this paper, we investigate the learning method based on GAN for the image-based pre-rendering so as to achieve the high resolution real-time Web3D rendering. In our system, the well pre-trained GAN is deployed on the Web3D client to visualize 3D human organs.

The main contributions of this paper are as follows:

1) The Web3D client-enhanced global illumination generation via GAN is proposed for Web3D applications. The performance of the network architecture is better than the existing state-of-the-art methods in terms of producing high quality effects on Web3D.
2) The proposed method enables to effectively reduce the interaction latency and storage space compared to cloud rendering.
3) The rapid structure-aware 3D warping method is proposed so as to reduce the inference frequency of GAN and decrease hole artifacts caused by inference latency.

II. RELATED WORKS

A. WEB3D REMOTE RENDERING SYSTEM

For real-time rendering system, Web3D remote rendering refers to the high complexity 3D rendering task placed on the remote cloud server by making full use of its powerful hardware rendering capabilities to quickly get the high quality rendering results. The remote rendering system is divided into two categories based on the participation of Web3D client in the entire rendering task.

1) **Server-end Remote Rendering Module.** It puts the rendering task completely on the server and transmits image frames and video streams to the Web3D client. Since the Web3D client is only used to receive and display results, it does not perform any rendering tasks, thus decreasing the hardware requirements of the Web3D client [11]. After the system assigns the rendering tasks to the cloud server, it only needs to store the modeling data on the cloud server and render it. This allows users to view the rendering results of these models only on the Web3D client. It helps to protect the modeling data, because the data can not be directly accessed [12]. Because this kind of system requires excessive computing power, the scalability can be severely limited. In addition, the inevitable interaction latency will have a significant impact on the quality of the user experience [11]. Shi *et al.* [13] proposed a high-quality low-latency remote rendering visualization system and a mobile web-based remote rendering visualization system proposed by Maamar [14].
2) **Collaborative Remote-Rendering Module.** It allows the Web3D client to participate in rendering. The rendering tasks are distributed based on not only the performance of the hardware devices of the Web3D client, but also the cloud server respectively. In practice, the rendering tasks with high computing intensity are assigned to the cloud server, while those with low computing intensity to the Web3D client. This could use the rendering capability of the Web3D client while reducing the rendering burden of the cloud server. This class of system has been widely applied to the illumination rendering in a dynamic scene. Crassin et al.'s CloudLight system firstly proposed the idea of sharing illumination rendering tasks [15]. The asynchronous method for cloud-based rendering suggested by Keith Bugeja enables a collaborative illumination rendering that is irrelevant to the viewpoint [16]. The cloud baking system proposed by Chang *et al.* [8] implements real-time collaborative global illumination rendering on the Web3D client. The Web3D client implements direct illumination rendering, while the cloud server performs global illumination rendering. We will make a full comparison with this system.

B. GENERATIVE ADVERSARIAL NETWORKS

The real-time global illumination rendering is practically a mapping from screen space buffer, for instance, the position map, normal map, reflection map, and other attributes in a virtual 3D scene, to various frame images of screen effect. Therefore, this is the typical task of image-to-image translation. For this task, the image is taken as input for translation into another representation of the scene. Besides, GAN has made remarkable achievements in the image-to-image translation field, including 'pix2pix' [10], generate photograph from sketch [17], [18], domain adaptation [19], image editing [20], [21], video prediction [22]. Neural Style Transfer [23] is another form of image-to-image translation that combines the style of one image with the content of another image to generate the final image. In addition, DualGAN [24], CycleGAN [25] and DiscoGAN [26] introduce the cycle consistency loss to deal with cross-domain image-to-image translation tasks. In fact, the deep learning-based global illumination rendering is present, such as CNN-based Deep Shadering [27], the conditional GAN-based Deep illumination [28]. This study will be expanded based on the networks as mentioned above.

C. 3D IMAGE WARPING

3D image warping is widely used in 3D video processing [29] and image-based rendering [30]. In real-time interactive rendering of 3D graphics, this method is often used to reduce interaction latency [8], [9], [13]. This method will produce the hole distortion when the information in the input image is not sufficient to produce the entire image at a new viewpoint. There are multiple solutions to solve this problem such as the layered depth images [31], the super visual distortion [32] and the double warping [13].

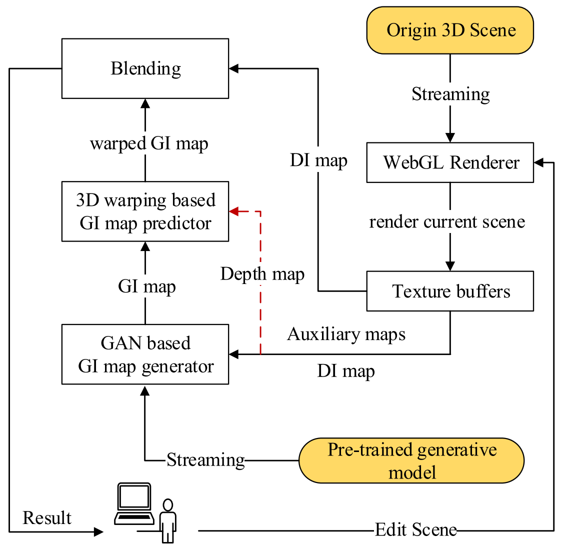


FIGURE 1. The architecture of our proposed system, GIGAN. The red dashed line allows the system to directly generate the warped image using the latest GI map by structure-aware warping method instead of generating the GI map via GAN.

III. OVERVIEW

We propose a new rendering AI system called GIGAN for the global illumination effect of the Web3D client via GAN. Figure 1 illustrates the whole workflow of our GIGAN system. When the system runs, the Web3D client loads the original 3D model and the pre-trained generative network. When the users edit the scene on the Web3D client, the client uses the WebGL renderer to generate the direct illumination (DI) map including diffuse and specular reflection effects at the current viewpoint v_{t-1}. Then, the Web3D client applies the pre-trained network to generate a GI map at the same viewpoint v_{t-1}. Moreover, to guarantee the same input images of generative network as the training phase, the WebGL renderer needs to render auxiliary texture buffers, such as albedo map, normal map and depth map. All of them help the network to generate the GI map that approximates the quality of offline rendering (see Sec.IV in detail). Finally, the system blends the DI map with the GI map (weighed averaging for every pixel, see Equation (1)).

$$P = \tau * P_{DI} + (1 - \tau) * P_{GI} \tag{1}$$

where P, P_{DI}, P_{GI} represent the corresponding pixel in the DI map, GI map and final output frame respectively. τ is the parameter and $\tau \in [0, 1]$.

The inevitable interaction latency still exists in GIGAN system. The Web3D client consumes 20-30 ms to generate one GI map, which has less time-consuming than the traditional global illumination real-time rendering method (such as voxel cone tracing [33]). However, the camera may have moved to another viewpoint v_t (different from viewpoint v_{t-1}) due to this interaction latency. Since the time of the DI map rendering is negligible compared to the time of GI map generation, the DI map may be at viewpoint v_t and the GI map is still at viewpoint v_{t-1}. Therefore, blending GI map and DI map from different viewpoints will inevitably produce the image ghosting distortion(see Figure 4(a)). To solve this issue, we propose the structure-aware 3D image warping method to predict a GI map at viewpoint v_t by using the latest generated GI map (see Sec.V). This method has three advantages as follows. Firstly, it can eliminate the inconsistency of blending. Secondly, it always maps the latest GI map pixels (in units of primitives) to the DL map at the current viewpoint v_t, so it allows us to use the different cameras' field of views (FOV) to render DL map and generate GI map respectively. Finally, we can set wider FOV to render the training dataset. This can guarantee that the GI map generated by GAN has more scene information than the DI map rendered by WebGL renderer, so we can make fullly use of enough information of one GI map in prediction phase (note that there are two sets of cameras in the scene. One is the local camera C_{local} for Web3D rendering. The other is the GI camera C_{GI} only for recording the viewpoint when generating the GI map and without rendering). The system predicts a GI map in every 30 frames instead of every frame in order to improve the whole system performance and reduce the predicted frequency of GAN (see Table 3). That is to say, we use the generative network to predict one GI map and then use structure-aware 3D image warping method in the $1st$ frame, but only use the structure-aware 3D image warping method in the remaining 29 frames. Therefore, the interaction latency in GIGAN system consists of the GI prediction, structure-aware 3D image warping, and blending phase, which has less latency compared to the cloud baking system (see Table1). Figure 2 shows the difference between only direct illumination by Web3D rendering and our GIGAN method (blending DI map rendered by Web3D and GI map generated by GAN) in the lung, human blood vessel, and digestive system scenes respectively.

IV. GENERATIVE ADVERSARIAL NETWORK

Both the rendering and image synthesis tasks have the same objective, but the way of processing is different. For the former one, it uses the rendering pipeline. For the latter, the values of the network-related parameter are calculated to generate the map. In respect to the proposed system, we apply GAN to generate a GI map on Web3D client instead of

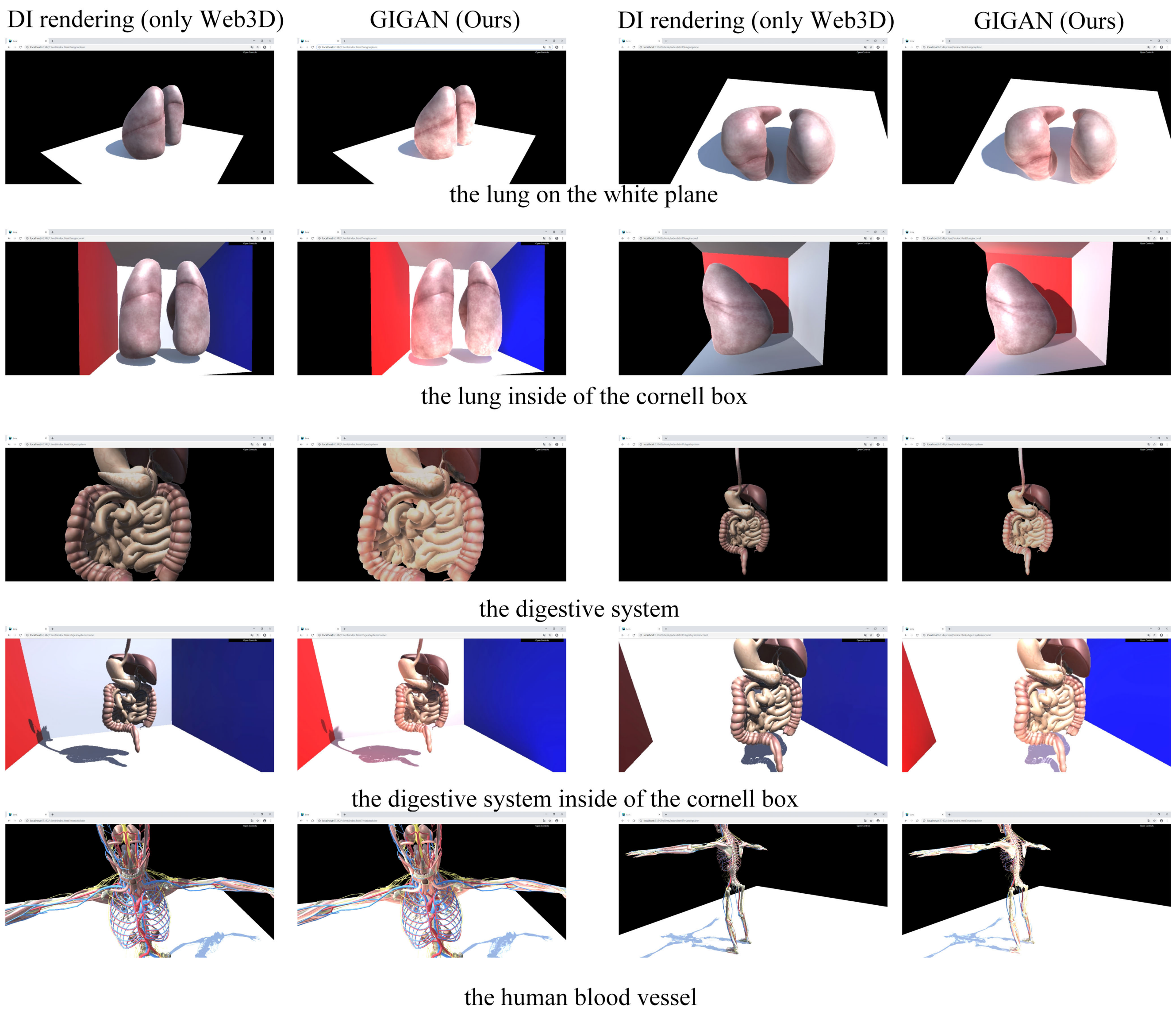

DI rendering (only Web3D)
GIGAN (Ours)
DI rendering (only Web3D)
GIGAN (Ours)
the lung on the white plane
the lung inside of the cornell box
the digestive system
the digestive system inside of the cornell box
the human blood vessel

FIGURE 2. **Comparison of the final results in various medicine models. Web3D rendering only has direct illumination. Our GIGAN blends direct illumination with GI generated by GAN locally.**

rendering it on the cloud server. Deep Illumination is our direct inspiration for this work because it has excellent performance in GI map generation with GAN. In this section, we mainly explain the proposed network.

A. NETWORK STRUCTURE

Figure 3 shows the network architecture of GIGAN. For the generator network, we use the encoder-decoder framework with skip connection based on U-net [34]. Skip connection connects the *ith* layer and the $(N - i)th$ layer of the network (N represents the overall number of layers of the network), which enables the information to be transmitted directly from the encoder layer to the decoder layer. Skip connection is conducive to the sharing of low-level information between input and output, and can also alleviate the gradient disappearance problem.

To generate the GI map, we apply auxiliary buffers, including the albedo map, normal map in view space, and depth map. The albedo map provides the color information of texture to the network. The normal map helps the network to detect silhouettes of objects and discontinuities in shading better. The depth map helps the network to identify the distance from the camera. In general, the auxiliary buffers help to disambiguate the colors by providing information about the contours and silhouettes of the scene objects, as well as about different materials. DI map and auxiliary buffers are concentrated as input to the generator and then go through eight down-sampling layers and eight up-sampling layers to generate a GI map. *LeakyReLu* is used as the activation function throughout the down-sampling process. *ReLu* and *tanh* (the last layer only) is used as the activation function during the up-sampling process. Besides, batch normalization

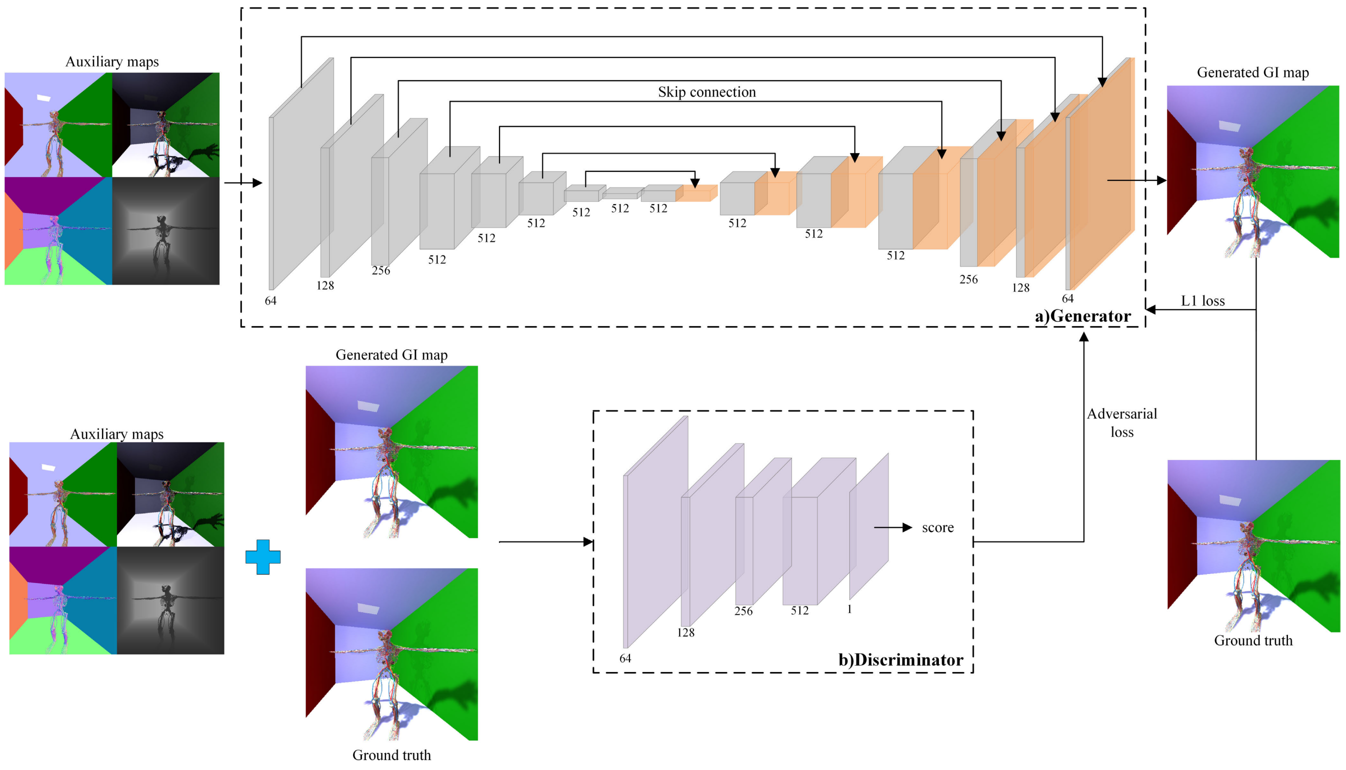


FIGURE 3. The structure of GIGAN for global illumination rendering. (a) The generative network and (b) the discriminative network takes the input and output of generator or takes the input and ground truth as its input.

layer is also used except for the first layer and the last layer. Moreover, we use dropout layer in the first three up-sampling process instead of providing random noise to the generative network [10].

For the discriminator network, since the global illumination generated by multiple bounces of the light source can produce a color enhancement effect in the local areas of scenes, we employ the Markovian patch GAN structure [23] to capture the high-frequency information contained in the local image. Moreover, Markovian patch GAN discriminates images in the patch size instead of the entire image, which has fewer parameters and high efficiency in training. The discriminator consists of 5 encoders, using *LeakyReLU* (excluding the last layer) as the activation function. In order to avoid introducing the interdependence of different samples in the same batch, the discriminator does not use the batch normalization layer to ensure the effectiveness of the gradient penalty.

B. TRAINING STRATEGY

We combine conditional Wasserstein GAN with gradient penalty (CWGAN-GP) [35] and L1 regularization loss to make the generated image as similar as possible to the target image and to improve the stability of the training process and the diversity of output results. The conditional Wasserstein GAN with the gradient penalty loss is defined as:

$$\begin{aligned}\mathcal{L}_{CWGAN-GP}(G, D) = {} & \mathbb{E}_{c,y\sim p_{tr}(c,y)}[D(c, y)] \\ & - \mathbb{E}_{c\sim p_{tr}(c)}[D(c, G(c))] \\ & + \lambda_{GP}\mathbb{E}_{c\sim p_{tr}(c),\bar{y}\sim p_{gp}} \\ & \times [(\left\|\nabla_{\bar{y}}D(c, \bar{y})\right\|_2 - 1)^2] \qquad (2)\end{aligned}$$

where G is a generator, D is a discriminator, $c, y \sim p_{tr}(c, y)$ are training images respectively from the source domain and the corresponding target domain, $\bar{y} \sim p_{gp}$ represents the distribution after linear interpolation between the real data distribution and the generated data distribution, λ_{GP} is a hyperparameter.

L1 loss is equivalent to per-pixel loss, which is defined as

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{c,y\sim p_{tr}(c,y)}[\|y - G(c)\|_1] \qquad (3)$$

Our final objective is given by:

$$G^*, D^* = arg \min_G \max_D \mathcal{L}_{CWGAN-GP}(G, D) + \lambda_{L1}\mathcal{L}_{L1}(G) \qquad (4)$$

Alternativaly, the training time of the network consumes 3-4 hours on a single GPU. During training, the network randomly reads data from the data set in batch size to 8. The G and D alternately use mini-batch stochastic gradient descent and the Adam solver (learning rate ϵ is 0.0002; β_1 is 0.5; β_2 is 0.999; λ_{GP} is 10; λ_{L1} is 100) to update the weight of the network.

C. DATA ACQUISITION

We will now describe the preparation of the training data. The training set contains 5 scenes, which are the lung on the white plane, the lung inside of the cornell box, the digest system on the white plane, the lung inside of the cornell box, and the human blood vessel on the white plane. We start with a smooth camera fly-through animation with 1500 frames for each scene available for training and 100 frames for validation. Every frame consists of depth, normal, albedo, direct illumination maps and the ground truth from the path tracing. Meanwhile, a direct light source moves along the x-axis and y-axis of the scene to provide changes in illumination. All image pairs are in $512 * 512$ resolution.

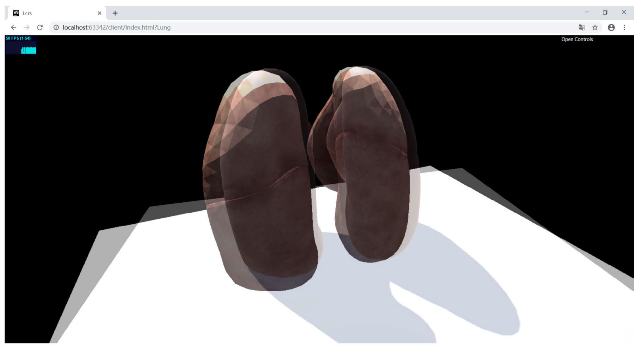
While fly-throughs of 3D scenes provide a convenient way to collect arbitrary amounts of training data, changes can be a problem. The network learns to reconstruct global illumination it sees during training. If the training set does not have any new objects or change the color of lights, its ability to reconstruct such unseen features can remain limited. Ideally, we could thus train the network with dozens of very different scene geometries, lighting setups, and camera motions.

Algorithm 1 Structure-Aware 3D Image Warping Method

```
Input: $\boldsymbol{M}_p, \boldsymbol{M}_v, \boldsymbol{M}_w$, GI map $I_{GI}$, local camera $C_{local}$, depth
       map in local camera $D_{local}$,depth map in GI camera $D_{GI}$
       //The depth map in the GI camera is derived from the
       auxiliary information generated by the local camera during
       the GAN prediction phase(see the red dotted line in Figure 1).
Output: The warping image $I_{warping}$
1: Given the vertex coordinates in model space $V_p$ rendered by $C_{local}$
2: Given the $\boldsymbol{M}_p$, $\boldsymbol{M}_v$ and $\boldsymbol{M}_w$ of the GI camera
3: Calculate UV texture coordinates $UV$ based on Equation 5
4: Calculate $I_{warping}.rgb$ based on sampling $I_{GI}$ according to $UV$
5: if $UV.x <= 0.0$ or $UV.y <= 0.0$ or $UV.x >= 1.0$ or $UV.x >= 1.0$ or abs($D_{local}$-$D_{GI}$)>0.05 then
       //if can't find UV in GI map or inconsistent depth value.
6:     Calculate $I_{warping}.rgb$ based on using traditional linear texture filtering method
7: end if
8: return $I_{warping}$
```

V. STRUCTURE-AWARE 3D IMAGE WARPING

The two cameras projecting the DI and GI information of the scene must have the same direction and position in world space. However, it takes 20-30 ms to predict one GI map. Even if DI map can be rendered immediately with new viewpoints, the GI map corresponding to the new viewpoint will be generated later. This latency causes the DI map and GI map to become asynchronous, which causes the ghost distortion(see Figure 4). Cloud baking [8] is optimized with pixel-level 3D image warping because it aims at massive scenes (the size of the scene model data is vast). The limited loading capability of the Web3D client requires that massive amounts of data (mainly model data in the scene) can only be loaded and


a) Blend without structure-aware 3D image warping

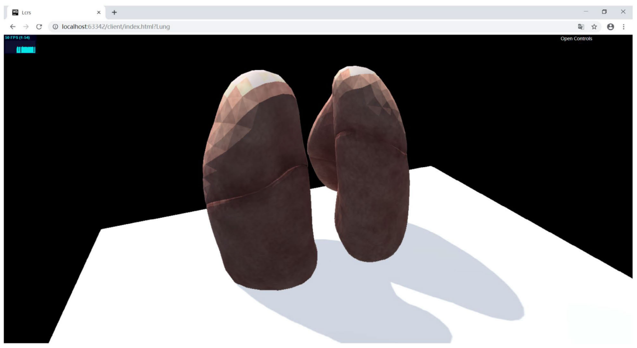

b) Blend with structure-aware 3D image warping

FIGURE 4. Blending maps with structure-aware 3D image warping vs. without it.

displayed after being lightweight. However, the lightweight model data is different from the original data. These differences makes cloud baking system optimize distortion based on the pixel-level 3D image warping method.This paper aims to the medical models, so the scene data is not very huge, and there is unnecessary to use lightweight models at the Web3D client. Therefore, this paper proposes a more efficient structure-aware 3D image warping method, which reconstructs the high quality illumination information mainly by rebinding the 3D vertices and texture coordinates, as illustrated in Figure 5(b).

The pixel-level 3D image warping method divides into two steps (see Figure 5(a)): 1) Reference image projection. It projects pixels on the reference image plane into 3D world space to form the 3D point cloud with color; 2) 3D point cloud re-projection. They are then projected onto the target image plane to form a new image. This technique is commonly used in image-based rendering applications to address image misalignment caused by camera parameter variations. Cloud baking makes full use of this technology, which uses the GI map as the reference image, and the DI map rendered by the Web3D client as the target image. The GI map completely fits with the DI map after 3D warping method.

For web-based virtual reality applications, the properties of the 3D model like vertices and materials are accessible. Therefore, this paper replaces the pixel-level 3D image warping with the structure-aware 3D image warping. This method

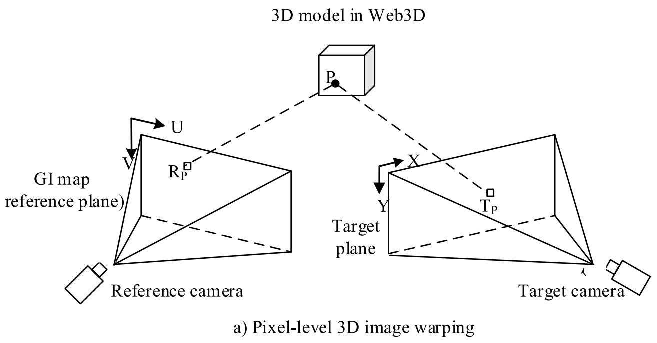


FIGURE 5. Pixel-level 3D image warping vs. Structure-aware 3D image warping.

makes fully use of the particularity of the web-based virtual reality application and eliminates the most complicated first step in the pixel-level 3D image warping. Instead of projecting pixels on the reference image into the 3D space, the image as a texture is directly mapped to the 3D model of the Web3D client. This algorithm reduces the amount of computation from pixel-by-pixel calculation to key pixel (the 3D model vertices) calculation (see Figure 5(b)). Moreover, the entire mapping process uses the linear texture filtering, which is more efficient than conventional pixel-by-pixel filtering. In this paper, we set the GI map generated by GAN as a reference texture and then calculate the corresponding UV coordinates based on the vertex coordinates of the 3D model and the matrix of the GI camera C_{GI}. Finally, the texture is bound to the vertices of the 3D model by the UV coordinates and then projected to the local camera. Of course, there is occlusion between different objects in the scene, so we need to consider the depth information between local camera and GI camera during warping (see Algorithm 1).

$$UV = \frac{(\boldsymbol{M}_p * \boldsymbol{M}_v * \boldsymbol{M}_w * P).xy}{(\boldsymbol{M}_p * \boldsymbol{M}_v * \boldsymbol{M}_w * P).w * 2.0} + (0.5, 0.5) \qquad (5)$$

Equation(5) shows the calculation of the UV coordinates, where P represents the vertex coordinate of the 3D model in the model space, which contains four elements x, y, z, and w; x, y, and z respectively represent the components on the x, y, and z axes; w is the homogeneous value; UV represents the texture coordinate of the reference image, which is a 2D vector; $\boldsymbol{M}_p, \boldsymbol{M}_v, \boldsymbol{M}_w$ respectively represent the projection matrix, the view matrix and the world matrix of the GI camera.

VI. RESULTS AND DISCUSSION

For the web client, we use the laptop with an Intel Core i7-7700HQ 2.8 GHz CPU, an Nvidia GeForce GTX1060M GPU, and 8GB of RAM. The computer runs Windows 10 and uses Google Chrome version 71 as the web client. We build on Tensorflow.js framework to predict GI maps using GAN on the Web3D client.

A. CLOUDBAKING VS. OUR GIGAN SYSTEM

In this section, the experiment compares the difference between cloud baking and our GIGAN system.

TABLE 1. Cloud baking vs. GIGAN system (Ours) in interaction latency.

Interactive latency(ms)	Cloud baking	GIGAN (Ours)
Transmission Time	100	/
Rendering/Inference Time	45	25
Image encoding time	30	/
Image decoding time	20	/
Warping time	15	5
Blending time	3	3
Total time	215	35

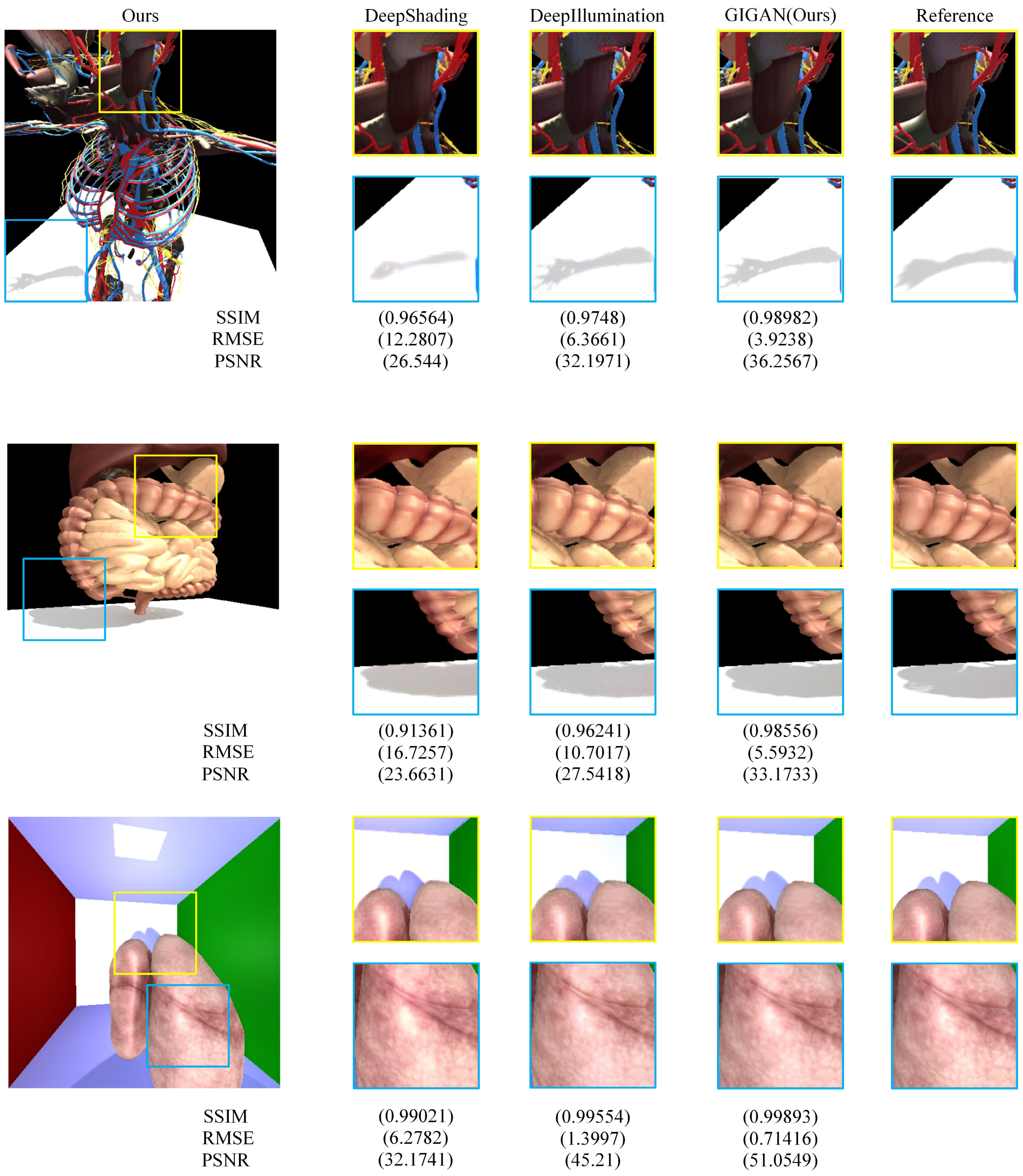
TABLE 2. Cloud baking vs. GIGAN system (Ours) in storage space occupation.

Scene	Cloud baking	GIGAN (Ours)
Lung	1.15 GB	0.2 GB
Digestive system	2.38 GB	0.2 GB
Human blood vessel	3.36 GB	0.2 GB

Firstly, we measure the average interaction latency for every request. The interaction latency in GIGAN system consists of the GI prediction, structure-aware 3D image warping, and blending phase, and has less latency compared to the cloud baking system. Table1 shows the results that since GIGAN system only has a Web3D client, the transmission phase, the image encoding, and the image decoding phase are eliminated. Besides, we can see that generating GI maps via GAN is faster than real-time rendering of GI maps. Moreover, structure-aware 3D image warping has higher computational efficiency than pixel-level 3D image warping. Secondly, we evaluate the storage space occupation. Since the cloud baking system uses the GI map tree to reduce interaction latency, the saved GI maps take up a lot of storage space. In contrast, the storage space of our GIGAN system is mainly occupied by the GAN model, and the storage space of our GIGAN system is still the same, even for different scenarios.

B. COMPARISON WITH DEEP SHADING AND DEEP ILLUMINATION

In this section, the experiment studies the quality of generated GI map by GAN. To compare various shading algorithms of deep learning, we choose three different evaluation metrics,



	DeepShading	DeepIllumination	GIGAN(Ours)
SSIM	(0.96564)	(0.9748)	(0.98982)
RMSE	(12.2807)	(6.3661)	(3.9238)
PSNR	(26.544)	(32.1971)	(36.2567)
SSIM	(0.91361)	(0.96241)	(0.98556)
RMSE	(16.7257)	(10.7017)	(5.5932)
PSNR	(23.6631)	(27.5418)	(33.1733)
SSIM	(0.99021)	(0.99554)	(0.99893)
RMSE	(6.2782)	(1.3997)	(0.71416)
PSNR	(32.1741)	(45.21)	(51.0549)

FIGURE 6. We use RMSE (lower is better), SSIM (higher is better), and PSNR (higher is better) to compare our results with baseline methods: Deep Shading [27] and Deep illumination [28] on a test set rendered in 512*512 resolution.

including RMSE, SSIM, and PSNR. We only present subsets of metrics in this paper for concise demonstration purposes, and some parts of the image are zoomed to facilitate detailed comparison. We have selected two state-of-the-art shading algorithms of deep learning to compare the outcomes with, including Deep Shading [27] and Deep illumination [28].

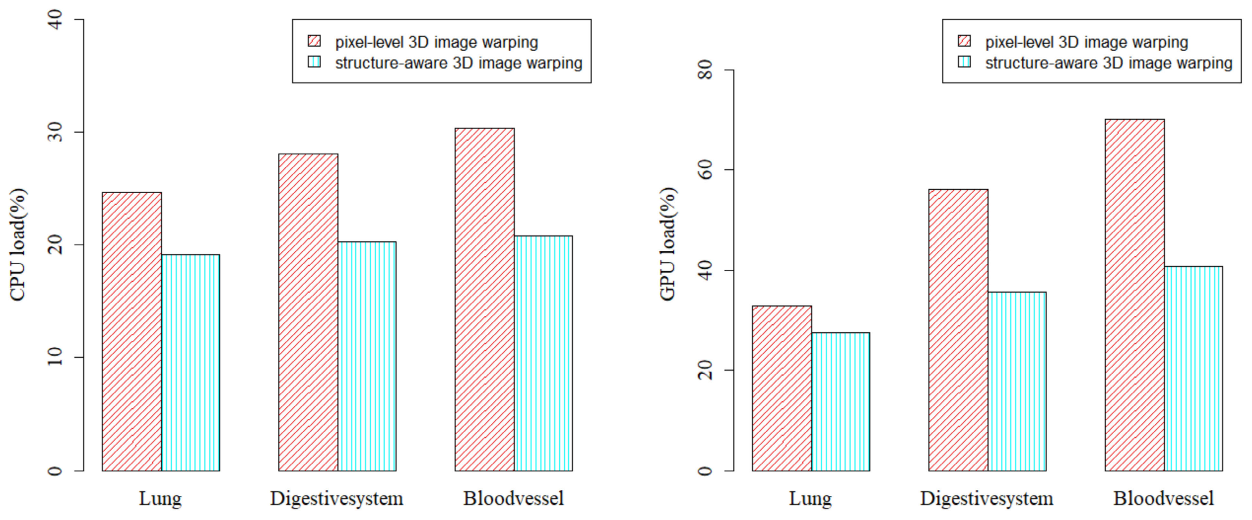


FIGURE 7. CPU(left) and GPU(right) load when using pixel-level 3D image warping and structure-aware 3D image warping method respectively in lung, digestive system and blood vessel scenes.

TABLE 3. Structure-aware 3D image warping vs. system performance.

Inference frequency	Frames per second	SSIM(with 3D warping)	SSIM(without 3D warping)
Every frame	5	0.98	0.97
Every 10 frames	17	0.97	0.88
Every 20 frames	27	0.96	0.83
Every 30 frames	35	0.94	0.76
Every 60 frames	50	0.91	0.6

To achieve a fair comparison, the input to all networks is albedo map, DI map, normal map in view space, and depth map with 512*512 resolution(see Figure 7). We have found that our method can perform better visual effect than others.

C. PIXEL-LEVEL 3D IMAGE WARPING VS. STRUCTURE-AWARE 3D IMAGE WARPING

In this section, the experiment compares the performance between structure-aware 3D image warping and pixel-level 3D image warping method. Figure 7 shows that the structure-aware 3D image warping method has a lower CPU and GPU load(in %) than the pixel-level 3D image warping method. This further illustrates that structure-aware 3D image warping is superior to pixel-level 3D image warping in both performance and rendering requirements. Moreover, structure-aware 3D image warping can also effectively improve the whole system performance. For our GIGAN system, the system performance is mainly reflected in the frame rate and the image quality after blending DL map with GI map. Therefore, we move the camera to evaluate the influence of inference frequency on the system frame rate and image quality. We set different frequency to test the system performance, for example, the third row in Table 3 means the generative network generate one GI map every 20 frames. Table 3 shows the frames per second(fps) and the image quality in the SSIM metric after blending DI map with GI map. It illustrates that the lower the inference frequency, the better the system performance. Furthermore,the blending image quality with structure-aware 3D image warping performs better results than without it.

D. INTERNAL CAMERA PROPERTY VS. QUALITY

The final display of the image depends on the inherent properties of the local camera, such as field of view, near plane, far plane, aspect ratio, and so on. The aspect of the camera is normally set according to the screen resolution of a device, therefore, in this experiment, we set the camera's aspect to 1 to ensure that the generative network make the correct inference in various devices. As another important attribute of the camera, FOV could determine the range of view. When the camera renders an image from the same viewpoint, a higher value of FOV will allow it to view more information about the scene. Undoubtedly, the increase in FOV could distort the object photographed by the camera, which may affect the final results. Let F_{GI} be the FOV of the GI camera to render training dataset, that is to say, the GI map generated by network is similar to rendered GI map in F_{GI}. F_{local} be the FOV of the local camera to render DI map and set it to 60°. We set different F_{GI} to render the training set to train the GAN and use SSIM to evaluate the impact of F_{GI} on the quality of

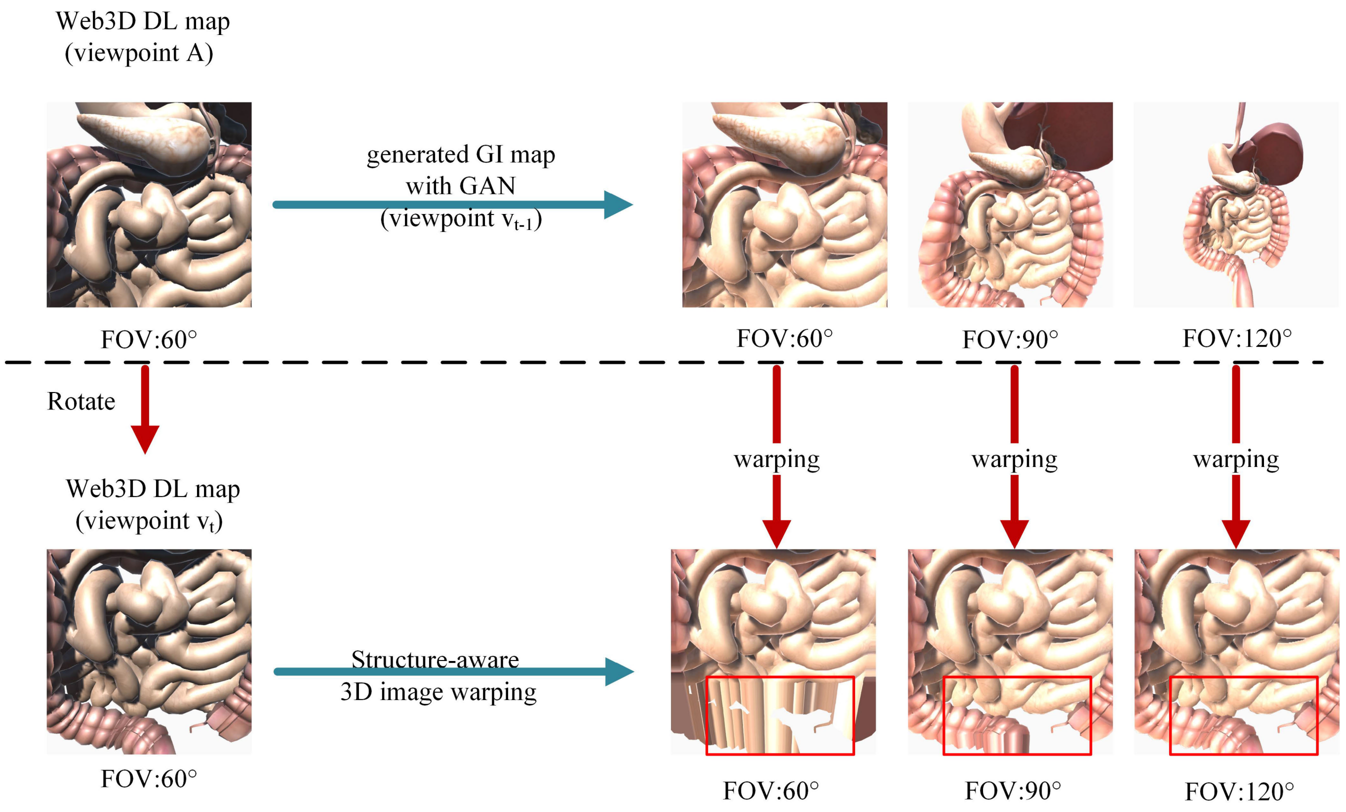


FIGURE 8. **The Web3D client render a DI map at viewpoint v_{t-1},and then using GAN to generate GI map under the same viewpoint v_{t-1} but in different FOV. In the next request, the Web3D client render a DI map under viewpoint v_t, we only use structure-aware 3D image warping instead of using GAN again to generate GI map under viewpoint v_t. We can find that the larger FOV, the better the warping result.**

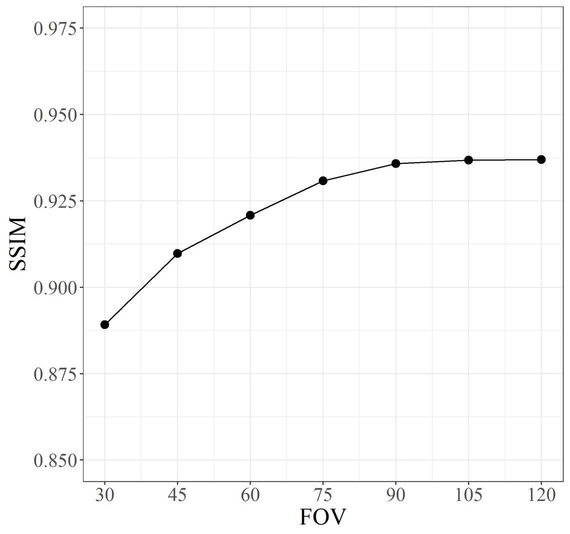


FIGURE 9. **The impact of C_{GI} on the generated image quality. The horizontal axis is the FOV in degrees and the vertical axis is the SSIM value (higher is better).**

the generated GI map. As shown in Figure 9, the generative networks are trained a certain number of iterations. We use SSIM to compare the ground truth and the generated GI map. It illustrates that as the FOV increases, the generated image quality also increases, but its growth trend slows down. When the FOV of the camera is 90 degrees or larger, there is no significant improvement in SSIM, and the visual effects are roughly similar.

Besides, we find that the higher F_{GI} can further improve system performance with structure-aware 3D image warping. Table 3 shows the structure-aware 3D image warping can improve the final result in low inference frequency. And it can be realized that as the F_{GI} increases, the GI map has more information than the DI map rendered by Web3D. In Figure 8, we train three GANs with different F_{GI} (60°, 90° and 120°). When the Web3D client renders a DI map by the local camera at viewpoint v_{t-1}, we use three GANs to generate GI map at the same viewpoint v_{t-1}. In the next request, the local camera rotates to viewpoint v_t, so the Web3D client renders a DI map at viewpoint v_t immediately. We only use the structure-aware 3D image warping method instead of using GAN again to predict the GI map at viewpoint v_t. It uses the GI map in different FOV of the previous frame as the reference plane and uses the DL map at the viewpoint v_t as the target plane. We can find that the larger the FOV, the better the warping result. Therefore, the inference frequency can be further reduced, but the final image result may be blurred due to the large FOV value. Therefore, we need to weigh the values of FOV and inference frequency. In this paper, we set F_{GI} is 90°, and every 30 frames to predict one GI map.

VII. CONCLUSION

The system proposed in this paper is capable of producing vivid global illumination on web client and can display high-quality rendering of medicine models. The Web3D client uses the pre-trained generative model to predict the GI map. Compared with the previous method, the system does not need the server to render the GI map, which can avoid the latency caused by the client-server mode. Moreover, structure-aware 3D image warping can further improve the overall performance of the system.

At present, due to the viewpoint correlation of our GIGAN system, although the 3D warping method can effectively reduce the prediction frequency, if the user moves the camera greatly, the latest GI map is not enough to supplement the GI information at the latest viewpoint, the Web3D client needs to use the generative network to re-predict the GI map at the latest viewpoint, which will lead to frequent prediction. In addition, although generative adversarial network can effectively approximate global illumination and still achieve the expected result for some unknown areas of the scene, the GI map prediction for some complex dynamic scene is still a huge challenge such as blood flow in blood vessels and cells in the future work.

REFERENCES

[1] S. Birr, J. Mönch, D. Sommerfeld, and B. Preim, "A novel real-time Web3D surgical teaching tool based on WebGL," in *Proc. des Workshops Bildverarbeitung für die Medizin, Algorithmen-Systeme-Anwendungen*, T. Tolxdorff, T. M. Deserno, H. Handels, and H.-P. Meinzer, Eds. Berlin, Germany: Springer, Mar. 2012, pp. 404–409, doi: 0.1007/978-3-642-28502-8_70.

[2] J. L. Crossingham, J. Jenkinson, N. Woolridge, S. Gallinger, G. A. Tait, and C.-A.-E. Moulton, "Interpreting three-dimensional structures from two-dimensional images: A web-based interactive 3D teaching model of surgical liver anatomy," *HPB*, vol. 11, no. 6, pp. 523–528, Sep. 2009.

[3] H. Brenton, J. Hernandez, F. Bello, P. Strutton, S. Purkayastha, T. Firth, and A. Darzi, "Using multimedia and Web3D to enhance anatomy teaching," *Comput. Edu.*, vol. 49, no. 1, pp. 32–53, Aug. 2007.

[4] N. W. John, "The impact of Web3D technologies on medical education and training," *Comput. Edu.*, vol. 49, no. 1, pp. 19–31, Aug. 2007.

[5] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proc. 11th Annu. Workshop Netw. Syst. Support for Games (NetGames)*, Nov. 2012, p. 2.

[6] L. Hsu, "Web 3D simulation-based application in tourism education: A case study with Second Life," *J. Hospitality, Leisure, Sport Tourism Edu.*, vol. 11, no. 2, pp. 113–124, Jul. 2012.

[7] C. Liu, J. Jia, Q. Zhang, and L. Zhao, "Lightweight WebSIM rendering framework based on cloud-baking," in *Proc. ACM SIGSIM Conf. Princ. Adv. Discrete Simul. (SIGSIM-PADS)*, 2017, pp. 221–229.

[8] C. Liu, W. T. Ooi, J. Jia, and L. Zhao, "Cloud baking: Collaborative scene illumination for dynamic Web3D scenes," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMM)*, vol. 14, no. 3s, p. 59, 2018.

[9] M. Zhu, G. Morin, V. Charvillat, and W. T. Ooi, "Sprite tree: An efficient image-based representation for networked virtual environments," *Vis. Comput.*, vol. 33, no. 11, pp. 1385–1402, Nov. 2017.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.

[11] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "Gaminganywhere: The first open source cloud gaming system," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 10, no. 1s, p. 10, 2014.

[12] 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, p. 695, Aug. 2004.

[13] 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.

[14] H. R. Maamar, A. Boukerche, and E. Petriu, "Streaming 3D meshes over thin mobile devices," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 136–142, Jun. 2013.

[15] C. Crassin, D. Luebke, M. Mara, M. McGuire, B. Oster, P. Shirley, and P.-P. S. C. Wyman, "Cloudlight: A system for amortizing indirect lighting in real-time rendering," *J. Comput. Graph. Techn.*, vol. 4, no. 4, pp. 1–27, 2015.

[16] K. Bugeja, K. Debattista, and S. Spina, "An asynchronous method for cloud-based rendering," *Vis. Comput.*, vol. 35, no. 12, pp. 1827–1840, Dec. 2019.

[17] W. Chen and J. Hays, "SketchyGAN: Towards diverse and realistic sketch to image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018.

[18] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017.

[19] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017.

[20] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, "Invertible Conditional GANs for image editing," 2016, *arXiv:1611.06355*. [Online]. Available: https://arxiv.org/abs/1611.06355

[21] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," 2016, *arXiv:1609.07093*. [Online]. Available: https://arxiv.org/abs/1609.07093

[22] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 613–621.

[23] C. Li and M. Wand, "Precomputed real-time texture synthesis with Markovian generative adversarial networks," in *Proc. ECCV*. Springer, 2016, pp. 702–716.

[24] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017.

[25] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017.

[26] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1857–1865.

[27] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel, "Deep Shading: Convolutional Neural Networks for Screen Space Shading," *Comput. Graph. Forum*, vol. 36, no. 4, pp. 65–78, Jul. 2017.

[28] M. M. Thomas and A. G. Forbes, "Deep illumination: Approximating dynamic global illumination with generative adversarial network," 2017, *arXiv:1710.09834*. [Online]. Available:https://arxiv.org/abs/1710.09834

[29] A. Smolic, K. Müller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems," in *Proc. 15th IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 2448–2451.

[30] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *Proc. Stereoscopic Displays Virtual Reality Syst.*, May 2004, pp. 93–104.

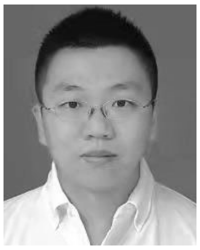
[31] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, "Layered depth images," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. New York, NY, USA: Association for Computing Machinery, 1998, pp. 231–242, doi: 10.1145/280814.280882.

[32] P. Bao and D. Gourlay, "Remote walkthrough over mobile networks using 3-D image warping and streaming," *IEE Proc., Vis. Image Process.*, vol. 151, no. 4, p. 329, 2004.

[33] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive indirect illumination using voxel cone tracing," *Comput. Graph. Forum*, vol. 30, no. 7, pp. 1921–1930, Sep. 2011.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Springer, 2015, pp. 234–241.

[35] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst. 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. New York, NY, USA: Curran Associates, 2017, pp. 5767–5777. [Online]. Available: http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf



NING XIE received the M.E. and Ph.D. degrees from the Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, in 2009 and 2012, respectively. In 2012, he was appointed as a Research Associate with the Tokyo Institute of Technology. Since 2017, he has been an Associate Professor with the School of Computer Science and Engineering, UESTC. His research interests include computer graphics, game engine, and the theory and application of artificial intelligence and machine learning. His research is supported by the research grants, including NSFC (China), MOE (China), CREST (Japan), and The Ministry of Education, Culture, Sports, Science and Technology (Japan).

YIFAN LU received the B.Ec. degree in instructional technology from Nanchang Hangkong University, Nanchang, China, in 2019. He is currently pursuing the M.Eng. degree with the University of Electronic Science and Technology. His current research interests include computer graphics, game engine, and deep learning.

CHANG LIU received the Ph.D. degree from the School of Software Engineering, in 2019. He has been with the School of Information Engineering, Nanchang Hangkong University, Nanchang, China, where he is currently a Lecturer. He is a member of ACM and Chinese Computer Federation (CCF). His research interests include medical images analysis, WebVR visualization, virtual reality, and deep learning.