Photorealistic Rapid Computer-Generated Holography Employing an Enhanced Path Tracing Technique With Sequence Generated Trial

Xindi Chen, Aiming Ge, Dongsheng Zhu, Qiuyang Wang, Jiangbo Wu, and Shuo Liu

***Abstract*—Fast holographic generation technology depends on the swift creation of computer-generated holograms (CGHs). Currently, the fast generation method of computer-generated holograms based on ray tracing is still based on non-physical ray tracing methods, making it difficult to accurately simulate and generate computer-generated holograms. There is a scarcity of research on employing path tracing to swiftly generate CGH with a minimal computational workload and reduced computation time. To address this issue, we employed physically-based path tracing technique for realistic rendering and made improvements on the bidirectional scattering distribution function (BSDF) and ray sampling methods. We have developed a path tracing technique based on improved BSDF and single ray sampling to generate CGHs with realism. In comparison to conventional point cloud and ray tracing methods, the path tracing approach employs distinct shading rendering and lighting models to enhance a scene's sense of realism and generate CGHs rapidly. Our proposed technique takes approximately 10 ms to generate CGHs in various scenes, and the reconstructed images featuring photorealistic appearances are produced through optical reconstruction experiments. This novel method offers an efficient solution for generating CGHs and we attempted to generate a sequence using our method, applying our method to rapidly generate CGH videos.**

***Index Terms*—Computer-generated hologram, light model, parallel computing, path tracing.**

I. Introduction

SINCE its inception in 1966, CGH has been employed to rapidly capture the light fields of three-dimensional(3D) objects [1]. The phase information of light fields represents the depth information of the 3D objects [2]. The primary advantage of CGH lies in its ability to generate holograms quickly and reproduce the 3D light field data, thus enabling 3D display.

There are currently various methods that can be proposed for generating CGHs. Many CGH generation methods have been derived from the traditional point cloud method [3]. Ichikawa et al. proposed a method for computing CGH using parallelized ray tracing techniques [4]. Building on this, Wang et al. proposed an efficient and realistic method for generating CGHs based on ray tracing techniques [5]. Blinder et al. first applied the point cloud method with path-tracing global illumination techniques to generate CGH [6]. Zhong et al. proposed a method for generating CGH based on volumetric representation with an improved ray tracing algorithm [7].

Several methods have been proposed to facilitate the rapid generation of Computer-Generated Holograms, particularly in accelerating the point cloud method. There are also acceleration methods that rely on the point cloud method [8], such as the look-up tables [9], [10], adaptive point spread spherical wave acceleration technique [11], the fast calculation method based on patch modeling [12], the wave-front recording plane method [13], and high-performance hardware calculation approaches [14]. The point cloud method is limited by the number of physical points that make up an object. A large number of physical points can generate a higher quality CGH, but each physical point is diffracted to the holographic recording plane as a light source. The more physical points there are, the greater the computational load. This calculation process can be accelerated using look-up tables, but it requires additional storage space and can also take a long time to compute.

Additionally, various accelerated techniques that build upon traditional ray tracing methods have been suggested, such as parallel computing acceleration [15], [16], [17], wavefront records plane acceleration [18], [19], [20], and foveated rendering acceleration [21]. However, most of these ray-tracing approaches use the empirical Phong model [22]. While this model requires less computation, it makes rendering objects with global illumination difficult, thus reducing the overall realism of the scene.

As computing devices improve, the more advanced path tracing technique has emerged from the foundations of ray tracing. Path tracing is an evolution of the ray tracing method [23], building on ray casting while differing in shading rendering to create more realistic outcomes. Rendering equations through path tracing increases the computational load [24]. Several techniques have been introduced to reduce the computational complexity of path tracing methods [25], [26], [27], [28], [29], as path tracing has been demonstrated to generate large-scale scenes successfully [30].

The principles of path tracing and traditional ray tracing share similarities, as both involve projecting rays from the receiving surface into the scene (ray casting) [31]. However, the distinction

Manuscript received 6 September 2023; accepted 7 September 2023. Date of publication 12 September 2023; date of current version 22 September 2023. The work of Ge Aiming was supported by the Yiwu Research Institute of Fudan University under Award 20-1-20. *(Corresponding author: Aiming Ge.)*

Xindi Chen and Dongsheng Zhu are with the Academy for Engineering & Technology, Fudan University, Shanghai 200433, China (e-mail: xdchen20@fudan.edu.cn; dszhu20@fudan.edu.cn).

Aiming Ge, Qiuyang Wang, Jiangbo Wu, and Shuo Liu are with the School of Information Science and Technology, Fudan University, Shanghai 200433, China (e-mail: amge@fudan.edu.cn; wangqiuyang21@m.fudan.edu.cn; jbwu21@m.fudan.edu.cn; liushuo21@m.fudan.edu.cn).

Digital Object Identifier 10.1109/JPHOT.2023.3314525

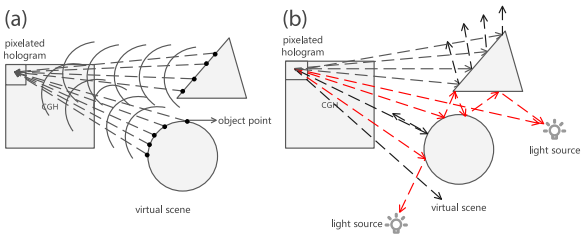

Fig. 1. Principle of (a) traditional point cloud method and (b) path tracing method.

arises in that path tracing depends on accurately calculating the energy of all rays using radiometric metrics and rendering equations [24]; only rays striking the light source contribute to the final energy of the rays being calculated. In comparison, traditional ray tracing utilizes a local computing model, relying on empirical models [22], which makes it challenging to precisely calculate the paths and energies of all rays.

Currently, it is common to use the Phong model in ray tracing to generate computational holograms and perform global illumination calculations [5], [7], [18]. However, the Phong model is not capable of accurately calculating global illumination. On the other hand, the method of global illumination calculation based on path tracing often requires a long computation time [6], making it challenging to quickly generate computational holograms with accurate lighting.

To address the aforementioned challenges, this article proposes an improved method for fast computational hologram generation based on path tracing. We have made modifications to the complex BSDF calculation method and adopted a single-ray sampling approach to generate the computational hologram. By leveraging parallel computing on general-purpose graphics processing units (GPUs), photorealistic CGHs are generated quickly, and the optical reconstruction of virtual 3D objects using spatial light modulators (SLMs) proves highly effective for CGH generation. Compared to the previous path tracing method [6] and ray tracing method [5], [7], [18], our method demonstrates certain advantages in terms of computation time.

II. Proposed Method

A. Path Tracing

CGHs are commonly calculated using the point cloud method. This article integrates CGH generation with path tracing technology, utilizing a unique light model to render virtual scenes more realistically. CGHs based on path tracing can accommodate objects in virtual spaces under complex lighting conditions. Path tracing involves emitting rays towards objects in virtual spaces to ascertain the light emitted or reflected from these objects. This characteristic is useful for determining the phase of rays. A comparison of the point cloud method and path tracing is provided in Fig. 1. In particular, Fig. 1(a) shows that the point cloud method necessitates the calculation of the total number of pixels multiplied by the total number of object points. As depicted in Fig. 1(b), the path tracing hologram calculation employs backward light entering the virtual scene. Path tracing requires computing the total number of pixels multiplied by the number of rays emitted by a single pixel. Both approaches aim to discern the ray path from the pixel to the light source in a 3D virtual space and accumulate the radiance of ray hit points along the ray path. In the end, we can acquire the radiance of each pixel, which can be considered as the amplitude of the light field. In Fig. 1(b), the red rays signify those striking the light source and contributing to the ultimate light energy computation, whereas the black rays denote those failing to reach the light source and hence not factored into the final light energy estimation.

Path tracing and traditional ray tracing both adopt the principle of rays casting. The primary difference between path tracing and traditional ray tracing lies in the calculation of light contributions to pixels. In traditional ray tracing, contributions are determined when a ray intersects an object. In contrast, path tracing calculates contributions only when a ray strikes the light source, assigning a contribution value of zero for rays that miss the light source. While path tracing adopts the ray casting principle of traditional ray tracing, it deviates in its approach to calculating light contributions to pixels. This discrepancy originates from traditional ray tracing's reliance on local illumination models, which overlook the influence of indirect lighting from surrounding objects, consequently reducing computational overhead. Conversely, path tracing utilizes a global illumination model that encompasses both direct and indirect lighting effects on objects, resulting in enhanced rendering realism. Depending on the different designs of global illumination models, the computational complexity of path tracing may vary. Therefore, we will first introduce the illumination model of our method.

The general expression of the object light wave function is

$$O(x,y) = A(x,y)\exp[i\varphi(x,y)] \tag{1}$$

where $A(x,y)$ is the optical amplitude function, and $\varphi(x,y)$ is the optical phase function. $A(x,y)$ can be calculated by:

$$\begin{aligned} A(x,y) &= \sqrt{L_o} \\ &= \sqrt{L_e + \int_{\Omega} f_r \times L_i \times (n \cdot \omega_i)\, d\omega_i} \\ &= \sqrt{L_e + \frac{1}{m} \times \sum\nolimits_{i=0}^{m} \frac{f_r \times L_i \times (n \cdot \omega_i)}{p}} \end{aligned} \tag{2}$$

In the given paragraph, Ω represents the spherical sampling region of the look-at direction of the virtual camera, with n being the surface normal direction itself. The final observed radiance is denoted by L_o, while the emitted radiance by the object is denoted as L_e. The bidirectional scattering distribution function, which is represented by f_r, Bidirectional Scattering Distribution Function (BSDF), dictates the attenuation coefficient of the light upon striking the object, as well as the direction of light scattering. The radiance from indirect lighting of other objects is represented by L_i, and the direction of the light incident on the object is given by ω_i. Lastly, m refers to the number of rays, and p denotes the probability density of ray sampling.

For the calculation of f_r, BSDF, the Schlick method is employed to compute the Fresnel equations, determining the

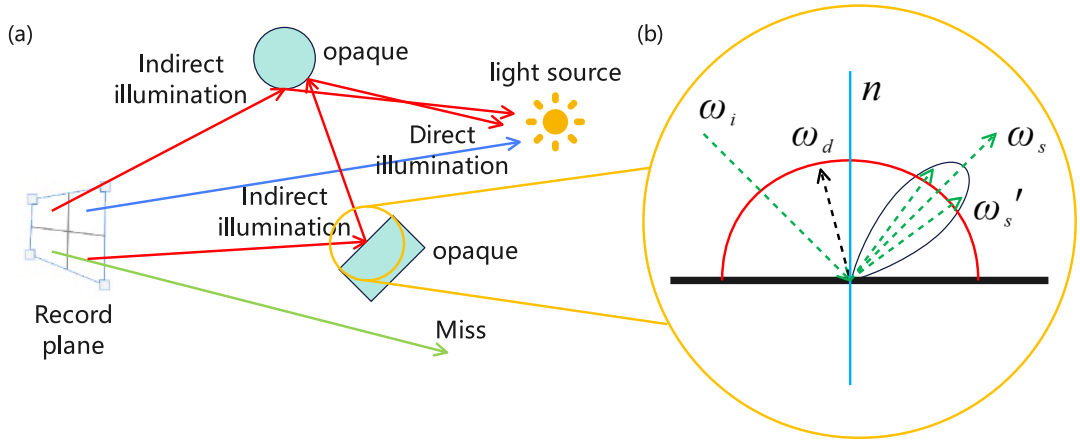

Fig. 2. (a) Global illumination of our proposed method and (b) the illumination model of opaque object.

interaction between the light ray and the object. the Schlick method can be expressed as:

$$F = F_0 + (1 - F_0)(1 - h \cdot \omega_o)^5 \tag{3}$$

The Fresnel term, F, considers F_0 as the fundamental reflectance of the plane, determined by the refractive indices of the two media in question. In this context, h denotes the half vector, while ω_o signifies the vector of the emergent ray.

We ensure that each ray of light, after hitting an object, has only one outgoing direction, instead of multiple rays being scattered when hitting a Lambertian surface. In other words, when a ray of light hits an object, there is a specific surface that causes the ray to reflect as a single ray, and the number of rays in the scene does not increase. In BSDF, we need to determine the probability density function p, the outgoing direction ω_{finalout} of the light ray after it hits the object and the attenuation coefficient f_r. In this case, our method employs a relatively simple judgment and approximation to calculate the global illumination, as well as the direction and intensity of the light rays in the model. As a result, the computational speed can be partially accelerated. The BSDF is composed of the bidirectional reflectance distribution function (BRDF) and the bidirectional transmittance distribution function (BTDF).

1) Opaque Model (BRDF): Objects in the scene are categorized into two types, with the first type being opaque objects, illustrated in Fig. 2(a). Upon a ray's interaction with an opaque object, as depicted in Fig. 2(b), a random number ran is computed.

The probability of specular, denoted by P_s, is then determined as $\text{lerp}(M, 1, F * S)$, where F is the fresnel term (calculated from (3)), M and S represent the material's metallicity and smoothness, respectively, and can be customized based on the object's properties. The linear interpolation function $\text{lerp}(A, B, weight) = (1 - weight) \times A + weight \times B$.

If $P_s > ran$, the reflection is deemed smooth, and the direction of the light ray upon impact is shown in Fig. 3. The probability density p is $p = P_s$.

The reflection direction, ω_s, glossy reflection direction, ω_s', and diffusion direction ω_d are shown in Fig. 2(b). We can

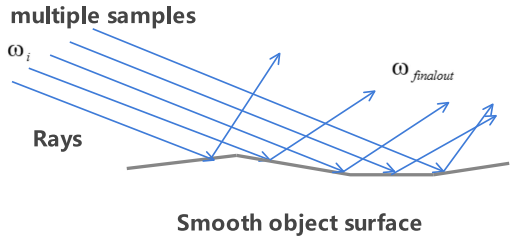

Fig. 3. Behavior of rays on a smooth surface in our model.

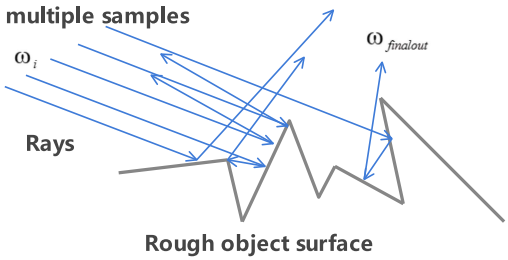

Fig. 4. Behavior of rays on a rough surface in our model.

calculate $\omega_d = \text{normalized}(n + RV)$, where RV denotes a random unit vector on the sphere. And the reflection direction is $\omega_s = \omega_i - 2(\omega_i \cdot n) \cdot n$, where ω_i is the vector of the incident light ray and n is the normal of the surface. And the ω_s' is $\text{normalized}(\text{lerp}(\omega_d, \omega_s, S))$, where S denotes the object's smoothness, which is determined based on the desired material of the object. Therefore, we can calculate ω_{finalout} using the equation $\omega_{\text{finalout}} = \text{lerp}(\omega_d, \omega_s', P_s)$.

At this stage, the attenuation coefficient is given by $f_r = \text{lerp}(samplecolor, specularcolor, P_s)$, where value $samplecolor$ is the sampled color at the hit point and the value $specularcolor$ is the pre-set color used for the attenuation coefficient.

Conversely, when $P_s \leq ran$, the reflection is classified as diffuse (Fig. 4). In this case, the probability density p is equal to $1 - \text{lerp}(M, 1, F * S)$.

At this point, the diffuse reflection direction can be calculated as $\omega_d = \text{normalized}(n + RV)$, while the specular reflection direction can be calculated as $\omega_s = \omega_i - 2(\omega_i \cdot n) \cdot n$.

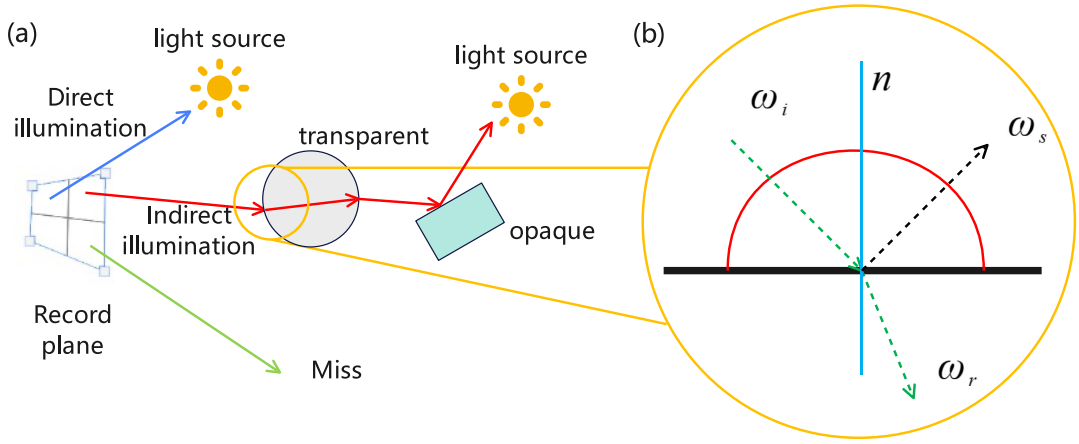

Fig. 5. (a) Global illumination of our proposed method and (b) the illumination model of transparent object.

The glossy reflection direction ω_s' is obtained by $\omega_s' = \text{normalized}(\text{lerp}(\omega_d, \omega_s, S))$. Finally, the final outgoing direction of the light ray can be calculated as $\omega_{\text{finalout}} = \text{lerp}(\omega_d, \omega_s', P_s)$.

The attenuation coefficient at this point is determined as $f_r = \text{lerp}(samplecolor, specularcolor, P_s)$.

2) Transparent Model (BTDF): The second category pertains to transparent objects, as depicted in Fig. 5(b). When a ray encounters a transparent object, its reaction is illustrated in Fig. 5(b).

We compute F using formula 3 and compare it to the random number ran.

If $F \geq ran$, we treat the ray as a mirror reflection ray with direction $\omega_{\text{finalout}} = \omega_s = \omega_i - 2(\omega_i \cdot n) \cdot n$.

The probability density p is equal to F. When the ray serves as a mirror reflection ray, the attenuation coefficient is 1, denoted by $f_r = 1$, and no attenuation occurs.

When $F < ran$, we consider the ray as a refracted ray with direction $\omega_{\text{finalout}} = \omega_r = \frac{\eta_1}{\eta_2}\omega_i - (\frac{\eta_1}{\eta_2}(\omega_i \cdot n) + \sqrt{1-\frac{\eta_1}{\eta_2}^2(1-(\omega_i \cdot n)^2)})$.

The probability density p in this case is $1 - F$.

When the ray is a refracted ray, the attenuation coefficient adheres to the Beer-Lambert law, expressed as $f_r = e^{-(1-color)\times d \times EC}$, where $color$ denotes the sampled color at the hit point, d represents the distance traversed by the ray within the transparent object, and EC signifies the set attenuation coefficient.

The goal of designing this global illumination model is to expedite the computation of f_r and precise ray directions. Employing random number judgement offers the benefit of swiftly obtaining f_r and accurate ray directions without incurring extra memory or computational costs. Given the model's highly complex material composition, the computational load required to accurately depict the interaction between material and light is substantial. Utilizing random numbers significantly streamlines this process, accelerating calculations, yet simultaneously imposes restrictions on the variety of representable materials.

We have successfully determined the accurate direction for the subsequent light ray, in addition to acquiring the f_r of each object impacted by the ray, thus facilitating the calculation of the rendering equation. The L_i in the formula represents recursion, which implies that $L_{i0} = f_{r0} \times L_{i1} \times (n_0 \cdot \omega_{i0})$, $L_{i1} = f_{r1} \times L_{i2} \times (n_1 \cdot \omega_{i1})$, and finally $L_o = L_e + \frac{1}{m} \times \sum_{i=0}^{m} \frac{f_{r0} f_{r1} \cdots f_{rk} \times L_{light} \times (n_0 \cdot \omega_{i0})(n_1 \cdot \omega_{i1}) \cdots (n_k \cdot \omega_{ik})}{p}$, assuming the recursion depth of a ray is k times. The formula incorporates the product of the attenuation coefficients for each hit of the ray on the objects, $f_{r0} f_{r1} \cdots f_{rm}$, and the product of the dot product of the surface normal and the light direction at each hit point, $(n_0 \cdot \omega_{i0})(n_1 \cdot \omega_{i1}) \cdots (n_k \cdot \omega_{ik})$. L_{light} signifies the radiance of the light source, indicating that rays not striking the light source contribute nothing.

B. CGH

To generate CGHs, we need to calculate the complex amplitude. The light wave propagates from the virtual space object to the hologram plane and is recorded. This process effectively transforms the object light wave function into the complex amplitude function of the hologram plane. This conversion occurs through discrete display surface pixels. In general, when a ray is emitted from a pixelated hologram, it is emitted from the center of the pixel. The range of the ray entering the scene is denoted as θ, and the spacing between rays is $\delta\theta$. Typically, this spacing needs to be about $\frac{1}{60}^\circ$ [4]. Within the scene, rays may either miss hitting an object (Miss), intersect with an object (Intersection), or intersect with a light source (Light Intersection), as shown in Fig. 6.

Utilizing the GPU, the complex amplitude function on the holographic plane can be calculated in parallel. For each pixelated hologram in each CGH, we emit and trace a single ray toward the 3D scene until the ray either reaches the light source or achieves the predetermined maximum bounce count. Finally, we accumulate the tracing results for each ray. $E(u, v)$ denotes the complex light field.

$$E(u,v) = \sum_{n=0}^{m} \frac{A_n}{r_n} \exp\left[i\frac{2\pi}{\lambda} r_n\right] \tag{4}$$

$$r_n = \sqrt{(x_n - u)^2 + (y_n - v)^2 + z_n^2} \tag{5}$$

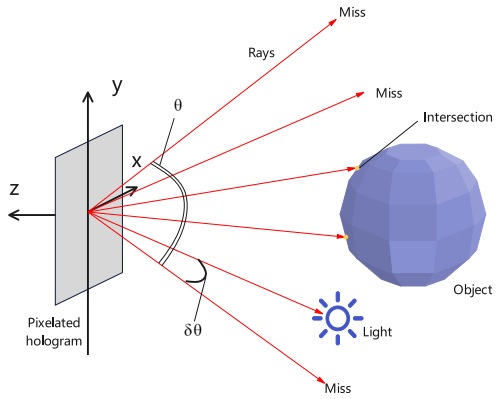

Fig. 6. Ray emission and intersection.

Here, m denotes the number of virtual rays striking the objects in the virtual scene. A_i represents the light amplitude acquired by a single virtual ray impinging upon the virtual scene. λ is the light wavelength, while r is the closest hit by a single virtual ray to the virtual scene. As demonstrated in (5), x_n, y_n, and z_n individually denote the horizontal, vertical, and depth coordinates of each intersection, whereas u and v specify the horizontal and vertical coordinates of every pixelated hologram, respectively.

During the process of path tracing, we can compute the point on the object where a light ray emitted from each pixel hits. From this, we can calculate r_i. Meanwhile, path tracing can obtain the object amplitude of this pixel sample through the formula 2.

We use the Single-Fast Fourier Transform(S-FFT) method to generate the CGH, and the complex wavefield of hologram plane $H(x, y)$ is given by

$$H(\xi,\eta) = \mathcal{F}\left\{E(u,v)\times\exp\left\{i\frac{\pi}{\lambda z}\times\left(\left(u\frac{\lambda z}{Mp}\right)^2+\left(v\frac{\lambda z}{Np}\right)^2\right)\right\}\right\} \times\frac{\exp\left\{i\frac{2\pi}{\lambda}z\right\}}{i\lambda z}\times\exp\left\{i\frac{\pi}{\lambda z}\left(\xi^2+\eta^2\right)\right\} \quad (6)$$

In the (7), $\mathcal{F}$ represents the Fourier transform, $E(u, v)$ is the complex wave field on the object plane, λ is the wavelength, z is the propagation distance, p is the pixel width of the spatial light modulator, and $M \times N$ represents the number of samples on the object plane.

Upon obtaining the complex wavefield of the hologram, image data can be encoded using various schemes to produce a CGH. Common encoding techniques encompass detour phase coding, Burch coding, W.H.LEE coding [32], and kinoform, among others. In this study, we employ the kinoform method [33] for CGH generation. Finally, we encode the complex amplitude function in a phase-only manner and record it onto a grayscale image.

$$Phase(\xi,\eta) = \arctan\{H(\xi,\eta)\} + \pi \quad (7)$$

In each iteration of path tracing, we cast a single ray per pixel into the scene to collect information sampled from the virtual scene, i.e., computing the light field function. In the next iteration of path tracing, we similarly allow each pixel to cast a single ray into the scene, but this time, the sampled information from the virtual scene is accumulated by adding it to the previous result of the light field function. Each sampled ray enters the scene from the center of the sampled pixel, within a spherical surface at an angle θ with respect to the z-axis. The sampling spacing satisfies $\delta\theta$.

Since the light intensity L_o can be superimposed, we can reuse the previously calculated L_o without needing to obtain all L_o at the same time. Therefore, each time we generate a computational hologram, we only sample one ray to obtain its L_o and φ, and then use the pre-calculated L_o for superimposition. Afterwards, we calculate the contribution of this ray using (4) and add it to $E(u, v)$. Overall, this reduces computational complexity and speeds up the computation time.

TABLE I
SPECIFICATION OF GPU

Product model of GPU	GeForce RTX 3080
Memory size	10 GB
Memory type	GDDR6X
Memory bus	320 bit
GPU clock	1440-1710MHz
Memory clock	19000MHz
CUDA units	8704

TABLE II
SPECIFICATION OF IMPLEMENTATION

Wavelegth	0.000632 mm
SLM	HDSLM36R Lite
modulation mode	phase-only
Resolution of SLM	4096×2240 (3840×2160)
Resolution of Hologram	2048×2048
Sampled Rays of each pixel	256
SLM pixel pitch	0.0036 mm
Size of SLM window	15.6 mm × 9.0 mm
Size of hologram	7.4mm × 7.4mm
Size of reconstructed image	12.6 mm × 12.6 mm
Diffraction distance	50mm

III. IMPLEMENTATION

In our approach, the path tracing based CGH is generated within the Unity3D engine, employing C# script programming and the DXR pipeline. The computation platform relies on the Microsoft Windows 10 21H1 operating system, utilizing an AMD Ryzen 7 5700 G processor and an NVIDIA GeForce RTX 3080 GPU, as outlined in Table I.

We utilize the light path depicted in Fig. 7 for optical reconstruction of the CGH. The laser first traverses a horizontal polarizer, followed by a beam expansion system. Subsequently, it passes through a beam spliter and impinges on the SLM. The SLM employed is an HDSLM36R Lite from UPOLabs. After SLM modulation, the beam returns to the beam spliter and is reflected towards the camera for detection. The camera used is a Canon 850D. Direct holographic imaging occurs on the receiving sensor of the camera without passing through any lenses. The implementation parameters' specifications are presented in Table II. Since the spatial light modulator operates solely on phase modulation, the complex wavefield function was encoded

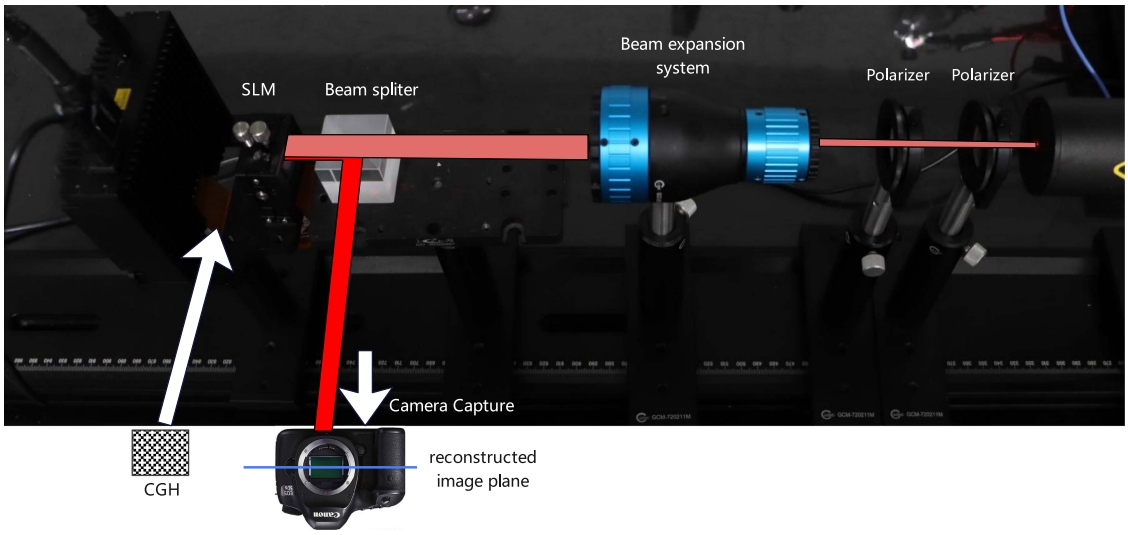

Fig. 7. Light path of CGH reconstruction.

as a phase hologram. This approach resulted in high diffraction efficiency and facilitated the generation of high-quality reconstruction images. We used a relatively short diffraction distance of about 40 mm during the capture. Therefore, the distance between the SLM and the beam spliter is very close. During the capture, the camera is also positioned very close to the beam spliter.

We conducted experiments using CGHs of size 2048×2048. This study focuses specifically on generating CGHs of square shapes with an exponential width of base 2, i.e., 1024×1024 and 2048×2048. The reason for this is that when performing fast Fourier transform using butterfly transformation, it can improve the computational efficiency and speed of square shapes with an exponential width of base 2. The pixel pitch of the CGH is consistent with the pixel pitch of the SLM.

To show our experimental device in detail, we have plotted the Fig. 7 to show the relative position of our lightpath and each device. After the CGH is loaded into the SLM, camera capture reconstruction images in reconstruction of surface directly.

The program's functionality is primarily implemented in two sections, as illustrated in Fig. 8. First, the Central Processing Unit (CPU) section manages the setup of the scene, camera, objects, and transfers parameters to the GPU, which subsequently invokes the ray tracing function, TraceRay(). Secondly, the GPU emits rays into the scene, samples the material properties of the objects intersected by the rays (i.e., BSDF Fig. 9), computes the amplitude and phase information, and encodes them to generate a CGH. Each CGH obtained from the calculation is accumulated with the previous one. The accumulated result is then output to the output buffer, displayed on the screen.

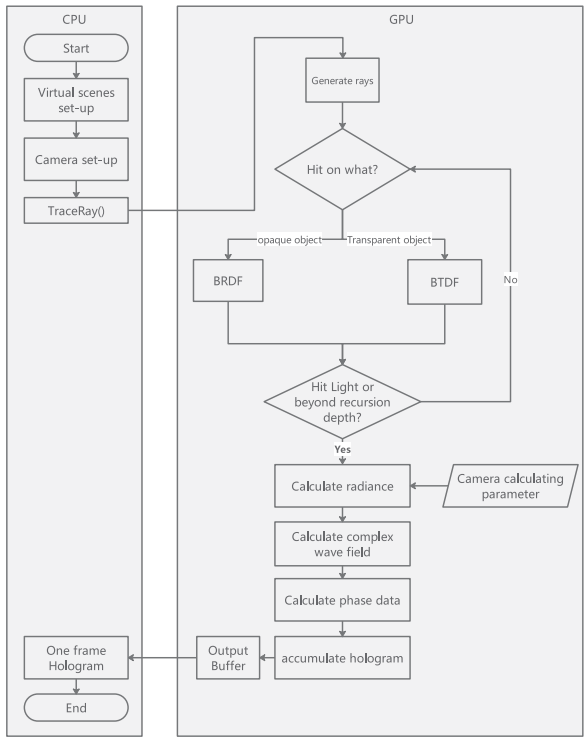

Fig. 8. Flow chart of our proposed method.

IV. Experiment

We initially conducted a numerical reconstruction utilizing a computational device. We compared our proposed approach with the ground truth using the point cloud method. We assessed the reconstruction image quality of CGHs generated by both the point cloud method and the path-tracing method. From the point cloud method, the complex amplitude distribution $E_p(u, v)$ of the CGH can be obtained by the following equation:

$$E_p(u,v) = \sum_{j=0}^{n} \frac{A_j}{r_j} \exp\left[i\left(\frac{2\pi}{\lambda} r_j + \phi_i\right)\right] \tag{8}$$

$$r_j = \sqrt{(u - x_j)^2 + (v - y_j)^2 + z_j^2} \tag{9}$$

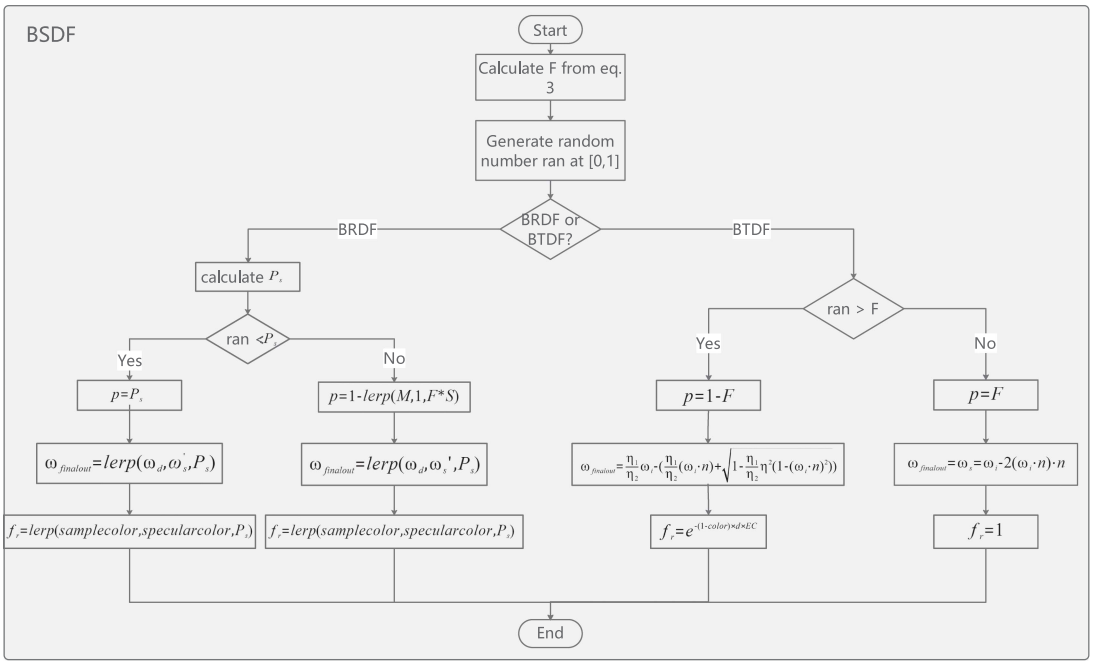

Fig. 9. Flow chart of BSDF.

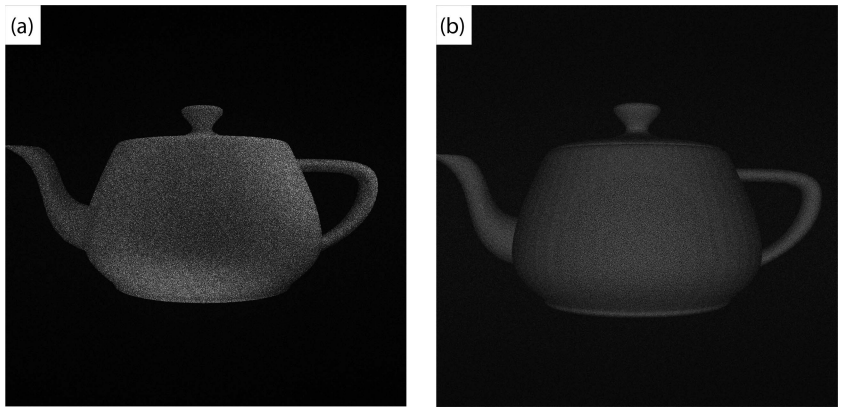

Fig. 10. Numerical reconstruction of the teapot scene's CGH generated using two different approaches: (a) the Proposed Method, and (b) the Point Cloud Method.

where, n represents the object points, (x_j, y_j, z_j) represents the coordinates of the points in the object, A_j means the intensity of the light emitted and ϕ_i is the initial phase angle.

We compared our method with the point cloud method as the ground truth. As shown in Fig. 10, the results of the two methods are quite similar. Our method has a generation time of 10.8 ms, while the point cloud method takes 35.9 ms for generation. We can see from the diagram that there is a slight difference in the reconstructed images of the hologram between the two different implementation methods. The reason for this might be that in Fig. 10(a), we need to calculate the effects of ambient light sources from different directions, while in Fig. 10(b), a constant ambient light coefficient is considered, resulting in a more uniform image. Both diffraction calculations in Fig. 10(a) and (b) are performed using fast Fourier transform.

To demonstrate the off-axis reconstruction capability of holograms generated using our method, we apply a spatial aperture of 1/4 size and perform reconstruction using S-FFT. The model, aperture, and reconstructed image used are shown in Fig. 11. It can be observed that our method enables off-axis numerical reconstruction of holograms.

To test the effectiveness of our method, we used mitsuba models to generate CGHs and reconstruct them, focused on both nearby and back objects, as shown in the Fig. 12. In Fig. 12(a) and (b) show the numerical reconstructed images at different diffraction distances from Matlab, and (c-d) show the corresponding optical reconstructed images. The diffraction distance in the figure is the distance from the spatial light modulator to the camera for the laser beam. The diffraction distance in Fig. 12(a) is 40 mm, and the diffraction distance in Fig. 12(b) is 50 mm.

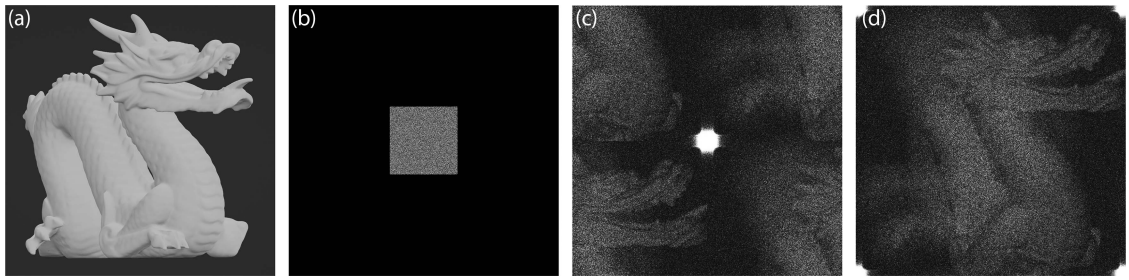

Fig. 11. Off-axis interference hologram reconstruction: (a) original virtual model object, (b) spatial aperture of a quarter of whole size, (c) numerical reconstruction of the hologram and (d) shift the zero-frequency component of (c) to the center of the spectrum.

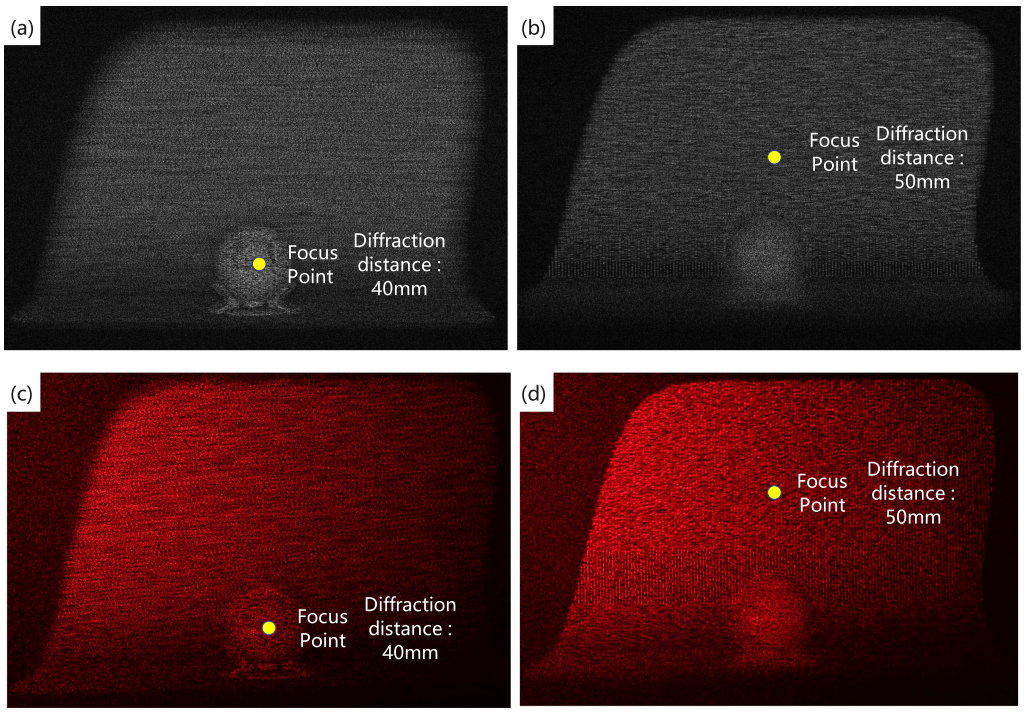

Fig. 12. (a), (b) Numerical reconstructed images at different diffraction distances and (c), (d) optical reconstruction images at different diffraction distances.

Therefore, the distance between the beam spliter and the SLM is very close in order for the camera to capture the reconstructed image corresponding to the diffraction distance.

We selected several representative models to generate CGHs using our method and performed reconstruction. The path tracing-rendered images of virtual scenes are presented in Fig. 13(a), while the optical reconstruction of a CGH from the same scenes is depicted in Fig. 13(b). Examining Fig. 13(a) from left to right, the rendering images correspond to the teapot, rabbit, and dragon scenes within the virtual environment. Because a three-dimensional object is composed of points, the probability of each sampled ray intersecting with a point on the object is extremely low during computation. However, for the rays themselves, the probability of hitting the surface of the object formed by its points is much higher and easier to calculate. In this process, we are not actually sampling object points of the object itself, but points on the plane of the object mesh. This object point, which does not exist in the point cloud method, is also a point that should be on the object. Fig. 13(b) displays the optically reconstructed images of the CGH for each scene shown in (a).

We incorporated a transparent cylinder between the camera and the opaque objects in the three previously tested scenes (Fig. 13) to evaluate the imaging effect of transparent elements. To demonstrate the BTDF lighting model in our method more clearly, we have positioned a transparent cylinder between the camera and the object. This transparent cylinder is used to detect the BTDF model when the light hits transparent objects, which causes either specular reflection or refraction of the light. The relative positioning of the camera, transparent cylinder, and object is illustrated in Fig. 14. Light rays passing through the transparent cylinder would adhere to the model, experiencing refraction or specular reflection, which enabled them to continue tracing the light source and reveal objects situated behind the cylinder. The conclusive outcome demonstrated the transparent cylinder's efficacy in refracting light rays. Fig. 15(a) features the rendered images of the same objects, accompanied by a transparent cylinder situated between

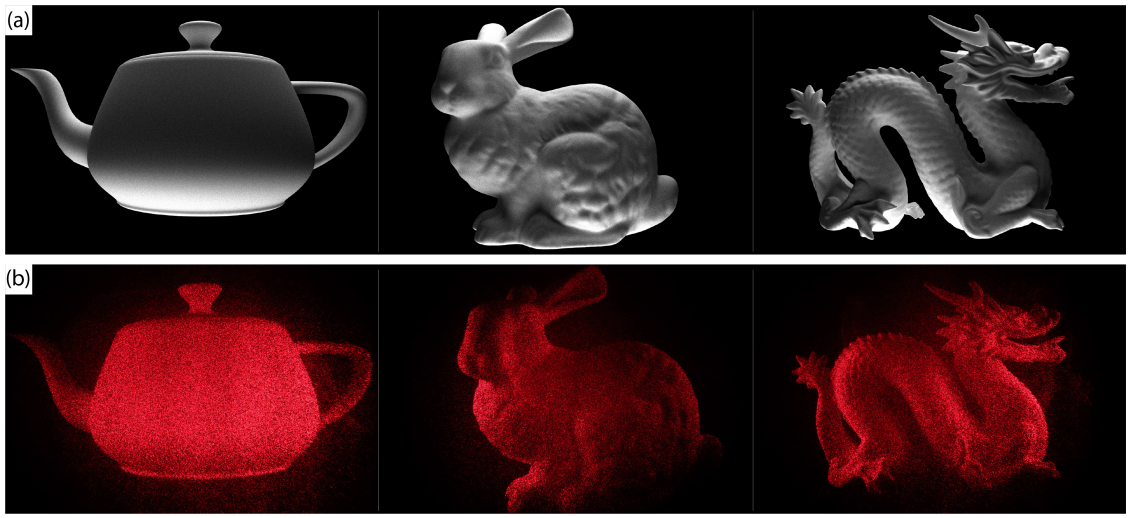

Fig. 13. (a) Various 3D virtual scenes each consisting of a single, different object including a teapot, a bunny, and a dragon arranged from left to right; (b) the optically reconstructed images of the respective scene objects.

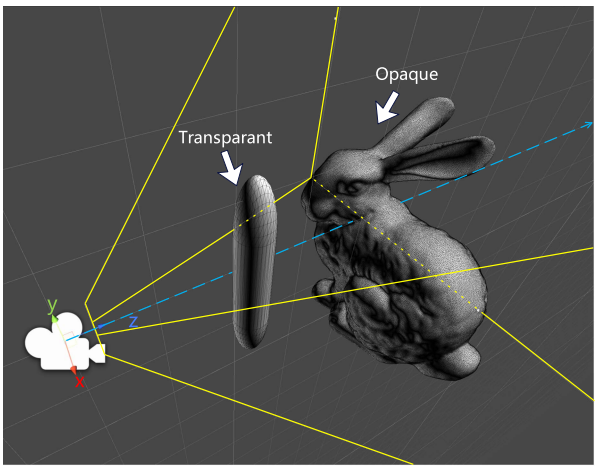

Fig. 14. Camera and the transparent cylinder, along with a diagram showing their relationship with the object, are depicted. The yellow lines represent the camera's field of view, with the z-axis direction indicating the camera's orientation. In order to demonstrate the BTDF model of light, the transparent cylinder is placed between the object and the camera. When the light hits the cylinder, the BTDF model is used.

the objects and the camera. Lastly, Fig. 15(b) presents the CGH optical reconstruction of the corresponding scenes illustrated in Fig. 15(a). We aimed to assess the models presented in Chapter 2, which encompass both opaque and transparent objects.

In order to compare the time complexity required by the same path tracing method in scenes with identical structures composed of the same objects, we employed the proposed method and constructed a new model scene with the same object layout as described in the article [6], as shown in Fig. 16. We generated a hologram of the scene using computational techniques and performed numerical reconstruction. The article [6] proposed an algorithm combining the point-based CGH with the accurate shading and flexibility of ray-tracing algorithms. The time taken by our method to generate the hologram is approximately 10 ms, whereas the generation time mentioned in the article [6] was 1 h and 50 minutes.

To compare the time complexity of our proposed method, we conducted tests using our device environment for the point cloud method, ray tracing method [7]. The ray tracing method proposed volume representation and improved layered ray tracing algorithm for real-time CGH generation. In All tests were performed using GPU-based parallel computing. Among the three methods, the point cloud method is computed and generated using formulas 8 and formula 9, while the ray tracing method utilizes a CGH generation technique based on volumetric representation and an improved ray tracing algorithm. We evaluated three methods for generating CGHs across three distinct scenes, as depicted in Table III. For a consistent comparison, we generated holographic reconstructions for all three methods using a resolution of 1024×1024. These scenes comprise different numbers of physical points. Since the point cloud method is not compatible with ray tracing, this method does not include ray tracing time. The table lists the initialization & ray tracing rendering time, wave field calculation time, and total calculation time for all three methods. The components of the initialization and ray tracing rendering time encompass scene and object initialization coupled with the computational time for all rays. Meanwhile, the wave field calculation time incorporates FFT execution time, data synchronization period, and the time complexity of CGH encoding.

We assessed the traditional point cloud method, the layer-based ray tracing method [7], and the method introduced in this article. We tested the layer-based ray tracing method on our experimental setup, employing 256 layers to compute three scene models, consistent with the other two methods. The results in Table III are in line with previous findings. The discrepancies may stem from our GPU device operating in silent priority mode instead of performance mode. Table III contains data collected using the GPU's silent mode.

As demonstrated in Table III, the point cloud method performed optimally for the teapot scene, with the fastest generation

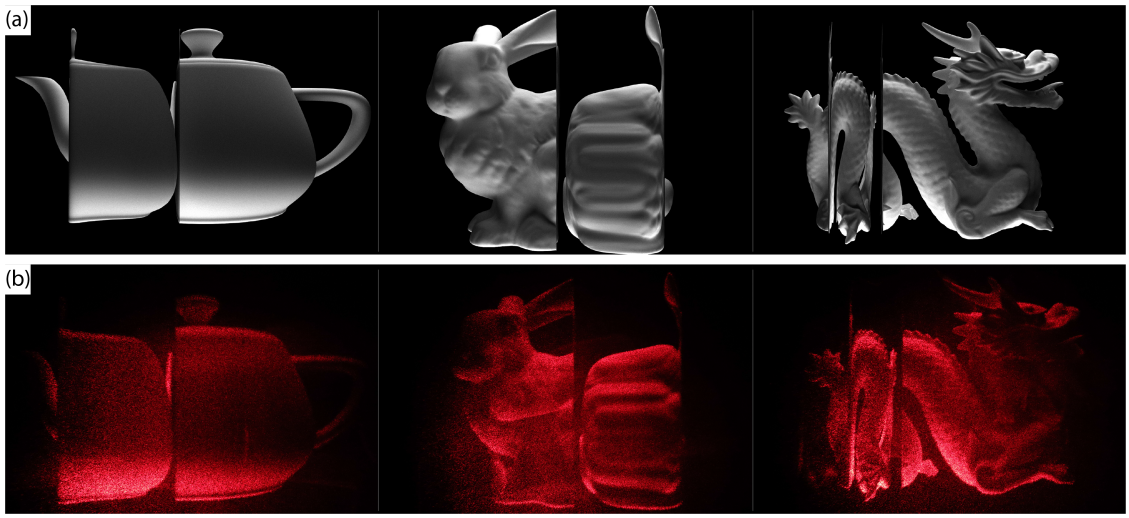

Fig. 15. (a) Based on the scene in Fig. 13(a), place a transparent cylinder in front of the object and (b) the optically reconstructed images of the corresponding scene objects, including the transparent cylinder.

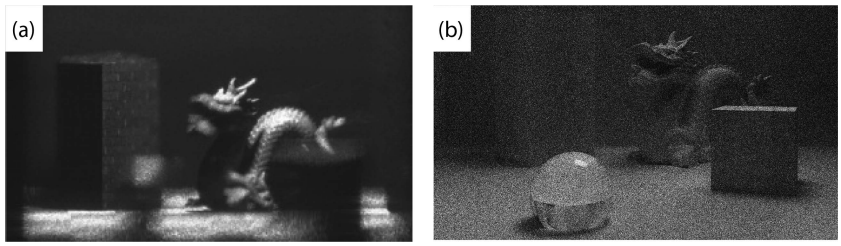

Fig. 16. Numerical reconstruction comparison between (a) using point cloud with global illumination [6] and (b) Proposed Method.

TABLE III
GENERATION TIME COMPARISON OF TWO METHODS IN DIFFERENT SCENES IN 1024 × 1024

Algorithm	Scene	Points of objects	Initialization & Ray rendering time / ms	complex amplitude & FFT / ms	Total time / ms
Point Cloud Method	teapot	3245	-	35.1	35.9
	bunny	99601	-	412	416
	dragon	438956	-	1,834	1,851
Ray Tracing	teapot	3245	13.2	28.2	41.5
	bunny	99601	13.6	28.5	42.1
	dragon	438956	13.3	28.5	41.7
Ours	teapot	3245	0.9	9.9	10.8
	bunny	99601	1.0	10.0	11.0
	dragon	438956	1.0	10.3	11.4

time of 35.9 ms. However, in scenes with more physical points, such as the bunny and dragon scenes, generation times increased to 416 ms and 1851 ms, respectively, difficult to meet the standard of real-time generation of CGHs, which is approximately 33 ms for single CGH. The layer-based ray tracing method maintained a stable overall generation time of approximately 40 ms, but its ray sampling resulted in an extended ray tracing time of around 13 ms. Moreover, this method entails summing complex optical fields for each layer on the holographic recording plane during wavefield calculation, necessitating roughly 28 ms. Our proposed method features advancements in ray tracing afforded by accurate object material implementation and single-ray sampling strategies. The ray samples from the hologram pixel to the objects and lights in the virtual scenes. In the way the ray samples, we can get the length of ray and the amplitude of the pixel. Our method also forgoes calculations of complex optical fields for each layer, expediting light field computations and the overall generation process. Our suggested method aligns with real-time generation standards and rapidly produces holographic reconstructions.

In order to compare the computational speed difference between our proposed method and other ray tracing methods, we calculated the total computation time of several methods based on the same scene at different resolutions. Among them, the layer-based ray tracing method refers to the method proposed in the article [7], while the traditional ray tracing method refers to

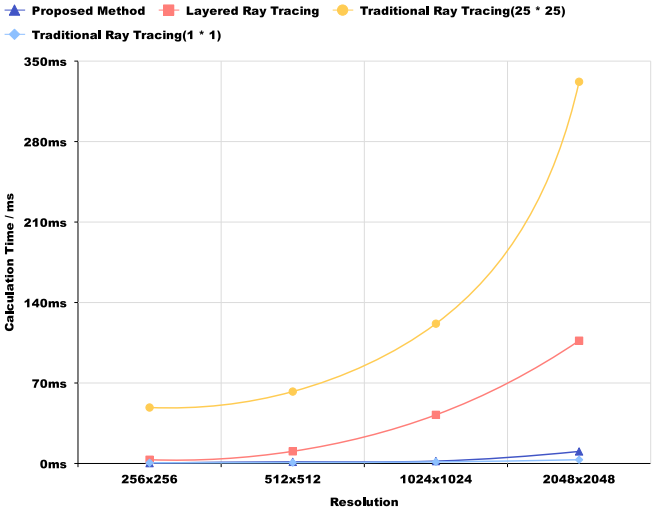

Fig. 17. Calculation time of different methods in different resolutions.

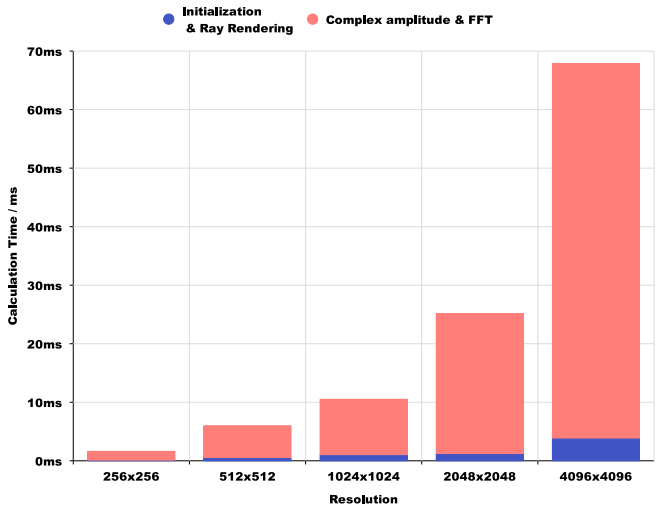

Fig. 18. Calculation time of proposed method in different procedures and resolutions.

the method described in the article [5]. In Fig. 17, the calculation time of different methods is compared. We conducted a comparative analysis of various ray tracing techniques. One approach utilized the backward ray tracing method presented in [5]. By re-implementing the code and deploying it on the same GPU testing device using 25×25 rays and 1×1 rays, this technique was classified as the traditional ray tracing method in Fig. 17. A separate method employed the program from [7] and adjusted the layer count to 256, denoted as the layered ray tracing method in Fig. 17.

Finally, we have our proposed method. We tested the computation time of three different methods for ray tracing with different resolutions. The single-ray sampling method showed a similar computation time to our proposed method. However, our method achieves shorter computation time by employing a single-ray sampling technique, adjusting the BSDF, and utilizing butterfly computation from FFT.

We test the calculation time of different procedures using proposed method in Fig. 18. We tested the initialization and single ray time and the wavefield function computation time. The ray rendering time refers to the time consumed by the ray tracing method for ray generation, intersection with objects, and computation of pixel colors. The computation time of the complex amplitude & FFT is composed of fast Fourier transform and complex wave field superposition. We conducted a comparative analysis of the computational time utilized by distinct stages across multiple resolutions. Our findings revealed that once the illumination model was optimized and the sampling quantity adjusted, the combined time of initialization and ray tracing rendering constituted a smaller fraction of the total computation time. Conversely, the wave field calculation time appeared more extensive, encompassing a larger segment of the complete calculation time.

We attempt to apply our method to the rapid generation of sequences, or more specifically, the rapid generation of videos. At the same time, the process of sequence generation also demonstrates the speed of our computational hologram

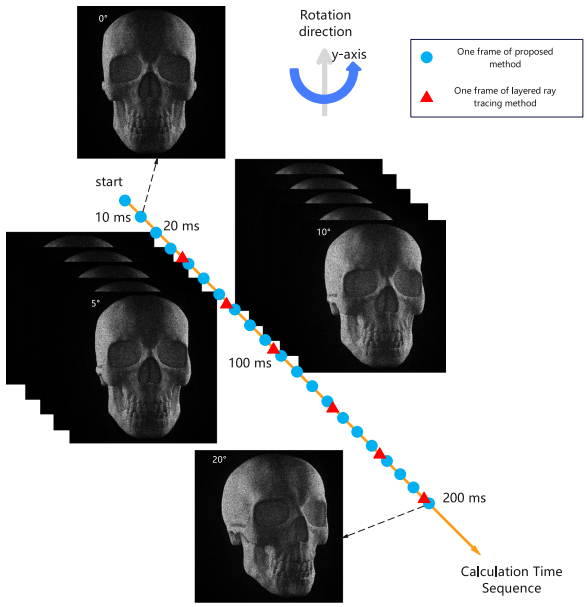

Fig. 19. Sequence of the numerical reconstruction of the computer-generated hologram for a rotating model.

generation to some extent. We plotted a timeline and captured the generated CGHs. We performed a CGH generation test on a skull model, utilizing a temporal sequence to demonstrate CGH generation order, as depicted in the experiment's timing diagram (Fig. 19). The virtual model was rotated at a constant speed while using our proposed method to capture the scene, generate the computer-generated holograms, and perform numerical reconstruction. The method we used to generate this sequence is to sample 5 rays per pixel at a fixed angle of the object, and then rotate the object by 1 °. The object remains stationary during the sampling of pixels. After the sampling is complete, the program rotates the object. After accumulating the results of these 5 ray samples, we perform a fast Fourier transform and generate a computed hologram. Once the computation is completed, the

previously stored accumulated data is cleared, and the process starts again.

V. Conclusion

In conclusion, this study introduces a rapid and high-fidelity holographic generation technique grounded in path tracing, employing GPU parallel computation and path tracing algorithms to expedite the generation of holographic generations. We employed an improved ray tracing method, which enhanced the calculation steps of BSDF. This allows for the efficient computation of the outgoing direction, probability density, attenuation coefficient, and other associated values after light interacts with objects. Additionally, we utilized a method of accumulating single-ray samples, which significantly reduced the sampling time. The experimental findings reveal that the proposed method can fast generate images with a heightened sense of realism and accelerated speed in comparison to the conventional point cloud and layer-based ray tracing techniques, markedly decreasing the time needed to produce holographic reconstructions. This method can generate real-time holographic reconstructions, potentially facilitating the advancement of real-time 3D holographic displays and paving the way for full-color 3D holographic displays in the future.

As for future research, we plan to concentrate on addressing the noise inherent in the path tracing and its CGH generation technique while striving to enhance image quality and further accelerate the generation process.

Acknowledgment

The authors would like to sincerely thank the editor and the reviewers for their rigorous comments and suggestions for improving the quality of the manuscript.

References

[1] B. R. Brown and A. W. Lohmann, "Complex spatial filtering with binary masks," *Appl. Opt.*, vol. 5, no. 6, pp. 967–969, Jun. 1966.

[2] Z. Mei, W. Feiyue, G. Zhen, T. Lele, and W. Xiao, "Overview and development of true 3D display technology: Principles and perspectives," *Pattern Recognit. Artif. Intell.*, vol. 35, no. 8, pp. 701–717, 2022.

[3] P. W. M. Tsang, T.-C. Poon, and Y. M. Wu, "Review of fast methods for point-based computer-generated holography [invited]," *Photon. Res.*, vol. 6, no. 9, pp. 837–846, 2018.

[4] T. Ichikawa, K. Yamaguchi, and Y. Sakamoto, "Realistic expression for full-parallax computer-generated holograms with the ray-tracing method," *Appl. Opt.*, vol. 52, no. 1, pp. A201–A209, 2013.

[5] Y. Wang, Z. Chen, X. Sang, H. Li, and L. Zhao, "High-efficiency photorealistic computer-generated holograms based on the backward ray-tracing technique," *Opt. Commun.*, vol. 410, pp. 768–773, 2018.

[6] D. Blinder, M. Chlipala, T. Kozacki, and P. Schelkens, "Photorealistic computer generated holography with global illumination and path tracing," *Opt. Lett.*, vol. 46, no. 9, pp. 2188–2191, 2021.

[7] C. Zhong, X. Sang, B. Yan, H. Li, D. Chen, and X. Qin, "Real-time realistic computer-generated hologram with accurate depth precision and a large depth range," *Opt. Express*, vol. 30, no. 22, pp. 40087–40100, 2022.

[8] J. Xiaoyu, G. Jinbin, L. Chao, Z. Liting, and L. Liyu, "Progress of fast generation algorithm of computer-generated hologram based on point source model," *Laser Optoelectron. Prog.*, vol. 55, no. 10, pp. 100005-1–1000005-10, 2018.

[9] J. Jia et al., "Reducing the memory usage for effectivecomputer-generated hologram calculation using compressed look-up table in full-color holographic display," *Appl. Opt.*, vol. 52, no. 7, pp. 1404–1412, 2013.

[10] D. Pi, J. Liu, R. Kang, Z. Zhang, and Y. Han, "Reducing the memory usage of computer-generated hologram calculation using accurate high-compressed look-up-table method in color 3D holographic display," *Opt. Exp.*, vol. 27, no. 20, pp. 28410–28422, 2019.

[11] Y. Mori and T. Nomura, "Fast hologram pattern generation by adaptive point-spread spherical wave synthesis," *J. Display Technol.*, vol. 12, no. 8, pp. 815–821, 2016.

[12] T. Sugawara, Y. Ogihara, and Y. Sakamoto, "Fast point-based method of a computer-generated hologram for a triangle-patch model by using a graphics processing unit," *Appl. Opt.*, vol. 55, no. 3, pp. A160–A166, 2016.

[13] D. Arai, T. Shimobaba, T. Nishitsuji, T. Kakue, N. Masuda, and T. Ito, "An accelerated hologram calculation using the wavefront recording plane method and wavelet transform," *Opt. Commun.*, vol. 393, pp. 107–112, 2017.

[14] X. Jin, J. Gui, Z. Jiang, G. Wang, and Y. Lou, "Fast calculation of computer generated hologram using multi-core CPUs and GPU system," *Proc. SPIE*, vol. 10818, pp. 310–316, 2018.

[15] Z. W. Xie, J. L. Zang, and Y. Zhang, "Accelerated algorithm for three-dimensional computer generated hologram based on the ray-tracing method," *J. Modern Opt.*, vol. 60, no. 10, pp. 797–802, 2013.

[16] L. Wei and Y. Sakamoto, "A fast animation generation system for computer-generated hologram using ray-tracing method," in *Proc. IEEE Int. Workshop Adv. Image Technol.*, 2018, pp. 1–4.

[17] H. Li et al., "Optimized layered method for real-time interactive holographic display based on ray-tracing technique," *Opt. Eng.*, vol. 59, no. 10, 2020, Art. no. 102408.

[18] Y. Wang, X. Sang, Z. Chen, H. Li, and L. Zhao, "Real-time photorealistic computer-generated holograms based on backward ray tracing and wavefront recording planes," *Opt. Commun.*, vol. 429, pp. 12–17, 2018.

[19] M. Sun, Y. Yuan, Y. Bi, S. Zhang, J. Zhu, and W. Zhang, "Acceleration and expansion of a photorealistic computer-generated hologram using backward ray tracing and multiple off-axis wavefront recording plane methods," *Opt. Exp.*, vol. 28, no. 23, pp. 34994–35005, 2020.

[20] Z. Sun et al., "Real-time interactive computer-generated hologram using fresnel zone plate extension method," *Proc. SPIE*, vol. 12318, pp. 335–340, 2022.

[21] L. Wei and Y. Sakamoto, "Fast calculation method with foveated rendering for computer-generated holograms using an angle-changeable ray-tracing method," *Appl. Opt.*, vol. 58, no. 5, pp. A258–A266, 2019.

[22] T. Whitted, "An improved illumination model for shaded display," *Commun. ACM*, vol. 23, no. 6, pp. 343–349, 1980.

[23] L. Fascione et al., "Path tracing in production," in *Proc. ACM SIGGRAPH Courses*, 2018, pp. 1–79.

[24] J. T. Kajiya, "The rendering equation," in *Proc. 13th Annu. Conf. Comput. Graph. Interactive Techn.*, 1986, pp. 143–150.

[25] T. Huwe and R. Hemmerling, "Stochastic path tracing on consumer graphics cards," in *Proc. 24th Spring Conf. Comput. Graph.*, 2008, pp. 105–111.

[26] K. Schwenk and T. Drevensek, "Radiance filtering for interactive path tracing," in *Proc. ACM SIGGRAPH Posters*, 2012, p. 1.

[27] V. Rao, D. Sinha, N. Srimal, and P. K. Maurya, "Statistical path tracing in timing graphs," in *Proc. 53 rd Annu. Des. Automat. Conf.*, 2016, pp. 1–6.

[28] D. Cline, J. Talbot, and P. Egbert, "Energy redistribution path tracing," in *Proc. ACM SIGGRAPH Papers*, 2005, pp. 1186–1195.

[29] A. Polychronakis, G. A. Koulieris, and K. Mania, "Emulating foveated path tracing," in *Proc. 14th ACM SIGGRAPH Conf. Motion, Interaction Games*, 2021, pp. 1–9.

[30] M. Jaroš, L. Říha, P. Strakoš, and M. Špetko, "GPU accelerated path tracing of massive scenes," *ACM Trans. Graph.*, vol. 40, no. 2, pp. 16:1–16:17, 2021.

[31] A. Appel, "Some techniques for shading machine renderings of solids," in *Proc. Spring Joint Comput. Conf.*, 1968, pp. 37–45.

[32] W.-H. Lee, "Binary synthetic holograms," *Appl. Opt.*, vol. 13, no. 7, pp. 1677–1682, Jul. 1974.

[33] L. Lesem, P. Hirsch, and J. Jordan, "Kinoform–A new wavefront reconstruction device," *IBM J. Res. Develop.*, vol. 13, no. 2, pp. 150–155, 1969.