Research Article | Vol. 29, No. 9 / 26 April 2021 / Optics Express 12849
Optics EXPRESS

GPU-accelerated calculation of computer-generated holograms for line-drawn objects

TAKASHI NISHITSUJI,[1,*] DAVID BLINDER,[2,3] TAKASHI KAKUE,[4] TOMOYOSHI SHIMOBABA,[4] PETER SCHELKENS,[2,3] AND TOMOYOSHI ITO[4]

[1]Faculty of Systems Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo, Japan
[2]Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), Pleinlaan 2, B-1050 Brussel, Belgium
[3]IMEC, Kapeldreef 75, B- 3001 Leuven, Belgium
[4]Graduate School of Engineering, Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba, Chiba, Japan
*nishitsuji@tmu.ac.jp

Abstract: The heavy computational burden of computer-generated holograms (CGHs) has been a significant issue for three-dimensional (3D) display systems using electro-holography. Recently, fast CGH calculation methods of line-drawn objects for electro-holography were proposed, which are targeted for holography-based augmented reality/virtual reality devices because of their ability to project object contours in space with a small computational load. However, these methods still face shortcomings, namely, they cannot draw arbitrary curves with graphics processing unit (GPU) acceleration, which is an obstacle for replaying highly expressive and complex 3D images. In this paper, we propose an effective algorithm for calculating arbitrary line-drawn objects at layers of different depths suitable for implementation of GPU. By combining the integral calculation of wave propagation with an algebraic solution, we successfully calculated CGHs of $1,920 \times 1,080$ pixels within 1.1 ms on an NVIDIA Geforce RTX 2080Ti GPU.

1. Introduction

With the growing demand for augmented reality and virtual reality technologies, there is a daily increase in the expectations from three-dimensional (3D) imaging technologies. Among the possible implementations of 3D displays, electro-holography is the most promising technology for photo-realistic 3D displays, owing to its ability of reproducing the complete light wave reflected from a 3D object. However, the large computational load required to calculate the computer-generated holograms (CGHs) is a significant issue for realizing practical holographic display systems.

Till date, many algorithms for fast calculations of CGH have been developed, including the look-up table method [1–7], ray-wavefront conversion methods [8,9], wave-front recording plane methods [10–13], deep-learning-based methods [14], layer-based methods [15–17], saccade suppression [18], and sparsity-based methods [19–21]. Hardware-based acceleration have also been proposed for devices, such as field-programmable gate arrays [22–24] , application specific integrated circuit [25] and GPU [26,27].

Recently, we proposed an algorithm for fast calculation of CGH (CG-line method) for line-drawn objects at layers of different depths [28]. Figure 1 provides an overview of our previous method. The algorithm is primarily based on the phenomenon that wavefronts created from point-light sources (PLSs) aligned on a line segment placed at a constant depth converge to a one-dimensional (1D) signal, provided the line length is sufficiently long. The CG-line method approximates the wavefronts created from arbitrary shapes consisting of constant-depth

#421230 https://doi.org/10.1364/OE.421230
Journal © 2021 Received 11 Feb 2021; revised 1 Apr 2021; accepted 5 Apr 2021; published 12 Apr 2021

Optics EXPRESS

line-drawn objects to synthesize the precalculated converged 1D wavefront along the normal direction of the line object. Although the proposed method can calculate CGHs at high speed even with consumer CPUs, effective implementation on massive parallel processors such as GPUs is difficult as it requires random memory access to synthesize 1D wavefronts. Since the computational load for this method increases with the complexity of the 3D object, implementing the algorithm on high-speed devices such as GPUs is essential for practical purposes. We also proposed a method for analytically solving the distribution of wavefronts from line-drawn objects suitable for GPU implementation [29]; however, the algorithm could only draw straight lines or arcs that lie at the same depth, which limits the expressiveness of the 3D image.

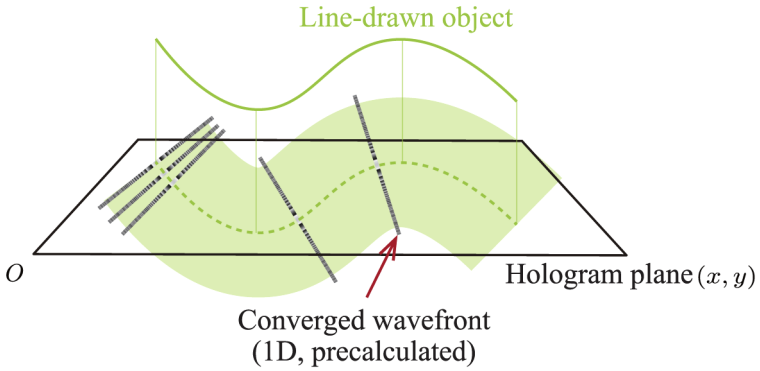

Fig. 1. Overview of CG-line method.

Consequently, we propose a novel algorithm for calculation of CGH for arbitrary line-drawn objects at layers of different depths, suitable for GPU implementation. The primary idea of the proposed method is based on the CG-line method, but it does not use precalculated data to avoid random memory access. In other words, the proposed method calculates the converged wavefronts directly and independently on each pixel of the hologram plane, which is suitable for GPU implementation. As a result, we succeeded in calculating CGHs 3 to 12 times faster than the GPU implementation using [29] for objects consisting of arcs and straight lines with equivalent image quality. Furthermore, the proposed method succeeded in drawing font-shaped line-drawn objects as an arbitrary line-drawn object at sufficiently high speed.

CGH encodes the complex wavefront computed with the wave propagation of digital 3D objects. The spatial light modulator (SLM), which is the digital device to display CGHs, can not modulate the complex amplitude directly in the current technology. There are several encoding methods for CGH (phase-only, amplitude-only, etc.), and the quality of the reproduced image varies depending on the method. The proposed method can be applied to any encoding scheme. In this paper, Kinoform CGH is demonstrated as an example. It is a phase-only coding method that can produce high-quality 3D images due to its high light utilization efficiency.

The remainder of this paper is organized as follows. Section 2 briefly presents the CG-line method. Section 3 describes the details of the proposed method. Section 4 presents the experimental results. Section 5 discusses the results. Finally, Section 6 presents the conclusions of the study.

2. CG-line method for fast generation of CGH for line-drawn objects

The calculations of CGH for a PLS-based 3D model can be regarded as a linear superimposition of the wavefront of each individual PLS. The wavefront of a single PLS located at the coordinates

Optics EXPRESS

$(\delta, \epsilon, \zeta)$ is defined with axial approximation as

$$P(x, y) = a \cdot \exp\left(\frac{i\pi}{\lambda\zeta}\left[(x-\delta)^2 + (y-\epsilon)^2\right]\right)\frac{1}{\zeta}, \tag{1}$$

where, (x, y) are the coordinates of the hologram plane, ζ the distance between PLS and the hologram plane, i the imaginary unit, λ the wavelength of the incident light, and a the amplitude of PLS.

According to Eq. (1), assuming the PLSs lie on the x-axis from $-\infty$ to $+\infty$ on the same ζ, forming a self-illuminated line of infinite length residing at a certain depth, the complex amplitude distribution on the hologram plane $L(x, y)$ is given as

$$L(x, y) = \frac{a}{\zeta}\int_{-\infty}^{\infty} \exp\left(\frac{i\pi}{\lambda\zeta}\left[(x-u)^2 + y^2\right]\right)du, \tag{2}$$

where the amplitudes of all the PLSs are considered to be equal. Applying the Fresnel integral, Eq. (2) becomes a pure function of y.

$$U(y) = \frac{1}{\zeta}\sqrt{\frac{\lambda\zeta}{2}}(i+1)\exp\left(\frac{i\pi y^2}{\lambda\zeta}\right). \tag{3}$$

Therefore, the wavefront of a self-luminous line of infinite length converges to a 1D pattern on the hologram plane. In practice, $U(y)$ does not always need to be calculated due to the diffraction limit of the SLM, which is a display device for CGH. In other words,

$$U(y) = \begin{cases} \text{Eq.(3)} & (y < R_\zeta) \\ 0 & (otherwise) \end{cases}$$

where,

$$R_\zeta = \zeta\frac{\lambda}{\sqrt{4p^2 - \lambda^2}}, \tag{4}$$

and p is the pixel pitch of the SLM.

The primary idea of the CG-line method is to approximate the calculation for the wavefront of the arbitrary line-drawn object by superimposing the 1D wavefront, which is obtained by Eq. (3), along the normal vector of the line-drawn object. Figure 1 provides an overview of the CG-line method. In the original CG-line method [28], the normal vector at each point on the line is calculated and the precalculated 1D wavefront is obtained; thus, it is a point-wise calculation of the line.

Since the computational complexity of the original CG-line method increases with the complexity of the line-drawn objects, a faster implementation is essential for more practical 3D images, which would facilitate the implementation on high-performance massive parallel processors such as GPUs, which are suitable for pixel-wise parallelization. However, since the original CG-line method requires tracing line objects, converting to a pixel-wise parallelization is required. In addition, as mentioned in [28], the blank pixels between the succesively synthesized 1D wavefronts causes image degradation, which also stems from the original CG-line method tracing the line and pasting the 1D wavefront along the normal direction on each point of the line.

Therefore, in this paper, we propose a method to calculate the CG-line method with pixel-wise parallelization and is suitable for GPU implementation. The problem regarding the blank pixels between the pasted 1D wavefronts in [28] can be resolved by our proposed pixel-wise calculations.

Optics EXPRESS

3. Algorithm for pixel-wise parallelization of the CG-line method

The basic idea of the pixel-wise computation of the CG-line method is to find an effective way for calculating the length of the normal vector passing through both a point on the line-drawn curve and a certain point on the hologram plane, which is an input for the 1D function of Eq. (3). Note that line-drawn objects are assumed to be at a constant depth ζ in a single integral calculation.

Figure 2 shows the overview of the algorithm. Given that line-drawn objects are formulated by parametric functions with coordinates $\vec{g}(t) = (x(t), y(t))$, the vector from a specific point on the hologram plane $\vec{r} = (\alpha, \beta)$ to $\vec{g}(t)$ becomes

$$\vec{h}(t) = \vec{g}(t) - \vec{r} = (x(t) - \alpha, y(t) - \beta), \tag{5}$$

where $0 \leq t \leq 1$.

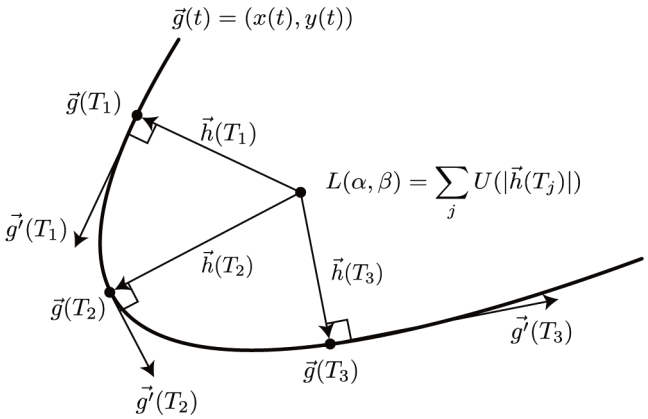

Fig. 2. Overview of the proposed method

The condition for $\vec{h}(t)$ to be a normal vector of the line-drawn curve is,

$$\vec{h}(t) \cdot \vec{g'}(t) = 0, \tag{6}$$

$$\rightarrow \{x(t) - \alpha\}x'(t) + \{y(t) - \beta\}y'(t) = 0. \tag{7}$$

where the apostrophe denotes the derivative with respect to t. Since the degree of Eq. (7) is $2n - 1$ where n is the maximum degree of $x(t)$ and $y(t)$ (assuming these functions are polynomials), Eq. (7) can be analytically solved if $2n - 1 \leq 4$; thus, $n \leq 2$ is preferable.

Given T_j as the j-th real number solution of Eq. (7), the norm of $\vec{h}(t)$ becomes;

$$|\vec{h}(T_j)| = \sqrt{\{x(T_j) - \alpha\}^2 + \{y(T_j) - \beta\}^2}. \tag{8}$$

Thus, from Eq. (3), the complex amplitude distribution at (α, β) becomes;

$$L(\alpha, \beta) = \sum_j U(|\vec{h}(T_j)|). \tag{9}$$

Since Eq. (9) can be computed independently for each pixel of the hologram plane, the above algorithms allow for the CG-line method to be computed in parallel by a massive parallel processor such as the GPU.

Many algorithms for approximating arbitrary curves with parametric functions, such as Bezier and Spline curves, can be considered. Among these, some practical algorithms that the maximum

Optics EXPRESS

degree of the equation is two are existed, such as B-Spline and quadratic Bezier curves [30]. Thus, we propose an algorithm for the quadratic parametric curve function by introducing the quadratic Bezier curve as an example, and it can be easily expanded to other quadratic curve functions. In addition, we develop solutions for the simpler cases of circle and straight line in the following subsections.

3.1. Quadratic parametric curve

First, we define the function for the quadratic parametric curve as:

$$x(t) = at^2 + bt + c, \tag{10}$$

$$y(t) = dt^2 + et + f, \tag{11}$$

where $a, b, c, d, e, f \in \mathbb{R}$. $\vec{h}(t)$ and $\vec{g'}(t)$ become,

$$\vec{h}(t) = (at^2 + bt + c - \alpha, dt^2 + ct + f - \beta), \tag{12}$$

$$\vec{g'}(t) = (2at + b, 2dt + e). \tag{13}$$

Thus, Eq. (7) becomes,

$$(at^2 + bt + c - \alpha)(2at + b) + (dt^2 + et + f - \beta)(2dt + e) = 0, \tag{14}$$

$$\rightarrow At^3 + Bt^2 + Ct + D = 0, \tag{15}$$

where,

$$A = 2(a^2 + d^2), \tag{16}$$

$$B = 3(ab + de), \tag{17}$$

$$C = b^2 + 2a(c - \alpha) + e^2 + 2d(f - \beta), \tag{18}$$

$$D = b(c - \alpha) + e(f - \beta). \tag{19}$$

We can obtain the real number solution T_j as an input for Eq. (3) by solving the cubic equation Eq. (15) using Cardano's formula [31].

Next, we introduce the quadratic Bezier curve as a practical example. The quadratic Bezier curve is defined as [30],

$$x(t) = (1 - t)^2 x_0 + 2(1 - t)t x_1 + t^2 x_2, \tag{20}$$

$$y(t) = (1 - t)^2 y_0 + 2(1 - t)t y_1 + t^2 y_2, \tag{21}$$

where, (x_0, y_0) and (x_2, y_2) are the start and end points of the curve, respectively, and (x_1, y_1) is the control point. According to Eqs. (20) and (21), the coefficients in Eqs. (10) and (11) would become:

$$a = x_0 - 2_x 1 + x2, \tag{22}$$

$$b = -2x_0 + 2x_1, \tag{23}$$

$$c = x_0, \tag{24}$$

$$d = y_0 - 2y_1 + y_2, \tag{25}$$

$$e = -2y_0 + 2y_1, \tag{26}$$

$$f = y_0. \tag{27}$$

Applying these coefficients to Eqs. (16)–(19), and solving Eq. (15) using Cardano's formula to obtain T_j, thereafter calculating Eqs. (8) and (9), we can obtain the complex amplitude distribution at a specific pixel of the hologram plane.

Optics EXPRESS

Figure 3 is an example of the CGH of a line-drawn object drawn by the quadratic Bezier curve and numerically reconstructed image of CGH and its original image. Note that although Fig. 3(a) is a kinoform type of CGH, the proposed method is applicable for any type of CGHs, including amplitude CGH. In this example, a line-drawn object lies on a depth of 0.1 m. As shown in the reconstructed image, the obtained CGH is succeeded in projecting the arbitrary curved line-drawn object in 3D.

Fig. 3. Example of CGH created from a line-drawn object with quadratic Bezier curve: (a) CGH, (b) numerically reconstructed image, (c) original image

3.2. Circles and arcs

Generally, it is difficult to draw a circle or an arc with quadratic parametric curve functions. Hence, we developed the algorithm for drawing the circle and arc independently, and it is simpler than the case of quadratic curves.

Figure 4 shows the overview for a circle. The purpose of the algorithm is to calculate the norm of $\vec{h}(t)$ when it is the normal vector of a line-drawn object. For a circle, the vector connecting the center of the circle and a point is always the normal vector of the circle, as shown in Fig. 4. Further, we should consider two normal vectors, $\vec{h_1}$ and $\vec{h_2}$, whose locations correspond to the near and far sides of the circle. Their norms are then given as follows:

$$|\vec{h_1}| = |\vec{r} - \vec{c}| - R, \tag{28}$$

$$|\vec{h_2}| = |\vec{r} - \vec{c}| + R, \tag{29}$$

where, $\vec{c}$ is the position vector of the center of the circle and R the radius of the circle. Therefore, the complex amplitude distribution at (α, β) becomes

$$L(\alpha, \beta) = U(|\vec{h_1}|) + U(|\vec{h_2}|) \tag{30}$$

For arcs, the complex amplitude distribution can be obtained by determining whether the angles of $\vec{h_1}$ and $\vec{h_2}$ are within the range of the angle of the arc.

3.3. Straight line

If the distance between a point and another on a straight line of infinite length is minimal, the vector connecting those points is the normal. Thus, for a line, it can be summarized as a problem of finding the minimum distance of a point to a line. Figure 5 shows the overview for a straight

Optics EXPRESS

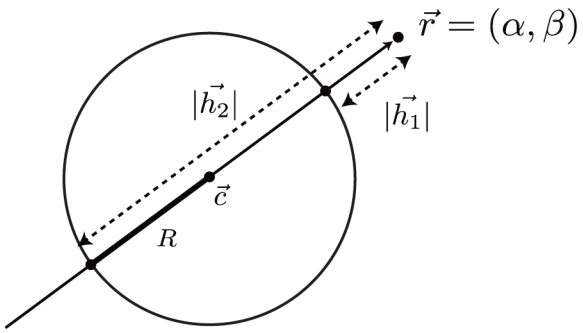

Fig. 4. Overview of obtaining the norm of the normal vector for a circle and arc.

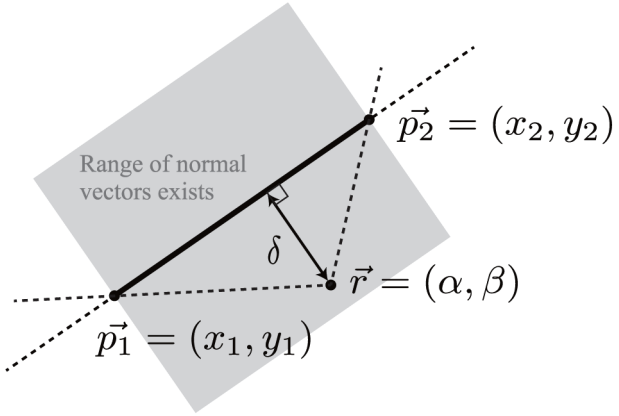

Fig. 5. Overview of obtaining the wavefront for straight-line objects.

line. The straight-line functions are defined as:

$$lx + my + n = 0, \tag{31}$$

where $l, m, n \in \mathbb{R}$. The minimum distance between the point on a line and $\vec{r} = (\alpha, \beta)$ can be obtained as follows:

$$\delta = \frac{|l\alpha + m\beta + n|}{\sqrt{l^2 + m^2}}. \tag{32}$$

When the straight line has a constant length, we should ascertain whether $\vec{r}$ is within the range where the normal vector of the line object exists. Defining the start and end points of a line as $\vec{p_1} = (x_1, y_1)$ and $\vec{p_2} = (x_2, y_2)$, respectively, the coefficients in Eq. (31) become,

$$l = y_2 - y_1, \tag{33}$$

$$m = -x_2 - x_1, \tag{34}$$

$$n = -x_1(y_2 - y_1) + y_1(x_2 - x_1). \tag{35}$$

Because $\vec{r} = (\alpha, \beta)$ is on the normal vector of the line, both the following conditions must be satisfied:

$$(\vec{p_2} - \vec{p_1}) \cdot (\vec{r} - \vec{p_1}) \geq 0, \tag{36}$$

$$(\vec{p_2} - \vec{p_1}) \cdot (\vec{p_2} - \vec{r}) \geq 0. \tag{37}$$

Optics EXPRESS

Therefore, the complex amplitude distribution of $\vec{r}$ that satisfies the condition of Eqs. (36) and (37) can be obtained by,

$$L(\alpha, \beta) = U(\delta). \tag{38}$$

4. Experiments

We investigated the feasibility of the proposed method, considering both the calculation speed and image quality for the numerically and optically reconstructed images. The results are compared with those of three reference methods suitable for GPU implementation, because they can perform independent pixel-wise calculations.

- **conventional point-based method**: directly calculates Eq. (1) for each coordinate of a hologram plane and every PLS.
- **angular spectrum method (ASM)**: calculates the fast Fourier transform (FFT)-based wave diffraction for a specific depth plane, where the PLS exists. The resolution of the plane is the same as that of the hologram plane. Given the number of planes as P, the total number of FFT to be executed becomes $P + 1$.
- **analytical line-drawn method**: solves the integral for calculating the complex amplitude distribution for arcs and straight-line segments that are located on the same depth plane [29].

We also compared the original CG-line method implemented on the CPU for verifying both the effectiveness of massively parallel implementation on GPU and filling the blank pixels that cause image degradation.

We employed three 3D models for the evaluation: (1) the SimpleShape model (Fig. 6(a)), which consists of 6 arcs and 40 lines, and the total line length is 17,777 pixels using 1,920×1,080 pixels (2k) resolution; (2) the Seigaiha model (Fig. 6(b)), which consists of 144 arcs, and the total line length is 29,331 pixels using 2k resolution; (3) the Tokyo model (Fig. 6(c)), which consists of 111 segments of quadratic Bezier curves and 2 straight lines, and the total length of the line

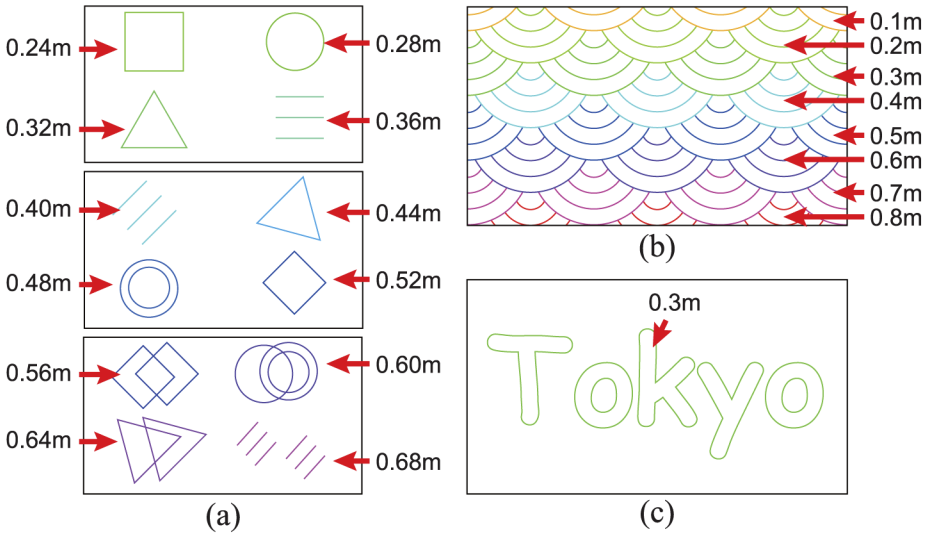

Fig. 6. Dataset: (a) "SimpleShape", (b) "Seigaiha", (c) "Tokyo", the same line color implies the same line depth.

Optics EXPRESS

is 9,876 pixels using 2k resolution. Since the analytical line-drawn method can only draw arcs and lines, we did not evaluate the Tokyo model for this method. Here, the line thickness for all models was 1 pixel in the digital image; i.e., it was assumed to be equal to the pixel pitch of SLM, which is 8 μm in this paper. In addition, the models do not apply to anti-aliasing. Furthermore, for the point-based method, we regarded each pixel of the line as PLS; for example, the 17,777 pixels in the SimpleShape model amounted to 17,777 PLSs. Note that the size of all 3D models is scaled to match the hologram resolution. For example, the 3D image used at 4k resolution is twice as wide and tall as the 3D image at 2k resolution.

We used the following computer environment: Microsoft Windows 10 Professional operating system; AMD Ryzen 9 3950X 3.50 GHz CPU, 64 GB DDR4–2132 memory; Microsoft Visual C++ 2019 compiler with single floating-point computational precision and NVIDIA Geforce RTX 2080Ti GPU with CUDA 11.0. The CGH resolution was set to 2k, 4k, 8k, 16k, 32k for investigating the trend of computational speed and image quality according to the resolution, and we assumed that the wavelength of the incident light to be 532 nm (green light).

Figure 7 shows the optical setup, which consists of a phase-modulation-type SLM (Holoeye Photonics AG, 'HEO1080P'), green laser with a wavelength of 532 nm (Thorlabs inc., 'CPS532'), beam expander (Thorlabs inc., 'GBE10-A'), polarizer (Sigma-koki, 'SPF-50C-32'), and beam splitter (Edmund optics, '47571'). We captured real images directly focused on the image sensor of the camera.

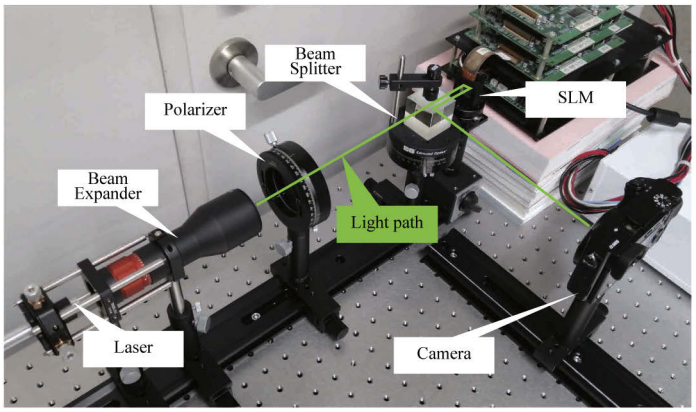

Fig. 7. Optical setup.

Figure 8 provides the CGH of a SimpleShape model created with the proposed method and point-based method in 2k resolution as an example.

4.1. Calculation speed

Table 1 and Fig. 9 shows a comparison of the calculation speed for each dataset and resolution. Here, Fig. 9 shows the total calculation time including the kernel execution and memory access. The ASM could only calculate until 8k resolution because the amount of memory used in the calculation exceeds the GPU's hardware limit. The calculation time of the proposed method was the shortest among all other methods for each 3D model and resolution. For example, it was 132 and 40 times faster than the Point-based and the ASM methods, respectively, for the SimpleShape model at 2k resolution and 5.4 times faster than the analytical method for the Seigaiha model at 2k resolution. The scale of the speed-up grows with increasing resolution. For example, it was 3444 times faster than the Point-based method for the SimpleShape model at 32k resolution.

The computational requirements of the Point-based method are proportional to the total length of all lines, regardless of whether they are curved or not. Therefore, the Seigaiha model takes the

Optics EXPRESS

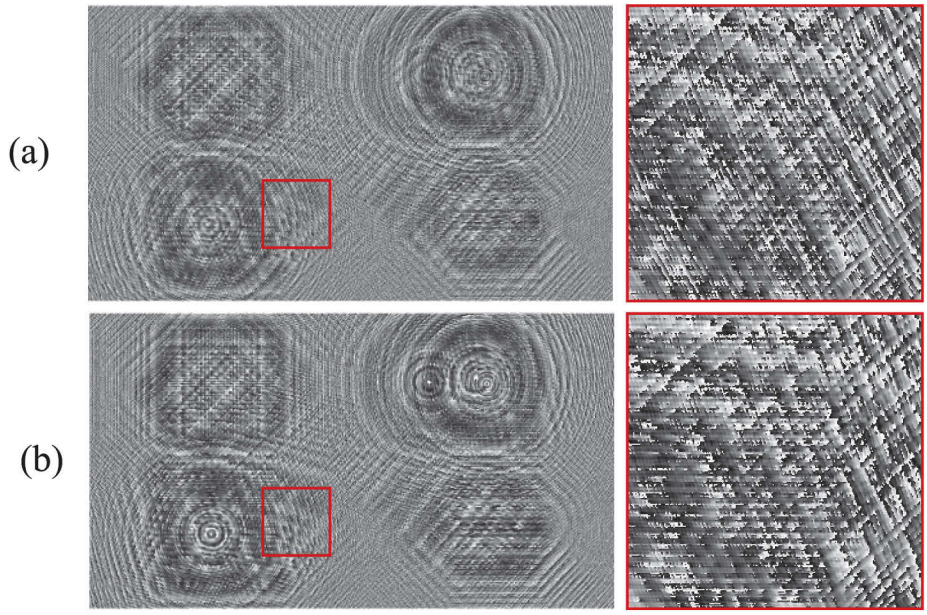

Fig. 8. CGH of SimpleShape model in 2k resolution: (a) proposed method, (b) point-based method.

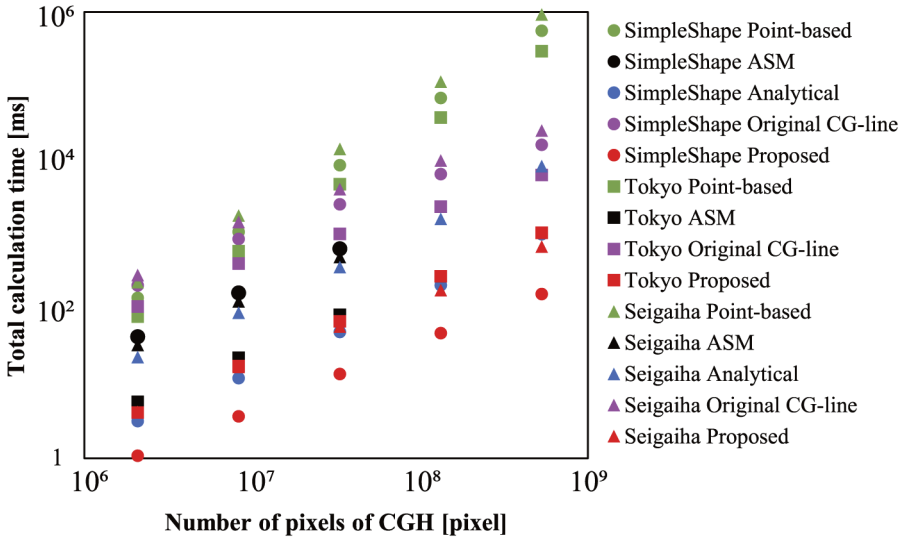

Fig. 9. Trend of calcualtion time for CGH resolution.

Table 1. Time for kernel execution (memory access) at 2k resolution

[ms]	Point-based	ASM	Analytical	Orginal CG-line	Proposed
SimpleShape	144(0.353)	35.0(8.51)	2.78(0.403)	221	0.799(0.290)
Seigaiha	235(0.362)	27.3(6.32)	22.7(0.254)	291	3.98(0.248)
Tokyo	80.1(0.354)	4.89(0.880)	-	111	3.86(0.289)

Optics EXPRESS

maximum computational time. As for the ASM method, the computational requirements are determined by the number of layers at different depths and the resolution of the aperture; thus, the SimpleShape model takes the maximum time to compute. Furthermore, the ASM method requires transferring the aperture data for all layers. In this study, we send the aperture data with the exact resolution to CGH as a straightforward implementation; thus, the memory access time is generally more significant than the other methods, and it depends on the number of layers. In the analytical method, the computational requirements are proportional to the number of line segments, independent of their length, and the formulas for arcs are more complicated than those of straight lines. Hence, the Seigaiha model would take a longer time than the SimpleShape model to compute. For the proposed method, the computational load is proportional to the total line length, and it is not significantly different among the straight lines and arcs, unlike the analytical method. However, the computational load of the Bezier curve is more intensive because it solves cubic equations. Thus, the Tokyo model takes the maximum time among the 3D models with the proposed method.

4.2. Image quality

To evaluate image quality, we calculated the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) between the numerically reconstructed images at the depth where the line-drawn object exists, using the reconstructed image obtained from the CGH of the Point-based method as a reference.

Tables 2, 3, and 4 compare the image quality for each 3D model by the PSNR and SSIM at 2k resolution. Fig. 10 provides the trend of the average of each value among all layer according to the CGH resolution. The image quality of the proposed method is degraded compared with the ASM method. It is the worst in the SimpleShape model and better in the Seigaiha model, compared with the analytical method at 2k resolution. The SSIM and PSNR of the SimpleShape and Seigaiha models are worse than those of the Tokyo model, which has only one layer. This may be because images reproduced at different depths affect those reproduced at other depths.

Table 2. Image quality (SimpleShape)

	ASM		Analytical		Original CG-line		Proposed	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.24	31.3	0.781	22.2	0.494	28.1	0.596	23.9	0.549
0.28	28.9	0.741	23.5	0.603	22.7	0.448	25.2	0.603
0.32	28.4	0.725	24.9	0.641	25.3	0.540	23.9	0.576
0.36	25.6	0.675	24.1	0.640	22.5	0.475	22.1	0.557
0.40	25.0	0.655	24.8	0.664	22.7	0.499	22.5	0.569
0.44	26.3	0.673	25.9	0.702	23.8	0.534	23.5	0.619
0.48	26.8	0.681	25.4	0.734	21.5	0.457	24.4	0.637
0.52	23.6	0.597	24.4	0.680	22.2	0.484	22.0	0.556
0.56	23.5	0.593	24.7	0.664	22.4	0.485	22.2	0.556
0.60	25.7	0.659	25.3	0.709	21.1	0.416	23.6	0.612
0.64	26.1	0.666	27.3	0.693	24.4	0.484	23.5	0.598
0.68	23.3	0.554	25.2	0.621	22.3	0.414	22.9	0.504
avg	26.2	0.667	24.8	0.654	23.3	0.486	23.3	0.578

Figures 11, 12, and 13 show examples of numerically and optically reconstructed images of each 3D model. Note that the reconstructed image of the SimpleShape model (Fig. 11) is zoomed around the object to enhance the visibility. The optically reconstructed images are overlayed

Optics EXPRESS

Table 3. Image quality (Seigaiha)

	ASM		Analytical		Original CG-line		Proposed	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.10	26.0	0.514	19.9	0.347	21.5	0.279	22.9	0.415
0.20	30.3	0.712	20.5	0.372	22.8	0.312	26.6	0.592
0.30	30.9	0.740	21.6	0.416	23.1	0.370	26.2	0.587
0.40	28.1	0.648	22.2	0.436	22.0	0.379	23.6	0.517
0.50	24.7	0.587	22.4	0.441	21.3	0.345	22.1	0.465
0.60	23.7	0.512	22.6	0.434	20.6	0.318	21.1	0.428
0.70	22.2	0.463	22.2	0.393	20.1	0.270	21.3	0.407
0.80	22.0	0.409	22.9	0.441	19.6	0.234	21.7	0.409
avg	26.0	0.573	21.8	0.410	21.4	0.313	23.2	0.477

Table 4. Image quality (Tokyo)

	ASM		Analytical		Original CG-line		Proposed	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.30	44.9	0.996	-	-	26.6	0.825	25.7	0.951

by unwanted light (background light of rectangle shape), which can generally be removed with a spatial filter system such as the 4f optical system. However, since the optical reproduction image quality was degraded due to distortions and aberrations caused by the lenses constituting the 4f optical system, this paper demonstrates the proposed method's effectiveness by showing a distortion-free reproduction image without introducing any spatial filter. We must mention that this noise is somehow suppressed by incident light's calibration using a polarization plate. According to the figures, the proposed method can reconstruct 3D images with approximately similarquality as the other method, which is sufficient to recognize the shape of the line-drawn objects.

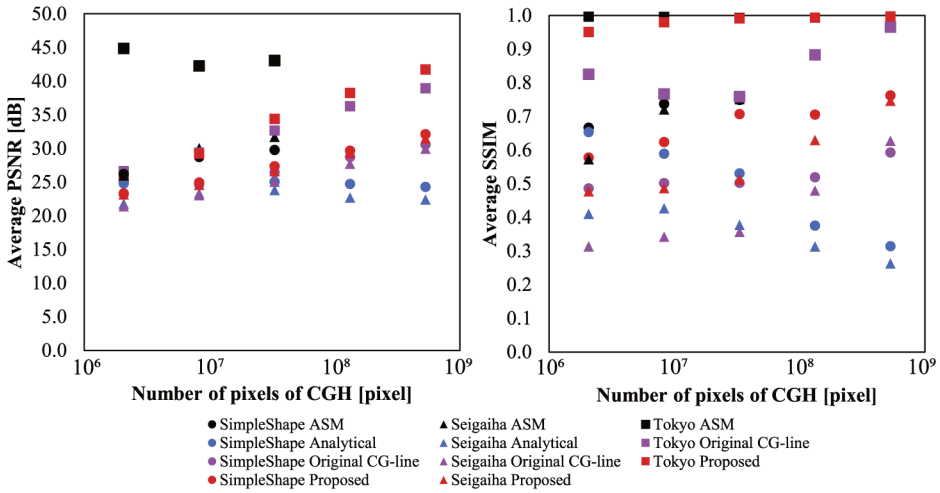

Fig. 10. Trend of image quality for CGH resolution.

Optics EXPRESS

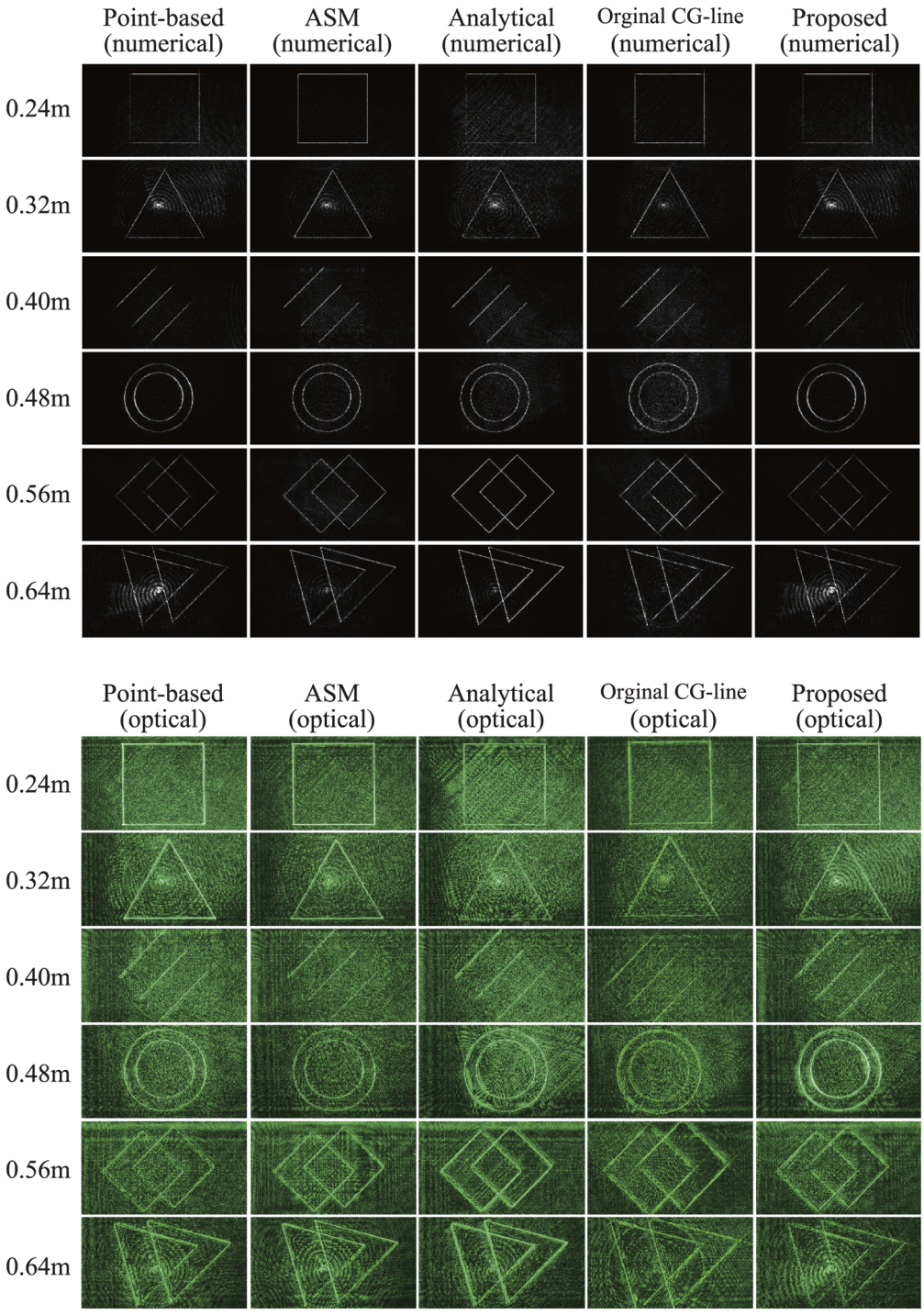

Fig. 11. Example of numerically and optically reconstructed images of the SimpleShape model.

Optics EXPRESS

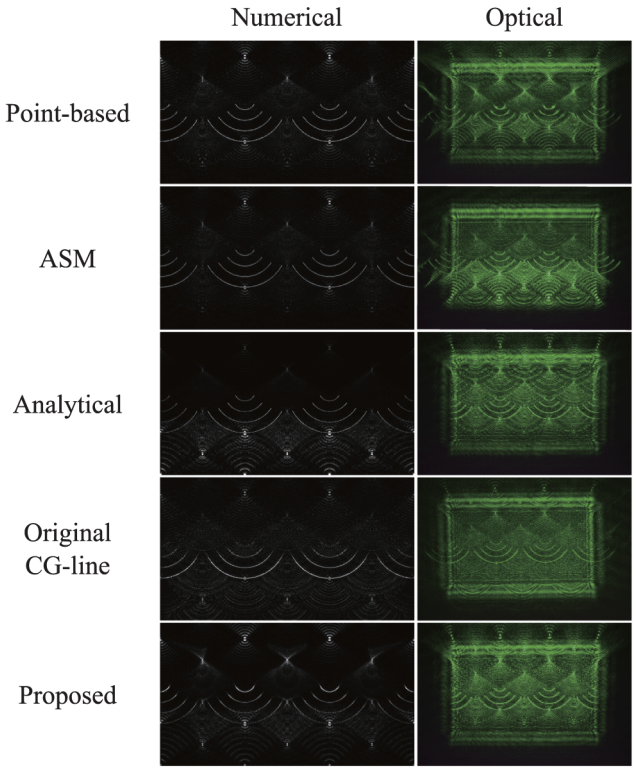

Fig. 12. Example of numerically and optically reconstructed images of the Seigaiha model.

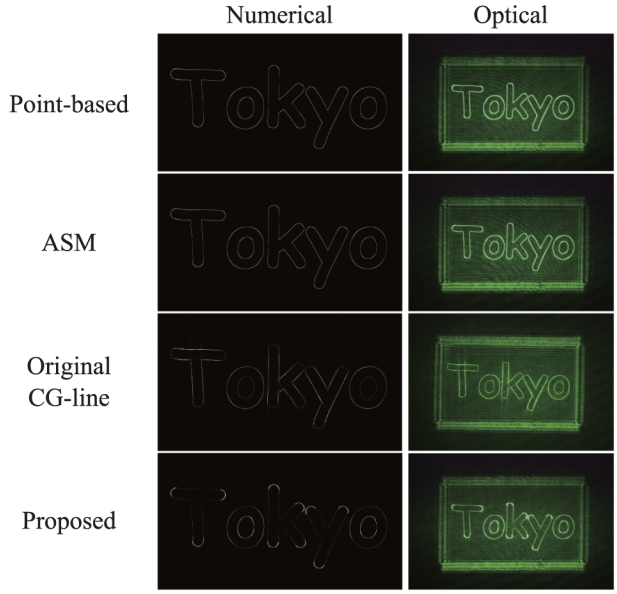

Fig. 13. Example of numerically and optically reconstructed images of the Tokyo model.

Optics EXPRESS

5. Discussion

5.1. Calculation speed

The computational speed of the proposed method is the highest among all the tested methods. The achieved framerate of the proposed method in 2k resolution was 918 fps for the SimpleShape model, 236 fps for the Seigaiha model, and 241 fps for the Tokyo model, and these values are high enough for the smooth rendering of 3D images, considering the framerate of current television systems. The proposed method's speed-up effect increases as the resolution increases, suggesting that it is an effective computational method for practical holographic displays.

5.2. Image quality

As shown in the comparison of the numerically and optically reconstructed images, the quality of the replayed images with the proposed method was not significantly degraded compared with that of the reference methods. The numerical values of the PSNR and SSIM of the proposed method were approximately equal or not significantly lower than that of the standard fair-quality image, where PSNR is 25 dB and SSIM is 0.88 or higher [32]. Therefore, considering the speed-up effect, the proposed method is superior to the reference methods. There are various causes for image degradation, including the influence of images in other layers and various noises, and since they are common to all methods, there is a need for further research.

However, a single-layer 3D projection such as the Tokyo model can produce a highly visible image, as indicated by the values of PSNR and SSIM and the optically and numerically reproduced images. This makes it practical for applications such as projecting onto the windshield of a car. Furthermore, optically reconstructed images are visually degraded primarily due to zero-order noise. This could be removed by 4f optical systems, which we will introduce in our future study.

The proposed method's image quality improves with the resolution, since the number of pixels occupied by lines in the total number of pixels of the reproduced image becomes relatively tiny at higher resolutions. Also, the Analytical method degrades the image quality because of aliasing caused by the steep diffraction angles, inducing frequencies surpassing the Shannon bound of the sampled hologram. However, this problem can be solved by making additional checks when evaluating the analytical integral expression.

Compared to the original CG-line method, the PSNR of the proposed method is either the same or better, except for the Tokyo model, and all SSIM are improved. In general, SSIM is regarded as a more accurate way to evaluate the objective quality of images. The SSIM of the proposed method is higher than that of the original CG-line method in all cases, which is evidence of the proposed method's superiority compared to the original CG-line method. This can be explained by the fact that the proposed method does not produce blank pixels that occur in the original CG-line method. Besides, the proposed method calculates the y of the convergent wavefront $U(y)$ as a continuous value. Therefore, the proposed method can superimpose wavefront with less error than the original CG-line method.

To enhance the image quality of the proposed method, one of the promising approaches is to flatten the intensity distribution between the straight and curved lines. The most striking example of this is the replayed image of the Tokyo model. Nonuniformity in the intensity distribution is observed between areas with small curvature and linear areas of the letters. One of the causes of this phenomenon is the difference in areas when obtaining the 1D wavefront between the curved and straight lines, according to the radius of curvature of the line.

Considering a simple case where the curve is an arc whose radius of curvature is constant, as shown in Fig. 14. Given that l is the length of the line and R the radius of curvature, the total area to obtain the 1D wavefront becomes $2R_\zeta l$ when $R_\zeta < R$ as shown in Fig. 14(a), and $\frac{l}{R}({R_\zeta}^2 + R^2)$ when $R_\zeta \geq R$ as shown in Fig. 14(b), and $2R_\zeta l$ when the line is straight, as shown in Fig. 14(c), where R_ζ is defined in Eq. (4). The light intensity on the reconstructed plane is correlated with

Optics EXPRESS

the area of the wavefront on the hologram plane; thus, it may cause nonuniformities between the straight and curved lines when $R_\zeta \geq R$.

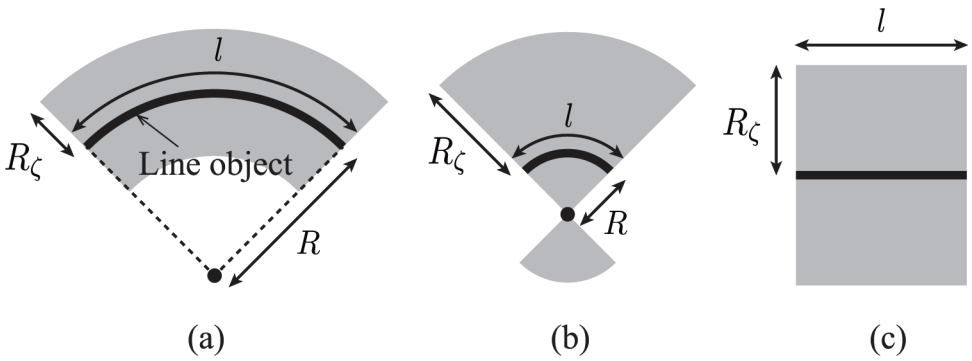

Fig. 14. Area to paste the 1D wavefront (gray area) for a line object: (a) for an arc when $R_\zeta < R$, (b)for an arc when $R_\zeta \geq R$, and (c) for a straight line.

To validate this hypothesis, we performed a numerical experiment to calculate the intensity distribution from the CGHs created by the line object, which consisted of a straight line and an arc of the same length, as shown in Fig. 15. In this experiment, we compared the sum of the intensity values for the pixels on the reconstructed image between the straight line and the arc; i.e., the intensity values were masked by the original image whose line thickness was 3 pixels and thereafter accumulated at each part. Figure 16 shows the results for the two cases: (a) lines at 0.1-m depth and (b) lines at 0.3-m depth. In Fig. 16, the circular mark indicates I_s/I_c where I_s and I_c are the sum of the measured intensity of the straight line and curved parts, and the cross mark indicates A_s/A_c where A_s and A_c are the areas of the obtained 1D wavefront for the straight line and curved parts, respectively. Here, R_ζ was 416 pixel (3.33 mm) for the 0.1-m depth and 1,249 pixel (9.99 mm) for the 0.3-m depth. The approximate shapes of the two graphs show a similar trend, which supports the hypothesis. Therefore, adjusting the intensity of the wavefront according to the radius of curvature of the line may solve this problem. However, this requires an accurate estimation of the radius of curvature of the quadratic parametric curve. Thus, this issue will be reported in our future study.

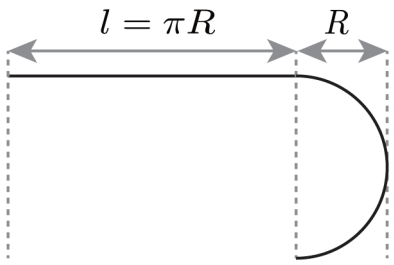

Fig. 15. Line object to validate the nonuniformity of intensity distribution between straight and curve lines.

Optics EXPRESS

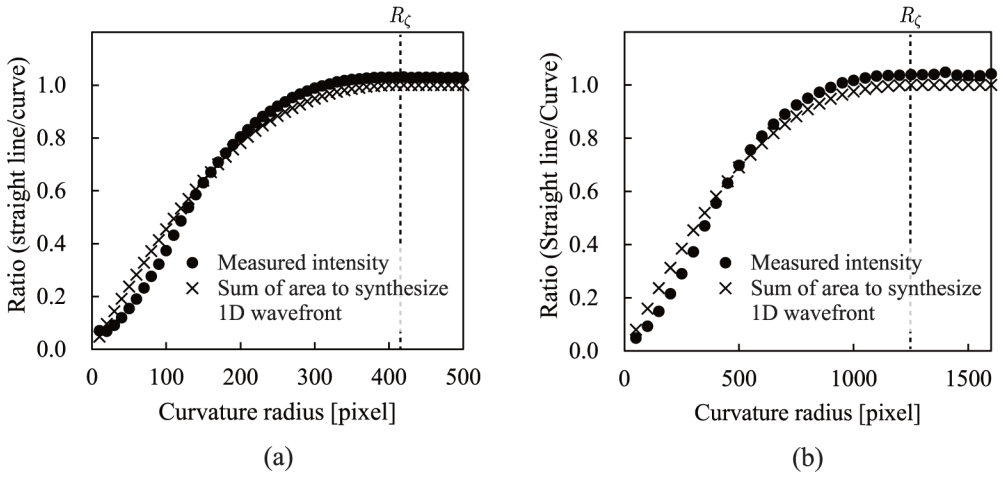

Fig. 16. Ratio of the sum of the values of intensity and area to obtain the 1D wavefront for the straight line and arc of same length: (a) lines on the 0.1-m depth and (b) lines on the 0.3-m depth.

6. Conclusion

A novel fast method for calculation of CGH for 3D line objects at layers of different depths that can be accelerated on a GPU is proposed in this paper. Compared with the reference methods, the proposed method can calculate holograms up to 3444 times faster while preserving the image quality. The proposed method can draw arbitrary curves that lie on the same depth plane and can be expressed with quadratic parametric functions such as the Bezier curve, which has practical applications in areas like the TrueType font [33]. Hence, it has many potential applications, including car windshields and head-up displays.

Funding. Grant-in-Aid for Research from the Faculty of System Design, TMU; Fonds Wetenschappelijk Onderzoek (12ZQ220N, VS07820N); Takayanagi Kenjiro Foundation; Japan Society for the Promotion of Science (19H01097, 20K19810).

Disclosures. The authors declare no conflicts of interest.

References

1. M. E. Lucente, "Interactive computation of holograms using a look-up table," J. Electron. Imaging **2**(1), 28–34 (1993).
2. S.-C. Kim and E.-S. Kim, "Effective generation of digital holograms of three-dimensional objects using a novel look-up table method," Appl. Opt. **47**(19), D55–D62 (2008).
3. S.-C. Kim and E.-S. Kim, "Fast computation of hologram patterns of a 3D object using run-length encoding and novel look-up table methods," Appl. Opt. **48**(6), 1030–1041 (2009).
4. T. Nishitsuji, T. Shimobaba, T. Kakue, N. Masuda, and T. Ito, "Fast calculation of computer-generated hologram using the circular symmetry of zone plates," Opt. Express **20**(25), 27496–27502 (2012).
5. T. Nishitsuji, T. Shimobaba, T. Kakue, and T. Ito, "Fast calculation of computer-generated hologram using run-length encoding based recurrence relation," Opt. Express **23**(8), 9852–9857 (2015).
6. D. Pi, J. Liu, Y. Han, A. U. R. Khalid, and S. Yu, "Simple and effective calculation method for computer-generated hologram based on non-uniform sampling using look-up-table," Opt. Express **27**(26), 37337–37348 (2019).
7. T. Zhao, J. Liu, Q. Gao, P. He, Y. Han, and Y. Wang, "Accelerating computation of CGH using symmetric compressed look-up-table in color holographic display," Opt. Express **26**(13), 16063–16073 (2018).
8. S. Igarashi, T. Nakamura, and M. Yamaguchi, "Fast method of calculating a photorealistic hologram based on orthographic ray-wavefront conversion," Opt. Lett. **41**(7), 1396–1399 (2016).
9. K. Wakunami and M. Yamaguchi, "Calculation for computer generated hologram using ray-sampling plane," Opt. Express **19**(10), 9086–9101 (2011).
10. Y. Zhao, M.-L. Piao, G. Li, and N. Kim, "Fast calculation method of computer-generated cylindrical hologram using wave-front recording surface," Opt. Lett. **40**(13), 3017–3020 (2015).

Optics EXPRESS

11. D. Pi, J. Liu, Y. Han, S. Yu, and N. Xiang, "Acceleration of computer-generated hologram using wavefront-recording plane and look-up table in three-dimensional holographic display," Opt. Express **28**(7), 9833–9841 (2020).
12. P. W. M. Tsang and T.-C. Poon, "Fast generation of digital holograms based on warping of the wavefront recording plane," Opt. Express **23**(6), 7667–7673 (2015).
13. T. Shimobaba, N. Masuda, and T. Ito, "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane," Opt. Lett. **34**(20), 3133–3135 (2009).
14. L. Shi, B. Li, C. Kim, P. Kellnhofer, and W. Matusik, "Towards real-time photorealistic 3D holography with deep neural networks," Nature **591**(7849), 234–239 (2021).
15. J.-S. Chen and D. P. Chu, "Improved layer-based method for rapid hologram generation and real-time interactive holographic display applications," Opt. Express **23**(14), 18143–18155 (2015).
16. Y. Zhao, L. Cao, H. Zhang, D. Kong, and G. Jin, "Accurate calculation of computer-generated holograms using angular-spectrum layer-oriented method," Opt. Express **23**(20), 25440–25449 (2015).
17. H. G. Kim and Y. Man Ro, "Ultrafast layer based computer-generated hologram calculation with sparse template holographic fringe pattern for 3-D object," Opt. Express **25**(24), 30418–30427 (2017).
18. L. Wei, F. Okuyama, and Y. Sakamoto, "Fast calculation method with saccade suppression for a computer-generated hologram based on fresnel zone plate limitation," Opt. Express **28**(9), 13368–13383 (2020).
19. D. Blinder and P. Schelkens, "Accelerated computer generated holography using sparse bases in the STFT domain," Opt. Express **26**(2), 1461–1473 (2018).
20. D. Blinder, "Direct calculation of computer-generated holograms in sparse bases," Opt. Express **27**(16), 23124–23137 (2019).
21. T. Shimobaba and T. Ito, "Fast generation of computer-generated holograms using wavelet shrinkage," Opt. Express **25**(1), 77–87 (2017).
22. T. Nishitsuji, Y. Yamamoto, T. Sugie, T. Akamatsu, R. Hirayama, H. Nakayama, T. Kakue, T. Shimobaba, and T. Ito, "Special-purpose computer HORN-8 for phase-type electro-holography," Opt. Express **26**(20), 26722–26733 (2018).
23. H. Kim, Y. Kim, H. Ji, H. Park, J. An, H. Song, Y. T. Kim, H. Lee, and K. Kim, "A Single-Chip FPGA holographic video processor," IEEE Trans. Ind. Electron. **66**(3), 2066–2073 (2019).
24. J. An, K. Won, Y. Kim, J.-Y. Hong, H. Kim, Y. Kim, H. Song, C. Choi, Y. Kim, J. Seo, A. Morozov, H. Park, S. Hong, S. Hwang, K. Kim, and H.-S. Lee, "Slim-panel holographic video display," Nat. Commun. **11**(1), 5568 (2020).
25. Y.-H. Seo, Y.-H. Lee, and D.-W. Kim, "ASIC chipset design to generate block-based complex holographic video," Appl. Opt. **56**(9), D52–D59 (2017).
26. Y. Ichihashi, R. Oi, T. Senoh, K. Yamamoto, and T. Kurita, "Real-time capture and reconstruction system with multiple GPUs for a 3D live scene by a generation from 4K IP images to 8K holograms," Opt. Express **20**(19), 21645–21655 (2012).
27. Y. Pan, X. Xu, S. Solanki, X. Liang, R. B. A. Tanjung, C. Tan, and T.-C. Chong, "Fast CGH computation using S-LUT on GPU," Opt. Express **17**(21), 18543–18555 (2009).
28. T. Nishitsuji, T. Shimobaba, T. Kakue, and T. Ito, "Fast calculation of computer-generated hologram of line-drawn objects without FFT," Opt. Express **28**(11), 15907–15924 (2020).
29. D. Blinder, T. Nishitsuji, T. Kakue, T. Shimobaba, T. Ito, and P. Schelkens, "Analytic computation of line-drawn objects in computer generated holography," Opt. Express **28**(21), 31226–31240 (2020).
30. Bhatia, *Computer Graphics* (I. K. International Pvt Ltd, 2008).
31. W. H. Beyer, *CRC handbook of mathematical sciences 5th edition* (CRC, 1978).
32. A. Moldovan, I. Ghergulescu, and C. H. Muntean, "A novel methodology for mapping objective video quality metrics to the subjective MOS scale," in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, (2014), pp. 1–7.
33. F. Yuan, *Windows Graphics Programming: Win32 GDI and DirectDraw* (Prentice Hall Professional, 2001).