Research Article Vol. 31, No. 2 / 16 Jan 2023 / Optics Express 3072
Optics EXPRESS

Three-dimensional spline-based computer-generated holography

DAVID BLINDER,[1,2,4,*] TAKASHI NISHITSUJI,[3] AND PETER SCHELKENS[1,2]

[1]Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), Pleinlaan 2, B-1050 Brussel, Belgium
[2]IMEC, Kapeldreef 75, B-3001 Leuven, Belgium
[3]Faculty of Systems Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo, Japan
[4]david.blinder@vub.be
[*]dblinder@etrovub.be

Abstract: Electro-holography is a promising 3D display technology, as it can, in principle, account for all visual cues. Computing the interference patterns to drive them is highly calculation-intensive, requiring the design and development of efficient computer-generated holography (CGH) algorithms to facilitate real-time display. In this work, we propose a new algorithm for computing the CGH for arbitrary 3D curves using splines, as opposed to previous solutions, which could only draw planar curves. The solutions are analytically expressed; we conceived an efficiently computable approximation suitable for GPU implementations. We report over 55-fold speedups over the reference point-wise algorithm, resulting in real-time 4K holographic video generation of complex 3D curved objects. The proposed algorithm is validated numerically and optically on a holographic display setup.

1. Introduction

In computer-generated holography (CGH), the goal is to compute numerical diffraction patterns for various applications in holography. Because of the wave-based nature of light, all points in space can potentially affect all the wavefield pixels in the hologram, requiring intensive calculations. One of the main challenges in CGH is to compute holograms efficiently while minimizing accuracy loss. CGH has a broad range of applications, such as beam shaping [1], optical trapping [2], or optical distortion correction [3].

This work focuses on holographic displays [4], which can be considered the ultimate 3D display technology since they can account for all visual cues, i.e., accurate focus cues, continuous parallax, realistic shading and occlusion, and no accommodation-vergence conflict [5]. Many types of CGH algorithms have been devised [6], differently trading off calculation time and supported visual effects and geometries. Among many others, there are point-cloud [7], layer-based [8], and polygon-based [9] methods; there are various acceleration techniques as well, such as sparsity-based methods [10–12] and look-up table (LUT) techniques [13]

Recently, deep-learning-based CGH has had a significant impact on holographic display technology [14–16]. It has been used for accelerating or substituting different algorithmic components in CGH, and is particularly useful for highly non-linear optimization processes with complex loss functions [17,18], achieving unprecedented quality. However, they require representative training data, and may behave unpredictably for outlier inputs given their often black-box nature. This contrasts with the analytical approach taken in this paper. Analytical solutions tend to be more predictable, precise and often computationally simpler, though they are much less flexible and can be harder to optimize and integrate in end-to-end CGH systems.

We are considering "line-based CGH" techniques, allowing for efficient calculation of the wavefront pattern created by line and curve segments, rather than sampling them as a dense series

#480095 https://doi.org/10.1364/OE.480095
Journal © 2023 Received 8 Nov 2022; revised 26 Dec 2022; accepted 1 Jan 2023; published 11 Jan 2023

Optics EXPRESS

of points. We have proposed different techniques to that end [19–23]. Unlike these methods, only planar curves were supported parallel to the hologram plane. The work in [24] partially addressed this shortcoming by proposing an algorithm for computing CGH for 3D wireframes; however, this approach only supports straight lines but not 3D curves.

The present study alleviates this limitation by proposing a new algorithm for computing the CGH of 3D splines. Our specific contributions are as follows:

- We derive an analytical model for the diffraction pattern created by short tilted line segments and validate the model with numerical simulations.
- We generalize the geometric model in [21] for determining affected pixels by computing intersections of circle segments with the projected spline.
- We propose efficient approximations for high-speed evaluation of the intersections and wavefront amplitude value in each hologram pixel.
- A highly optimized version is implemented in CUDA for GPU, and we report real-time CGH at video frame rates.
- The algorithm is validated both in numerical simulations and on a holographic display setup.

Like its predecessors, the method does not need transforms (such as the FFT, STFT or wavelets) or large amounts of precomputed data, which are essential for most of the layer-based, polygon-based, sparsity-based, LUT-based, and deep-learning-based CGH methods. This facilitates efficient implementation for GPUs or for specialized hardware implementations using FPGAs or ASICs.

The remainder of this paper is organized as follows: in section 2., we derive an analytical model for the wavefield emanating from short tilted line segment apertures; section 3. covers the extended model for computing intersections in order to determine which hologram pixels need updating; an efficient approximation and GPU implementation is discussed in section 4.; the system is evaluated numerically and optically in section 5., and we finally conclude in section 6..

2. Theory

We begin from the expression for a point-spread function (PSF) in holography, created by a coherent luminous point at coordinates $(\delta, \epsilon, \zeta \neq 0)$, which will create a complex-valued interference pattern P in the hologram plane $z = 0$,

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

where a is the point amplitude and λ is the wavelength of light. We can integrate this expression to obtain the wavefront created from a line segment parallel to the hologram plane $L_{\parallel}$, which is given by

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

where a is now the constant line amplitude, and ℓ is half of the line segment length. Since diffraction expressions are rotationally symmetric around the propagation axis z, we can choose to align the line segment with the x-axis for notational simplicity without loss of generality. In

Optics EXPRESS

the limit for an infinitely long line, Eq. (2) can be transformed to a pure y-function

$$\lim_{\ell\to\infty} L_{\parallel}(x,y) = U_{\parallel}(y) = a\sqrt{\frac{\lambda}{\zeta}}\exp\left(\frac{i\pi y^2}{\lambda\zeta}\right), \tag{3}$$

up to a constant phase term, which is omitted as it does not alter the appearance of the hologram pattern. This expression was used in [21] to draw the CGH of planar curves lying in planes parallel to the hologram plane, sweeping a section of this one-dimensional wavefield signal along the curve, drawing this pattern on lines perpendicular to the curve in every point. This approach will unfortunately not work for arbitrary 3D curves, which are not confined to constant-depth planes, which we want to address in this paper. Let us now consider a line segment that is linearly varying in depth. The integral now becomes [25]

$$L(x,y) = \sqrt{1+m^2}\int_{-\ell}^{+\ell}\frac{a}{\zeta(u)}\cdot\exp\left(\frac{\pi\mathrm{i}}{\lambda\zeta(u)}\left[(x-u)^2+y^2\right]\right)\mathrm{d}u, \tag{4}$$

where $\zeta(u) = mu + d$ is the linearly varying depth. We use the substitutions $\rho = \frac{d}{m}$, $r = \sqrt{(x+\rho)^2+y^2}$, $\beta = \frac{2\pi}{\lambda m}$ and $u = mrv - \rho$, to get

$$\frac{a}{m}\sqrt{1+m^2}\cdot\exp\left(-\mathrm{i}\beta(x+\rho)\right)\cdot\int_{-mr\ell-\rho}^{+mr\ell-\rho}\frac{1}{v}\exp\left(\frac{\mathrm{i}\beta r}{2}\left[v+\frac{1}{v}\right]\right)\mathrm{d}v. \tag{5}$$

This integral has a radial symmetry in r, around the midpoint $(-\rho, 0)$. We can use another substitution, $e^{\mu} = v$, to solve for the integral part with general bounds

$$\int_0^w\frac{1}{e^{\mu}}\exp\left(\mathrm{i}\beta r\cdot\frac{e^{\mu}+e^{-\mu}}{2}\right)\mathrm{d}e^{\mu} = \int_0^w\exp\left(\mathrm{i}\beta r\cosh\mu\right)\mathrm{d}\mu = I_0(w,\beta r), \tag{6}$$

where $I_\nu(w,\cdot)$ is the incomplete modified Bessel function of the first kind and order ν [26,27]. To our knowledge, this is the first time a precise analytical expression is given for computing the diffraction pattern for tilted line segment apertures.

We can also take the limit of this expression for lines tending to infinite length,

$$\lim_{\ell\to\infty} L(x,y) = a\sqrt{1+m^2}\cdot\exp(-\mathrm{i}\beta(x+\rho))\cdot\frac{-2\pi\mathrm{i}}{m}J_0(\beta r), \tag{7}$$

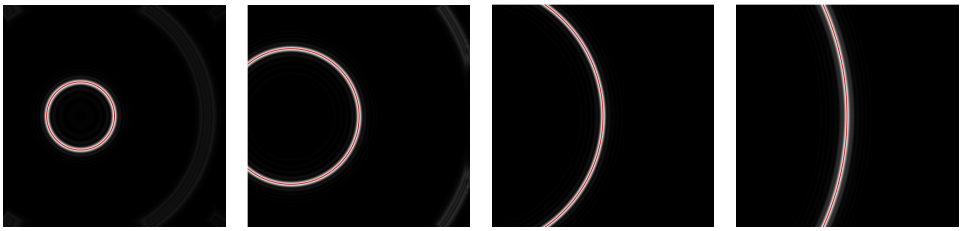
where $J_\nu(s)$ is the Bessel function of the first kind of order ν. Note that instead of having a symmetry along the y-axis as for the parallel line case, we now have a different radial symmetry along r. This effect can also be observed experimentally, where short tilted line segments create radially symmetric patterns with radius ρ, see Fig. 1.

Assuming that the signal is thus the most significant at radius $r = \rho$, we can express the latter along the circle with radius ρ, centered in $(-\rho, 0)$ along the angular parameter φ:

$$\frac{a}{m}\sqrt{1+m^2}\cdot J_0(\beta\rho)\cdot\exp\left(\mathrm{i}\beta\rho\cos\varphi\right), \tag{8}$$

since $x = \rho(\cos\varphi - 1)$. This can be viewed as a generalization of parallel line segments with zero tilt, tending in the limit to an infinite radius, resulting in a straight line. We will utilize this principle to draw *circular-segment-shaped wavefronts* along the curve instead of line-shaped ones to obtain holograms of general 3D curves, which we elaborate on in the next section.

Optics EXPRESS



(a) $\rho = 2.5$mm (b) $\rho = 5.0$mm (c) $\rho = 10.0$mm (d) $\rho = 20.0$mm

Fig. 1. Intensity of generated holograms of short tilted line segments shown in grayscale. The superimposed red circle segment had radius ρ and origin $(-\rho, 0)$, demonstrating the accuracy of the derived model. Simulation parameters: $\lambda = 532$ nm, pixel pitch $p = 8$ µm, resolution of 2048×2048 pixels with origin at the center.

3. Methodology

We consider 3D quadratic spline $g(t)$ bounded by $t \in [-1, +1]$, whose spatial coordinates are described by three quadratic polynomials

$$g(t) \begin{cases} g_x(t) = a_x t^2 + b_x t + c_x \\ g_y(t) = a_y t^2 + b_y t + c_y \\ g_z(t) = a_z t^2 + b_z t + c_z \end{cases}, \tag{9}$$

where $\{g_x(t), g_y(t), g_z(t)\}$ denote the Cartesian components of the spline along every dimension.

In previous work, we only considered planar splines placed at a constant distance $g_z(t) = c_z$ from the hologram plane, which amounts to the special case where $a_z = b_z = 0$. For every sample on the hologram plane, one needs to find out which points on the spline would contribute to what part of the holographic signal. Starting with the planar curve case $g_{xy}(t)$ (ignoring the z-component $g_z(t)$ for now), we have to find out which of the lines orthogonal to the curve intersect with the target sample with coordinates (u, v). This amounts to solving the equation

$$h(t) \bullet g'_{xy}(t) = |h(t)| \cdot \left|g'_{xy}(t)\right| \cdot \cos\theta, \tag{10}$$

where "$\bullet$" is the dot product, $|\cdot|$ is the Euclidean norm, the prime symbol $'$ denotes the derivative in t, and $h(t)$ is the relative distance function parameterized by t, given by

$$h(t) \begin{cases} h_x(t) = g_x(t) - u \\ h_y(t) = g_y(t) - v. \end{cases} \tag{11}$$

For planar splines, the lines intersect the curve at orthogonal angles $\theta = \frac{\pi}{2}$ as shown on Fig. 2(a), so that $\cos\theta = 0$, making the right-hand side of Eq. (10) zero. The left-hand side is the sum of two products of a quadratic polynomial with a linear polynomial,

$$h(t) \bullet g'_{xy}(t) = h_x(t) g'_x(t) + h_y(t) g'_y(t), \tag{12}$$

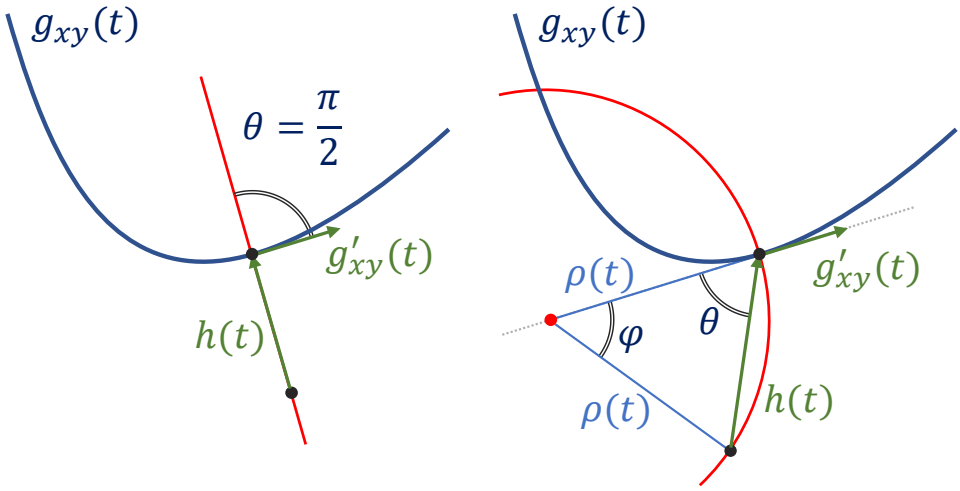
resulting in a cubic polynomial which can be solved analytically.

We extend this now for the general case where $\cos\theta \neq 0$, whenever we consider intersections with circles rather than straight lines. In Fig. 2(b), we can see that the angle θ is part of an isosceles triangle, with base length $|h(t)|$ and leg lengths $\rho(t)$. Using the law of cosines, we get

$$\cos\theta = \frac{\rho(t)^2 + |h(t)|^2 - \rho(t)^2}{2\rho(t)\,|h(t)|} = \frac{|h(t)|}{2\rho(t)}. \tag{13}$$

From the previous section, we know that the circle arc describing the signal is given by $\rho = \frac{d}{m}$ for an infinitesimal line segment with depth d and slope m along the z-dimension. This value is

Optics EXPRESS



(a) reference model (b) extended model

Fig. 2. Geometric problem to solve for the (a) special case of the planar spline, and (b) the general case of a 3D spline.

thus now parameterized along the curve by the function

$$\rho(t) = \frac{g_z(t)}{g'_z(t)}, \tag{14}$$

which will tend to infinity when $g'_z(t) \to 0$, matching the previous case of a straight line with infinite radius of curvature. Substituting these findings in Eq. (10), we get

$$h(t) \bullet g'_{xy}(t) = \frac{1}{2} |h(t)|^2 \left|g'_{xy}(t)\right| \frac{g'_z(t)}{g_z(t)}, \tag{15}$$

which we should solve for t to obtain all intersections and computing the CGH.

4. Implementation

Solving Eq. (15) analytically is not feasible in general, since it consists of a product of high-order polynomials and square-root terms of t because of $\left|g'_{xy}(t)\right|$. Since we are looking for solutions within a bounded interval $t \in [-1, +1]$, this can be numerically solved with root finding algorithms. However, these are not suitable for use on GPU; we would have to solve this complex expression using root-finding once in every pixel, leading to high computational load and thread divergence. Thus, this expression needs to be simplified further, ideally to become similar in complexity to the previous special planar spline curve case using a cubic polynomial solver.

Thankfully, parts of the expression Eq. (15) are common to all pixels, i.e., those that do not depend on u or v. As this should happen only once, these shared parts can be processed on the CPU beforehand, rendering the calculation cost comparatively negligible.

We can rewrite expression Eq. (15) to the form

$$f_0(t) + f_1(t)u + f_2(t)v + f_3(t)(u^2 + v^2), \tag{16}$$

where the $f_j(t)$ are some expressions purely in t. We now want to obtain the least-square cubic polynomial approximation $\mathsf{p}_j(t)$ for each $f_j(t)$, namely

$$\arg\min_{\mathsf{p}_j} \int_{-1}^{+1} \left(f_j(t) - \mathsf{p}_j(t)\right)^2 \mathrm{d}t \quad \text{where} \quad \mathsf{p}_j(t) = \sum_{k=0}^{3} \mathsf{p}_{j,k} \cdot t^k \tag{17}$$

Optics EXPRESS

and organize the coefficients into a matrix multiplication expression

$$\mathbf{M}\cdot\mathbf{u} = \begin{pmatrix} \mathsf{p}_{0,0} & \mathsf{p}_{1,0} & \mathsf{p}_{2,0} & \mathsf{p}_{3,0} \\ \mathsf{p}_{0,1} & \mathsf{p}_{1,1} & \mathsf{p}_{2,1} & \mathsf{p}_{3,1} \\ \mathsf{p}_{0,2} & \mathsf{p}_{1,2} & \mathsf{p}_{2,2} & \mathsf{p}_{3,2} \\ \mathsf{p}_{0,3} & \mathsf{p}_{1,3} & \mathsf{p}_{2,3} & \mathsf{p}_{3,3} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ u \\ v \\ u^2+v^2 \end{pmatrix}, \tag{18}$$

where the consecutive rows of $\mathbf{M}$ correspond to increasing powers of t, and the consecutive columns correspond to the different instances j of approximations to the $f_j(t)$. The coefficients of $\mathbf{M}$ only have to be computed once per curve segment. The values for $\mathbf{u}$ are pixel-dependent, so that this small matrix product should be computed in every pixel (u, v). The resulting 4 values correspond to the coefficients of a cubic polynomial that should be solved in each pixel.

To find $\mathbf{M}$ numerically, we can uniformly sample the interval $t \in [-1, +1]$ with Q samples $\{t_1, t_2, \ldots, t_Q\}$, and find a least squares solution to the matrix system $\mathbf{F} = \mathbf{T}\cdot\mathbf{M}$, defined as

$$\begin{pmatrix} f_0(t_1) & f_1(t_1) & f_2(t_1) & f_3(t_1) \\ f_0(t_2) & f_1(t_2) & f_2(t_2) & f_3(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_0(t_Q) & f_1(t_Q) & f_2(t_Q) & f_3(t_Q) \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_Q & t_Q^2 & t_Q^3 \end{pmatrix} \cdot \mathbf{M} \tag{19}$$

for which we can precompute the Moore–Penrose inverse $\mathbf{T}^+ = (\mathbf{T}^\mathsf{T}\mathbf{T})^{-1}\mathbf{T}^\mathsf{T}$, which will be the same for all curve segments, where $^\mathsf{T}$ is the matrix transpose. We can then calculate the least-squares solution $\mathbf{M} = \mathbf{T}^+\mathbf{F}$. The precise chosen value of Q appears to have a limited effect on the final quality of the hologram. Moreover, it's impact on computation time is negligible. In this work, we chose $Q = 32$.

We also need to consider whether a pixel is too far from a curve point, potentially giving rise to aliasing due to the signal frequency exceeding the Nyquist-Shannon bound, determined by the hologram pixel pitch p that is inversely proportional to the sampling rate. For the Fresnel approximation, the PSF shape should be a square [24]. We use a bounding box around the projected curve on the hologram plane to skip evaluating pixels that are too far away, thereby speeding up calculations. In addition, for valid solutions, we should verify whether

$$\max\left(|h_x(t)|, |h_y(t)|\right) < \frac{\lambda g_z(t)}{2p} \tag{20}$$

holds. To summarize, in order to determine the wavefront pattern created by a 3D spline, we need to solve a cubic equation in t for every pixel within the bounding box, whose polynomial coefficients are given by the matrix product $\mathbf{M}\cdot\mathbf{u}$. This will give rise to either 1 or 3 real solutions. Only real solutions satisfying both $t \in [-1, +1]$ and Eq. (20) will be used.

Whenever a sample pixel satisfies all of the above, we need to compute its complex-valued amplitude contribution with Eq. (8). We would like to re-use the already computed value of $h(t)$ instead of finding φ. This can be accomplished by utilizing the fact that $|h(t)|$ is the chord length of a circular arc with subtended angle φ, cf. Fig. 2(a). We can thus utilize the inverse of the chord function to get the relationship

$$\varphi = 2\arcsin\tfrac{|h(t)|}{2\rho(t)}. \tag{21}$$

By using the identity $\cos(2\arcsin z) = 1 - 2z^2$ in the last factor of Eq. (8), we now obtain the expression

$$a\exp\left(\frac{\pi\mathrm{i}}{2\lambda g_z(t)}\,|h(t)|^2\right) \tag{22}$$

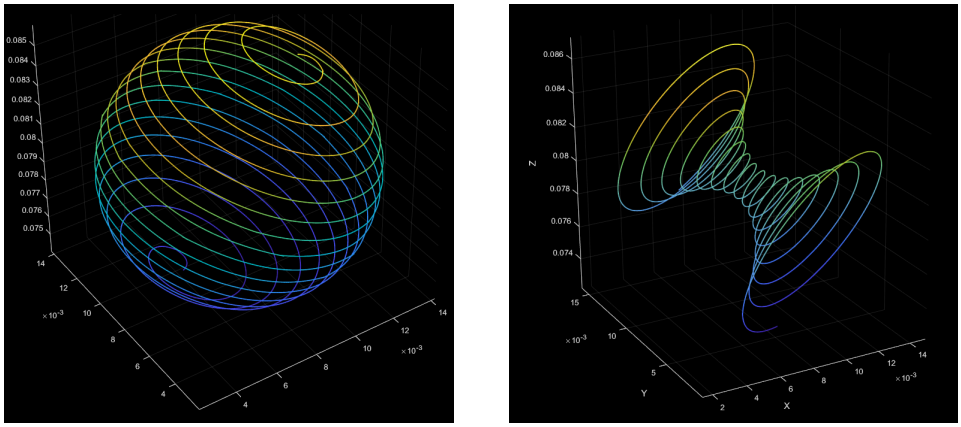
Optics EXPRESS

simplified by omitting the preceding factors which are constant in φ. This closely matches the expression from Eq. (3).

5. Experiments

We report two sets of experiments in this section, numerical experiments and optical experiments. The numerical experiments mainly evaluate computation speed, objective visual quality and numerically reconstructed views from the generated holograms. The optical experiments test and validate the algorithm speed and quality on a real-time holographic video display system.

The algorithms were implemented in CUDA enabling massively parallel GPU computation. They were run on a machine with an AMD Ryzen Threadripper 3960X processor, 64 GB of RAM and a NVIDIA Geforce RTX 3080 GPU running a Windows 11 OS. The code was implemented in C++17 with CUDA 11.6, enabling CUDA compute capability 8.6 and utilizing 32-bit floating-point precision. The experiments used the 3D models shown in Fig. 3.



(a) "Sphere" 3D model (b) "Spring" 3D model

Fig. 3. Representations of the two 3D models used in this paper. The models are color-coded according to their z-coordinates for viewing clarity.

5.1. Numerical experiments

For the numerical experiments, we use a hologram with a resolution of 4096×4096 pixels, a pixel pitch of $p = 4$ µm, and a wavelength of $\lambda = 532$ nm. This was compared with the reference algorithm, where the splines were sampled using a minimal point sampling density of p to ensure that the lines appear continuous [20]. This was computed via Eq. (1):

$$H(x, y) = \sum_j a_j \cdot \exp\left(\frac{\pi \mathrm{i}}{\lambda \zeta_j}\left[(x - \delta_j)^2 + (y - \epsilon_j)^2\right]\right) \tag{23}$$

computing the hologram pattern H by summing over multiple PSFs, where $(\delta_j, \epsilon_j, \zeta_j)$ are the 3D point cloud coordinates and a_j their corresponding amplitudes, for a sampled point cloud consisting of N_P points, so that $j = \{0, 1, \ldots, N_P - 1\}$. Pixels were only updated if they would not cause aliasing, i.e., the PSF affects pixels in a bounded square region as explained in [24]. Besides improving accuracy, this will also speed up the reference algorithm calculation times, resulting in a fairer comparison.

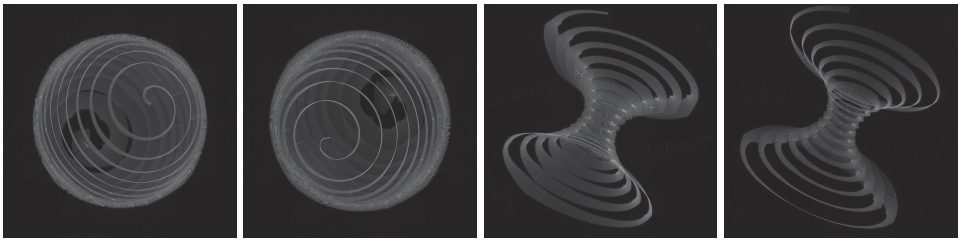
For both algorithms, a single GPU thread was assigned for every hologram pixel. In the reference method, each thread loops over every point and updates the hologram pixel according to

Optics EXPRESS

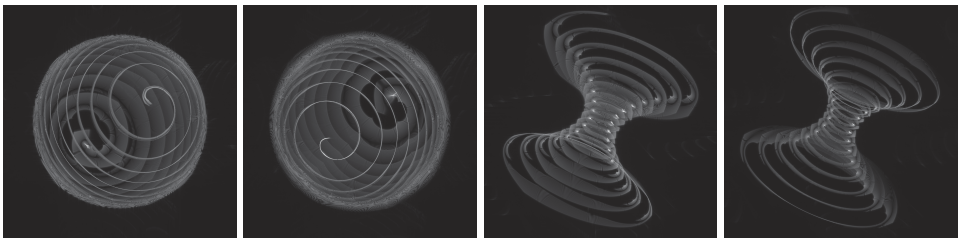
Eq. (23). In the proposed method, each thread loops all spline objects. For every spline, a thread performs a small matrix multiplication to obtain the cubic polynomial coefficients, cf. Eq. (18). This is followed by solving the cubic and using the solutions to update the pixels, analogous to the approach in [21]. In both algorithms, pixels too far away from the luminous object are skipped. Since these pixels tend to be in large contiguous segments, they will also mostly appear together in CUDA thread blocks, minimizing thread divergence and thus benefiting performance

Both algorithms were run 10 times to average the calculation times. For the "sphere" model, the reference algorithm took 1647.5 ms, while the proposed algorithm only took 29.5 ms. For the "spring" model, the reference and proposed algorithms took 2724.7 ms and 35.5 ms, respectively. We thus report speed improvements surpassing a factor of 55 over the reference implementation.

The visual quality is assessed by calculating the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) between reconstructed views of the holograms taken from the reference and proposed algorithms, respectively. These were obtained by numerically backpropagating the holograms using the angular spectrum method with zero-padding, taking the absolute value, and creating 1024×1024 8-bit grayscale images, as shown in Fig. 4. Both virtual object centers were placed 8 cm from the hologram plane, where the reported reconstruction depths are taken relative to that center. The "sphere" was reconstructed at relative depths of -4 and $+5$ mm, giving PSNRs of 30.85 and 29.93 dB, and SSIMs of 0.906 and 0.902; the "spring" was reconstructed at depths of -5 and $+3$ mm, resulting in PSNRs of 29.74 and 31.95 dB, and SSIMs of 0.896 and 0.912, for each respectively. This demonstrates that the proposed algorithm can calculate 3D spline object CGH at high speeds and acceptable visual quality.



(a) sphere ref. (−4mm) (b) sphere ref. (+5mm) (c) spring ref. (−5mm) (d) spring ref. (+3mm)



(e) sphere prop. (−4mm) PSNR = 30.85dB SSIM = 0.906 (f) sphere prop. (+5mm) PSNR = 29.93dB SSIM = 0.902 (g) spring prop. (−5mm) PSNR = 29.74dB SSIM = 0.896 (h) spring prop. (+3mm) PSNR = 31.95dB SSIM = 0.912

Fig. 4. Images of the numerically reconstructed of the "sphere" and "spring" holograms. The reference point-wise algorithm and the proposed 3D spline algorithm are shown, denoted "ref." and "prop." in the figure subcaptions. The images are calculated by taking the absolute value after backpropagating the hologram to a depth offset from the object center, denoted in brackets expressed in mm.

Although the used test objects each consist of a single smooth curve, this is not a requirement. Discontinuities or singular points (such as cusps) are supported as well. Since the computed spline diffraction patterns are independent, one can calculate the CGH of any number of splines, by linearly adding their CGH patterns.

Optics EXPRESS

In principle one can place objects at any display distance, but because of the bounded PSFs, one should take the same considerations into account as the "wavefront recording plane" (WRP) method [10]. Splines should ideally not be very close (less than 1mm or so) to the hologram plane. Otherwise the PSF shape will only be a few pixels wide causing strong approximations due to the coarse sampling. It may be desirable to put objects at some minimal distance from the hologram plane. Large distances will not affect the accuracy, but will impact calculation time as most or even all pixels will be affected. The latter can be addressed by combining the proposed method with a WRP to further speed up the method for far or deep objects.

5.2. Optical experiments

For the optical experiments, we rendered holograms on a holographic display system shown on Fig. 5. It consists of a phase-modulation type Spatial Light Modulator (SLM) (Jasper, 'JD7714') with a resolution of 4096×2400 pixels and a pixel pitch of $p = 3.74$ µm, a green laser with a wavelength of $\lambda = 532$ nm (Thorlabs, 'CPS532'), a beam expander (Thorlabs, 'GBE10-A'), a polarizer (Thorlabs, 'WP25M-VIS'), a polarized beam splitter (Thorlabs, 'CCM1-PBS251/M'), a half-wave plate (Thorlabs, 'WPH10M-532') and a quarter-wave plate (Thorlabs, 'Thorlabs WPQ10M-532'), a plano-convex lens (Thorlabs, 'LA1433-A-ML') and a hand-crafted ϕ=0.5 mm circular block filter created with 3M's aluminum coated tape. It was recorded on a Sony ILCE-6000 camera with an eyepiece (MEADE, '602416').

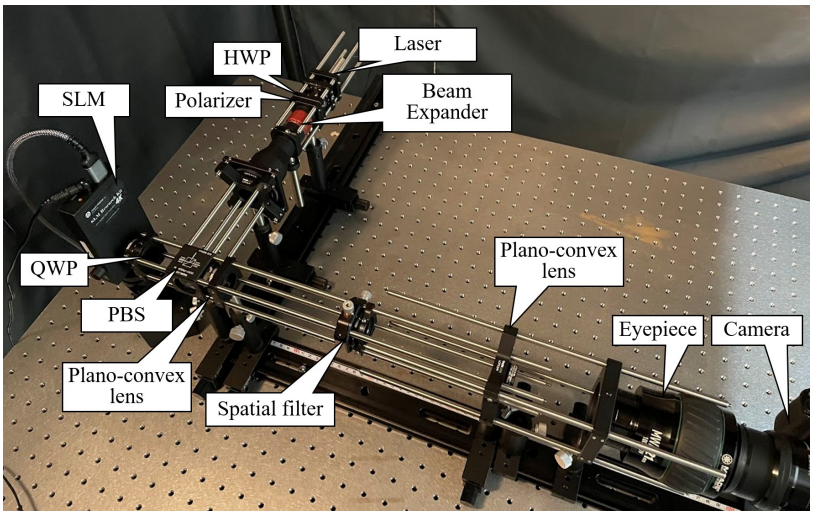


Fig. 5. Annotated photograph of the optical setup on which the generated holographic videos are shown. HWP: Half-Wave plate; QWP: Quarter-Wave plate; PBS: Polarized Beam Splitter; SLM: Spatial Light Modulator.

As displayed content, we rendered a video of the spinning "sphere" model at 30 frames per second. The model was centered w.r.t. the hologram origin and placed at a depth of 8cm. The CGH parameters (resolution, p, λ) were configured to match those of the SLM and the laser. The average generation time per frame took 21.5 ms, which suffices to achieve real-time rendering at 30 fps. On the other hand, the reference point-wise algorithm took an average of 1217.5 ms/frame, making it too slow for that purpose.

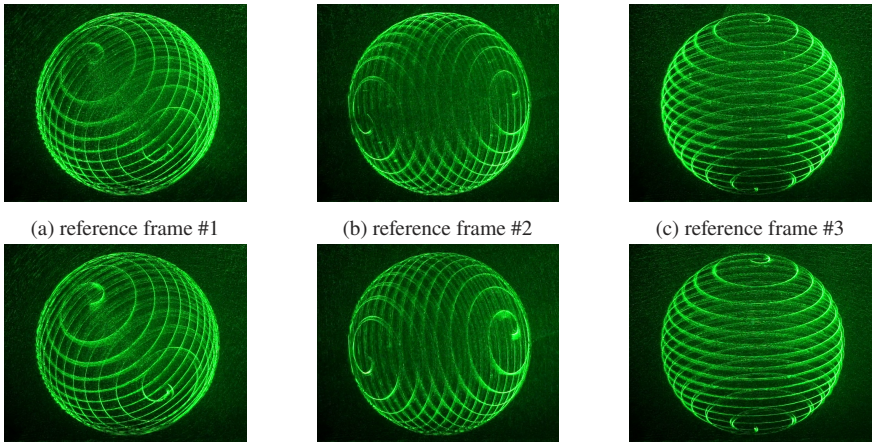
In Fig. 6, we can observe a side-by-side comparison of exemplary frames generated respectively with the reference and the proposed CGH algorithms, showing little noticeable visual quality differences. These frames were obtained directly from the camera sensor capturing the real image after focusing around the end of the eyepiece. Because the SLM is phase-only, some

Optics EXPRESS

information has to be discarded from the complex-valued CGH before displaying it on the SLM. The hologram amplitude was discarded to promote speed, and the phase values were quantized to 8-bit precision, matching the SLM precision. Namely, every complex-valued pixel value c was quantized to an 8-bit phase value representation φ by

$$\varphi = \left\lfloor \frac{128 \angle c}{\pi} \right\rfloor, \tag{24}$$

where $\lfloor \cdot \rfloor$ is the floor operator, and $\angle$ is the complex argument, returning a phase value from the interval $[0, 2\pi]$. Better visual quality could be achieved by using double-phase amplitude encoding, or phase retrieval algorithms that optimize the phase patterns at the cost of calculation speed [28,29].



(a) reference frame #1 (b) reference frame #2 (c) reference frame #3

(d) proposed frame #1 (e) proposed frame #2 (f) proposed frame #3

Fig. 6. Screenshots taken from the recorded videos of the optical setup, for three different frames using the reference CGH and the proposed CGH, respectively. The full videos can be seen in Visualization 1 and Visualization 2.

6. Conclusion

We proposed an algorithm for efficiently computing the CGH of 3D curves. Analytical expressions for tilted line segments were derived, leading to a model for how 3D curves create circularly-shaped wavefront patterns. These solutions were converted to efficiently computable approximations suitable for massively parallel processing architectures. An optimized implementation on GPU was presented, demonstrating video frame rate calculation of 4K-sized holograms with decent visual quality. The algorithm was validated using both numerical simulations and optical experiments on a digital holographic display setup. We report a 55-fold speedup over the reference point-based CGH implementation, validating the practical utility of the method. This work may contribute to accelerating CGH algorithms not only limited to display purposes and can potentially lead to new insights in numerical diffraction theory for holography and beyond.

Funding. Japan Society for the Promotion of Science (22H03616, International research fellow P22752); Fonds Wetenschappelijk Onderzoek (12ZQ220N, 12ZQ223N, VS07820N).

Disclosures. The authors declare no conflicts of interest.

Data availability. Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

Optics EXPRESS

References

1. Y. Guo, Y. Wang, Q. Hu, X. Lv, and S. Zeng, "High-resolution femtosecond laser beam shaping via digital holography," Opt. Lett. **44**(4), 987–990 (2019).
2. J. Liesener, M. Reicherter, T. Haist, and H. J. Tiziani, "Multi-functional optical tweezers using computer-generated holograms," Opt. Commun. **185**(1-3), 77–82 (2000).
3. X. Zeng, X. Zhang, D. Xue, Z. Zhang, and J. Jiao, "Mapping distortion correction in freeform mirror testing by computer-generated hologram," Appl. Opt. **57**(34), F56–F61 (2018).
4. J.-H. Park and B. Lee, "Holographic techniques for augmented reality and virtual reality near-eye displays," Light: Adv. Manuf. **3**(1), 1–14 (2022).
5. D. Blinder, A. Ahar, S. Bettens, T. Birnbaum, A. Symeonidou, H. Ottevaere, C. Schretter, and P. Schelkens, "Signal processing challenges for digital holographic video display systems," Signal Process. Image Commun. **70**, 114–130 (2019).
6. D. Blinder, T. Birnbaum, T. Ito, and T. Shimobaba, "The state-of-the-art in computer generated holography for 3D display," Light: Adv. Manuf. **3**(3), 1–29 (2022).
7. P. Tsang, T.-C. Poon, and Y. Wu, "Review of fast methods for point-based computer-generated holography [invited]," Photonics Res. **6**(9), 837–846 (2018).
8. N. Okada, T. Shimobaba, Y. Ichihashi, R. Oi, K. Yamamoto, M. Oikawa, T. Kakue, N. Masuda, and T. Ito, "Band-limited double-step fresnel diffraction and its application to computer-generated holograms," Opt. Express **21**(7), 9192–9197 (2013).
9. K. Matsushima, *Introduction to Computer Holography: Creating Computer-Generated Holograms as the Ultimate 3D Image* (Springer Nature, 2020).
10. 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).
11. M. Yamaguchi, H. Hoshino, T. Honda, and N. Ohyama, "Phase-added stereogram: calculation of hologram using computer graphics technique," Proc. SPIE **1914**, 25–31 (1993).
12. H. G. Kim and Y. M. 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).
13. 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).
14. M. H. Eybposh, N. W. Caira, M. Atisa, P. Chakravarthula, and N. C. Pégard, "DeepCGH: 3D computer-generated holography using deep learning," Opt. Express **28**(18), 26636–26650 (2020).
15. 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).
16. S. Choi, M. Gopakumar, Y. Peng, J. Kim, and G. Wetzstein, "Neural 3D holography: Learning accurate wave propagation models for 3D holographic virtual and augmented reality displays," ACM Trans. Graph. **40**(6), 1–12 (2021).
17. B. Lee, D. Kim, S. Lee, C. Chen, and B. Lee, "High-contrast, speckle-free, true 3d holography via binary cgh optimization," Sci. Rep. **12**(1), 1–12 (2022).
18. P. Chakravarthula, E. Tseng, H. Fuchs, and F. Heide, "Hogel-free holography," ACM Trans. Graph. **41**(5), 1–16 (2022).
19. 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).
20. 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).
21. T. Nishitsuji, D. Blinder, T. Kakue, T. Shimobaba, P. Schelkens, and T. Ito, "GPU-accelerated calculation of computer-generated holograms for line-drawn objects," Opt. Express **29**(9), 12849–12866 (2021).
22. T. Nishitsuji, T. Kakue, D. Blinder, T. Shimobaba, and T. Ito, "An interactive holographic projection system that uses a hand-drawn interface with a consumer CPU," Sci. Rep. **11**(1), 147 (2021).
23. T. Nishitsuji, N. Shiina, D. Blinder, T. Shimobaba, T. Kakue, P. Schelkens, T. Ito, and T. Asaka, "Variable-intensity line 3D images drawn using kinoform-type electroholography superimposed with phase error," Opt. Express **30**(15), 27884–27902 (2022).
24. D. Blinder, T. Birnbaum, and P. Schelkens, "Pincushion point-spread function for computer-generated holography," Opt. Lett. **47**(8), 2077–2080 (2022).
25. D. Blinder, T. Nishitsuji, and P. Schelkens, "Real-time computation of 3D wireframes in computer-generated holography," IEEE Trans. on Image Process. **30**, 9418–9428 (2021).
26. M. M. Agrest, M. S. Maksimov, H. E. Fettis, J. Goresh, and D. Lee, *Theory of Incomplete Cylindrical Functions and Their Applications*, vol. 160 (Springer, 1971).
27. D. Jones, "Incomplete bessel functions. I," Proc. Edinb. Math. Soc. **50**(1), 173–183 (2007).
28. T. Latychevskaia, "Iterative phase retrieval for digital holography: tutorial," J. Opt. Soc. Am. A **36**(12), D31–D40 (2019).
29. P. Chakravarthula, E. Tseng, T. Srivastava, H. Fuchs, and F. Heide, "Learned hardware-in-the-loop phase retrieval for holographic near-eye displays," ACM Trans. Graph. **39**(6), 1–18 (2020).