Research Article | Vol. 33, No. 16 / 11 Aug 2025 / Optics Express 34303
Optics EXPRESS

Sparse point cloud computer-generated holography with the Gabor transform

Bilal Moussa Fares,[1,*] Hyun-su Kim,[1,2] Peter Schelkens,[1,2] and David Blinder[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
[*]Bilal.Moussa.Fares@vub.be

Abstract: High computational requirements often limit computer-generated holography (CGH). Phase-added stereogram methods reduce this cost by approximating point-spread functions by plane wave segments. While this sparse representation allows for a speedup, this approximation introduces blocking artifacts and reduces accuracy. We present a novel approach that integrates the Gabor transform to make sparse diffraction calculations. Additionally, we partition the point cloud into Lozenge-shaped cells, optimizing memory management on GPU architectures by using cache-friendly operations. Experimental results on large point clouds demonstrate a significant improvement in speed and accuracy, achieving up to a 47× speedup over optimized brute-force methods with enhanced quality over classic phase-added stereograms.

1. Introduction

Dennis Gabor invented holography in the late 1940s [1]. This invention was a byproduct of the attempt to improve the resolution of electron microscopes. With the rise of digital computers, researchers such as Lohmann were soon able to mathematically compute holograms by simulating light diffraction [2]. These computer-generated hologram (CGH) techniques can be used for many applications, such as 3D displays [3–6], beam shaping [7], optical testing [8], interferometry [9], particle trapping [10], and surface measurements [11].

One of the difficulties of generating CGHs is their computational cost: every object element can potentially affect every hologram pixel. This makes brute-force approaches – where the impact of every element of the object is calculated for every pixel – impractical. We can classify CGH algorithms into the following groups: point-cloud [12], polygon [13], and ray-based methods [14]. The polygon method represents virtual objects by 2D polygons, typically triangles. The interference pattern on the hologram plane is then the summation of the waves of all polygons onto the hologram plane. Ray-based methods split the hologram plane into elementary holograms, where rays are launched from the center of these elements, whose intersection with the virtual hologram object is calculated. Every intersection is treated as a point source for which we can calculate the contribution to the hologram plane.

In this paper, we will focus on point-cloud methods. Here, each point is treated as an infinitesimal light source with spherical wave propagation. By evaluating the point-spread function (PSF) for all points, every contribution can be calculated for each pixel on the hologram. However, this is a costly computation to perform. Every point in the point cloud will affect every pixel of the hologram, yielding a time complexity of $O(QNM)$ [15], where Q is the number of points and $M \times N$ the number of pixels the x and y dimensions.

To accelerate the computations, phase-added stereograms (PAS) were introduced [16]. Instead of calculating individual PSF contributions for every pixel, the PAS method splits the hologram into blocks, where the PSF is approximated in each block by a plane wave segment. Each segment can be approximated by a single coefficient in the Fourier domain, following the Accurate

#565177 https://doi.org/10.1364/OE.565177
Journal © 2025 Received 8 May 2025; revised 21 Jun 2025; accepted 7 Jul 2025; published 5 Aug 2025

Optics EXPRESS

PAS (APAS) approach, which quantizes continuous plane wave components to the nearest DFT frequency. This block-wise Fourier representation is an instantiation of the short-time Fourier transform (STFT). Since only a single coefficient must be updated per block, this sparse representation enables significant acceleration of CGH algorithms.

Although sparsity levels below 1% are achievable, optimized GPU implementations of the basic PAS algorithm only achieve speedups of $\sim 3\times$ in practice because of memory access bottlenecks [17]. This speed-up can be significantly increased by efficiently managing memory distribution and access patterns. This was first done in [18], where Lozenge cells were introduced, partitioning 3D space. Points that fall within a Lozenge cell would affect a known small subset of coefficients in the sparse transform domain. Since this involves a few coefficients — all necessary operations can be made within the cache— the speedup of the hologram generation process increases from $\sim 3\times$ to $\sim 87\times$. However, the block-wise plane wave approximation used in PAS introduces blocking artifacts across all PSFs. To mitigate these artifacts while retaining most of the computational efficiency, we propose a novel approach: reformulating PAS in a new sparse domain—the Gabor transform domain. Unlike previous work that uses Gabor representations to model diffraction, we analytically compute only the most significant Gabor coefficients of each PSF and use them within the PAS framework. This shift to a sparse Gabor domain allows us to preserve accuracy while maintaining high performance in CGH computation.

The contributions of this paper are as follows:

- We reformulate PAS to operate in the Gabor transform domain and introduce an analytical method to compute only the most significant Gabor coefficients of each PSF.
- We integrate this Gabor-based formulation with PAS's Lozenge-cell spatial partitioning, organizing coefficients to exploit GPU memory locality and caching efficiency.
- We implement the full method in C++/CUDA and quantitatively evaluate its accuracy and performance against conventional PAS and point-based CGH baselines.

The remainder of this paper is organized as follows. In Section 2, we provide a theoretical discussion on the essential concepts utilized throughout our study. Section 3 elaborates on the methodology and includes a detailed description of the proposed algorithm. In Section 4, we present the experiments conducted using the proposed algorithm on point clouds, offering insights into its practical application and performance. Finally, Section 5 concludes the paper by summarizing our findings and suggesting potential areas for future research.

2. Theoretical background

CGH models often rely on the superposition of PSFs under the Fresnel approximation. A PSF for a single point at coordinates (δ, ϵ, z) is defined as:

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

Here λ is the wavelength, a is the complex-valued amplitude, and i is the imaginary unit.

For a set of Q points P_j with $j = 1, \ldots, Q$, we can calculate the corresponding hologram as:

$$H(x, y) = \sum_{j=1}^{Q} P_j(x, y) = \sum_{j=1}^{Q} a_j \cdot \exp\left(\frac{\pi i}{\lambda z_j}\left[(x-\delta_j)^2 + (y-\epsilon_j)^2\right]\right) \tag{2}$$

This is a costly computation to implement in a brute-force manner. Several methods have been proposed, among them the accurate PAS [19], which we use as a basis for the proposed method.

Optics EXPRESS

The PAS method partitions the hologram into blocks, which can be calculated independently. For every PSF, each block will approximate the PSF (cf. Equation (1)) using a singular plane wave. This corresponds to one coefficient in the Fourier domain. We can optimize this problem by minimizing the error between the PSF and the plane wave within the block boundaries [18]:

$$\arg\min_{u,v,\varphi} \int_{-A}^{+A}\int \left| \exp\left(\frac{\pi i}{\lambda z_j}\left[(x-\delta)^2+(y-\epsilon)^2\right]\right) - \exp\left(2\pi i(ux+vy+\varphi)\right)\right|^2 dx\,dy \tag{3}$$

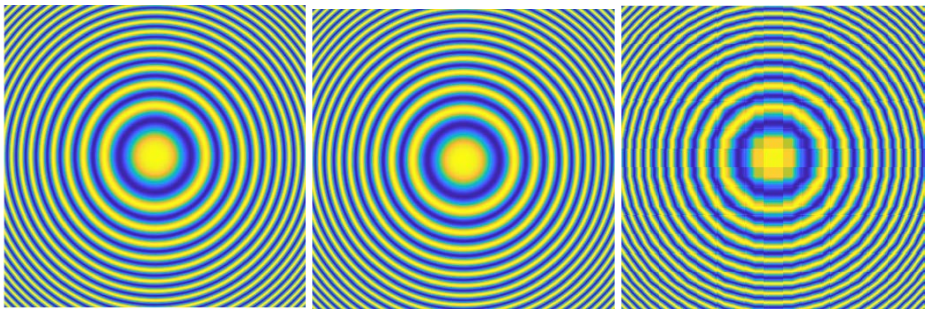
Where the first exponential expression represents the PSF, the second exponential expression is the plane wave to be optimized; u, v are the frequency components, φ is the phase delay, and $-A, +A$ are the block boundary coordinates, centered around zero.

Because we want to use the IFFT to retrieve the wavefield block from the computed frequencies efficiently, the possible frequencies to choose from are finite. The number of discrete frequency components is a free parameter, denoted to be $S \times S$ coefficients per block. S can be chosen independently from the block dimensions consisting of $B \times B$ hologram pixels. The solution to Eq. (3) was found in [18], namely

$$u = \left\lfloor \frac{(c_x-\delta)dS}{\lambda z} \right\rceil, \quad v = \left\lfloor \frac{(c_y-\epsilon)pS}{\lambda z} \right\rceil, \quad \varphi = \frac{(c_x-\delta)^2+(c_y-\epsilon)^2}{2\lambda z} - \frac{B}{2S}(u+v) \tag{4}$$

where $\lfloor\cdot\rceil$ is the rounding operator, c_x and c_y are the block center coordinates, d is the pixel pitch, and λ is the wavelength.

Since only one coefficient gets updated per point and per block, this speeds up the CGH process significantly. However, the approximation by plane wave segments results in lower accuracy, cf. Figure 1. We propose using a different transform and coefficient update scheme in order to increase accuracy while maintaining most of the speedup. We do this by using an analytical derivation to calculate the coefficients in the transform domain using chirplets.



(a) brute-force method. (b) Proposed method. (c) Accurate PAS method.

Fig. 1. Three calculations of a PSF, showing the real part of the hologram's signal; (a) contains the hologram calculated with the brute-force method with Eq. (1), (b) uses the proposed method, and (c) uses the accurate PAS algorithm.

2.1. Gabor atoms for sparse diffraction modeling

To overcome these limitations, diffraction modeling in Gabor space was proposed [20] by representing signals as *chirplets* – functions of the form:

$$\mathcal{G} = \left\{ f(t) = \exp(\alpha t^2 + \beta t + \gamma) \mid \alpha, \beta, \gamma \in \mathbb{C} \ \wedge \ \Re(\alpha)<0 \right\}. \tag{5}$$

Namely, the set of chirplets $\mathcal{G}$ comprises exponentials of quadratic polynomials with complex-valued ($\mathbb{C}$) coefficients, where the real part $\Re(\cdot)$ of α must be negative to ensure square integrability.

Optics EXPRESS

Chirplets can be used for efficiently encoding signals in holography [20]. For each chirplet, α, β, γ are complex parameters whose effect is described in Table 1.

Table 1. Interpretation of parameters in the chirplet family.

	α	β	γ
$\Re(\cdot)$	width	translation	scaling
$\Im(\cdot)$	chirp	modulation	phase shift

Calculating the diffraction in a transform space where mappings are sparse allows us to reduce the number of computations, sometimes drastically. Numerous candidates are possible. The Gabor transform provides a sparse representation by decomposing the signal into a set of Gaussians with different translations and frequency modulations. While the conventional PAS method approximates the PSF within a block to one plane wave, Gabor atoms overlap with each other, reducing boundary artifacts, and do not present inherent sparsity constraints, i.e., the amount of computed coefficients is selectable. This increases accuracy at the expense of the loss of a portion of the speedup. In this paper, we propose using the Gabor transform to calculate PSFs using a few well-chosen coefficients in the sparse domain.

A Gabor frame is a lattice of atoms generated by translating and modulating a window function g (cf. Figure 2). The lattice has coordinates $(m, n) \in \mathbb{Z}^2$, for which we can define each element using translation and modulation operators:

$$\{\mathbf{M}_m \mathbf{T}_n g\}_{m,n\in\mathbb{Z}} \tag{6}$$

for which translation and modulation can be computed for a chosen window function g:

$$\mathbf{T}_p : (\mathbf{T}_p g)(x) = g(x - p) \tag{7}$$

$$\mathbf{M}_q : (\mathbf{M}_q g)(x) = e^{2\pi i q x} g(x) \tag{8}$$

We choose the Gaussian window function for our use case due to its compatibility with the chirplet representation. Using Eq. (5), we can represent a Gabor atom with a modulation q, a translation p as a chirplet with $\alpha = \sigma$, $\beta = (iq - 2p\sigma)$ and $\gamma = (p\sigma - iq)p$:

$$F_{p,q}(x) = \exp\left(\sigma x^2 + (iq - 2p\sigma)x + (p\sigma - iq)p\right) \tag{9}$$

To calculate the diffraction of a hologram, we would need 2D Gabor atoms. This can be generalized using a separable definition along the x and y directions; we can write it as the multiplication of two 1D Gabor atoms:

$$G_{p_x,p_y,q_x,q_y}(x, y) = F_{p_x,q_x}(x) \cdot F_{p_y,q_y}(y) \tag{10}$$

Where the functions F represents the Gabor chirp as defined in Eq. (9), with each their translations $\{p_x, p_y\}$ and modulations $\{q_x, q_y\}$ along x and y.

Figure 3 shows a PSF in Gabor space. Its sparsity will allow us to approximate the PSF in Gabor space using only a few well-chosen frequency coefficients per translation.

2.2. Defining a PSF in Gabor space

PSFs can also be represented by chirplets. Noting that a PSF is separable along x and y, we can define a single 2D PSF by multiplying two chirplets, one for each direction. For a point with

Optics EXPRESS

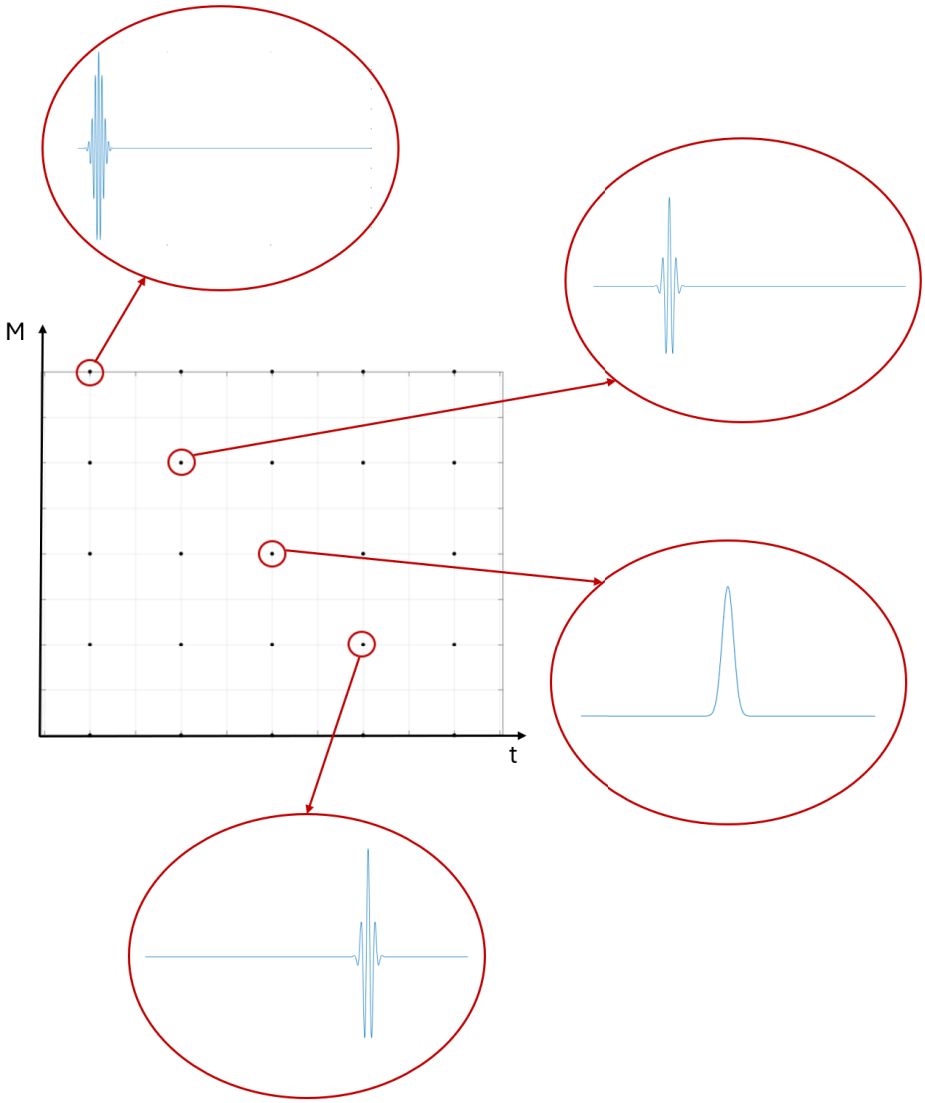


Fig. 2. The Gabor frame: Gaussian windows translated in space (t axis), modulated by frequency (M axis). Several lattice points have their corresponding Gabor atoms shown, plotting the real part of the respective signals.

Optics EXPRESS

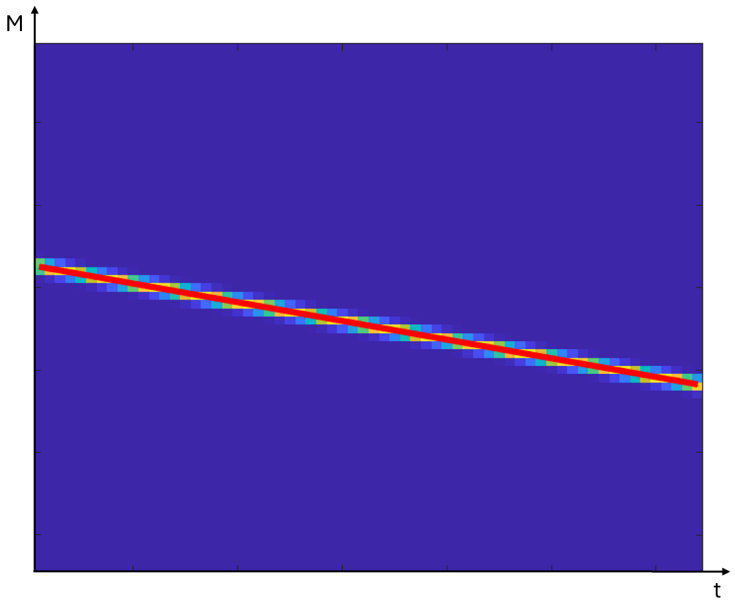


Fig. 3. One-dimensional (1D) PSF shown in the Gabor domain. The array represents the Gabor coefficient magnitudes, showing that most coefficients are (near-)zero. Only a few coefficients near the PSF's instantaneous frequency (red line) Eq. (4) are significant. This sparsity can be leveraged to reduce the computational cost.

coordinates (δ, ϵ, z), we get

$$P(x, y) = \psi(x, \delta) \cdot \psi(y, \epsilon). \tag{11}$$

Using Eq. (5), we can define the function ψ as

$$\psi(t, d) = \exp\left(\frac{i\pi}{\lambda z}(t^2 - 2dt + d^2)\right). \tag{12}$$

However, note that the real part of the PSF chirplet's quadratic coefficient is zero, denoted α in Eq. (5), making it a "degenerate" chirplet. This does not pose a problem in practice, as its inner product with Gabor atoms whose $\Re(\alpha)<0$ will always be square-integrable.

Just like the PSF, 2D Gabor atoms are also separable along x and y, allowing us to focus on the separate 1D components to determine the coefficient values. We can calculate a specific Gabor coefficient value analytically by computing the inner product between the corresponding Gabor atom and the PSF. The inner product is defined by

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)\,\overline{g(x)}\,dx. \tag{13}$$

For a PSF chirplet $f(x) = \exp(\alpha_1 x^2 + \beta_1 x + \gamma_1)$ and a Gabor atom $g(x) = \exp(\alpha_2 x^2 + \beta_2 x + \gamma_2)$. The product of chirps $f(x)$ and $g(x)$ is given by:

$$f(x) \cdot g(x) = \exp\left((\alpha_1 + \alpha_2)x^2 + (\beta_1 + \beta_2)x + (\gamma_1 + \gamma_2)\right), \tag{14}$$

That is, the two chirplets will form a new chirplet $u(x) = \exp(\alpha x^2 + \beta x + \gamma)$, for which we can calculate the integral over $\mathbb{R}$, giving us the corresponding Gabor coefficient:

$$\int_{-\infty}^{\infty} u(x)\,dx = \sqrt{\frac{\pi}{-\alpha}}\exp\left(\gamma - \frac{\beta^2}{4\alpha}\right). \tag{15}$$

Optics EXPRESS

Finally, we should efficiently characterize the coefficients that contribute the most to the PSF signal. The instantaneous frequency of a PSF can be calculated using Eq. (4). The resulting graph (Fig. 3) shows that Gabor coefficients become more significant as their modulation frequency is closer to the PSF's instantaneous frequency.

We define a parameter K determining the degree of sparsity, where we want to update $K \times K$ different modulations per translation of the 2D Gabor transform. Given a limited budget of coefficients to update, we should thus systematically only update the most significant ones.

To determine these coefficients, we first calculate the central frequency indices (m, n) using Eq. (4). Once the central index is determined, we form two sets of indices,

$$\Big\{\lfloor m - K/2 + 1 \rceil, \ldots, \lfloor m + K/2 \rceil\Big\} \quad \text{and} \quad \Big\{\lfloor n - K/2 + 1 \rceil, \ldots, \lfloor n + K/2 \rceil\Big\},$$

for the horizontal and vertical modulations, respectively. Each pair (m', n') in these sets represents a candidate modulation in the $K \times K$ block.

For every candidate frequency index, we compute a Gabor coefficient by taking the inner product of the Gabor atom – generated using its specific modulation and translation – with the PSF. This inner product quantifies the contribution of the PSF at the chosen modulation and ensures that only the coefficients with the strongest contributions (i.e., those with the most significant magnitudes) are updated.

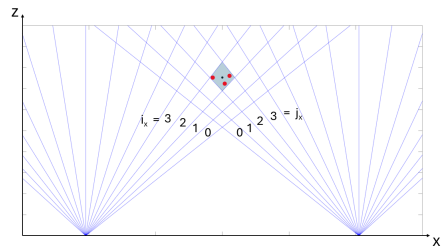
By selectively updating these $K \times K$ coefficients, the method balances computational efficiency with accuracy, updating only those coefficients that most effectively represent the PSF's structure in the Gabor space.

2.3. Memory efficiency via phase-space tiling

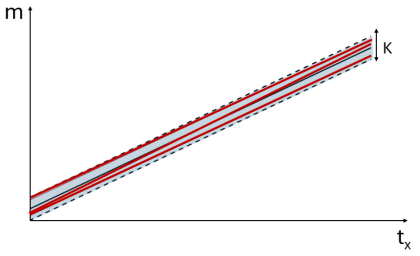
The final obstacle to tackle is memory management. While we could implement a straightforward method where all Gabor coefficients are kept in one large contiguous array, this would result in a memory bottleneck that would significantly limit the achievable speedup. In [18], the authors proposed to partition the point cloud into Lozenge cells. Points within these cells would affect a small square-shaped area in the phase space domain, i.e., the space of Gabor coefficients. One such cell is shown for a 1D hologram in Fig. 4(a).

Given a single PSF and its corresponding cell, we know that the most significant coefficients in phase space will fall within known bounds, cf. Figure 4(b,d). For a chosen p_x, we can initialize an array of size K where we store the corresponding coefficients, as derived in the previous section. Writing these coefficients directly to global memory will cause a significant degradation in speed. However, since this array of coefficients will be small, we can cache all the computations, speeding up the process significantly. The same can be said for a 3D point. Its corresponding frequency inside a block for an arbitrary (p_x, p_y) combination will fall within a square array of size $K \times K$, yielding the same speedup strategy to more efficiently compute a PSF in phase space (Fig. 5).

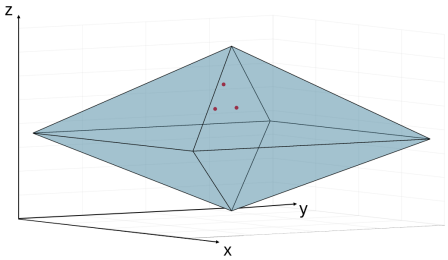
Optics EXPRESS



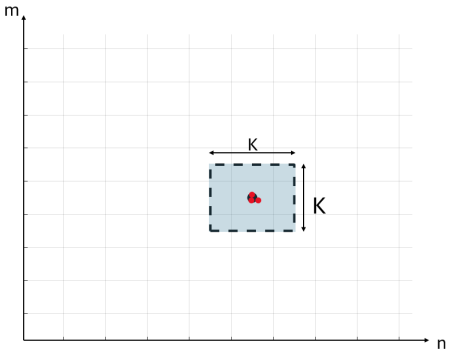
(a) Partition of space into Lozenge cells.



(b) Phase space representation



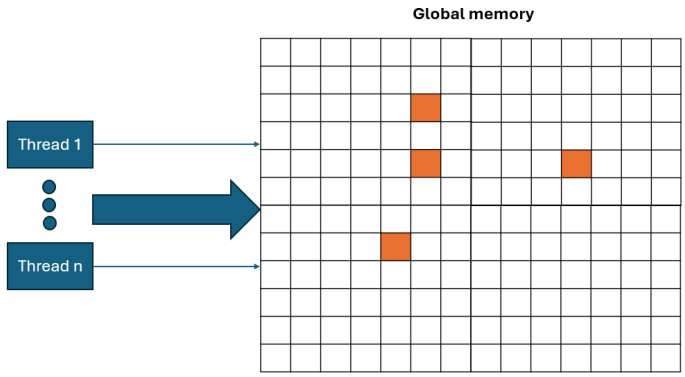
(c) Lozenge cell in 3D space.



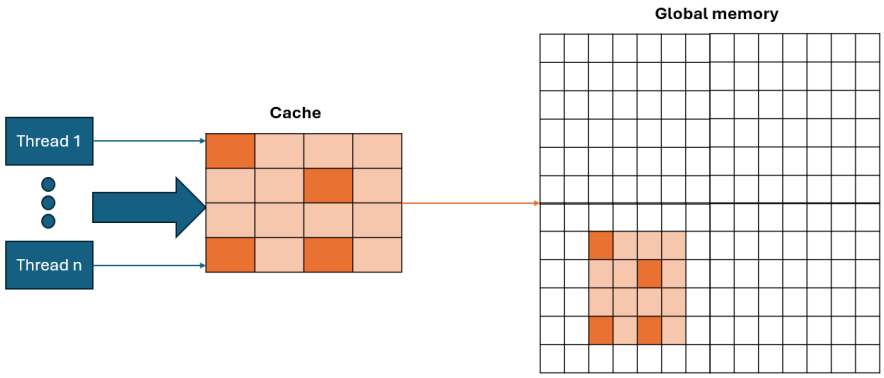
(d) Phase domain for three PSFs.

Fig. 4. (a) shows 3 exemplary points within a Lozenge cell, this representation shows (x, z) component of a point. Each cell can be uniquely determined by the cones (i_x, j_x). We can describe the (y, z) components of the grid in a similar fashion using (i_y, j_y). (b) All possible points in the Lozenge cell cover a parallelogram-shaped area in phase space. The exemplary points will map to lines in phase space. For a single translation p_x, the coefficients will fall within a K-sized 1D array. (c) shows three exemplary 3D points in a Lozenge cell. (d) shows the frequency space for one translation block (p_x, p_y). All significant active frequencies for points within the same Lozenge cell will fall within a square array of size $K \times K$.

Optics EXPRESS



(a) Conventional PAS method: each thread writes directly to global memory.



(b) Lozenge adaptation: threads write to cache, then flush to global memory.

Fig. 5. Comparison of memory write strategies in PAS implementation.

Optics EXPRESS

3. Methodology

The proposed algorithm accelerates point-based CGH by employing the Lozenge-cell method. The accuracy is increased by calculating diffraction in the Gabor space. This process is shown in Fig. 6. The algorithm consists of (1) Lozenge grid initialization and (2) kernel-based coefficient computation.

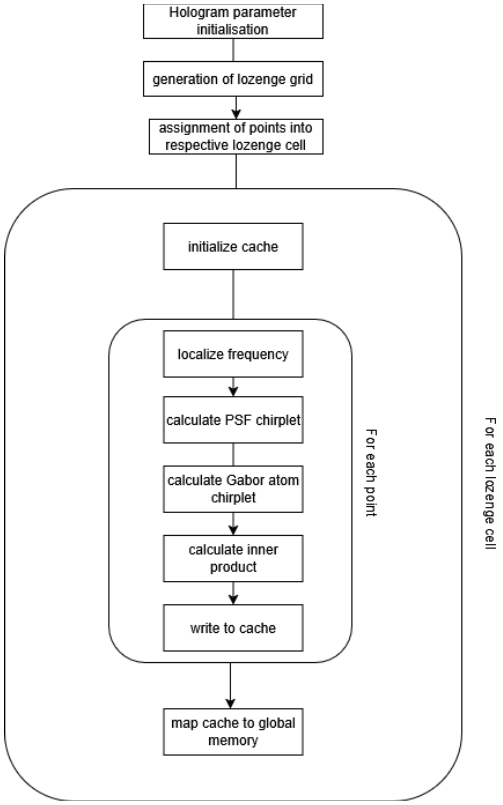


Fig. 6. A flowchart of the proposed algorithm.

3.1. Lozenge grid initialization

Given a point cloud

$$P = \{p_i \mid p_i = (\delta_i, \epsilon_i, z_i),\ i = 1, 2, \ldots, N\}, \tag{16}$$

we first construct a Lozenge cell grid and map every point to its corresponding cell. This grid is constructed by taking all combinations of cone intersections at the hologram corner points, cf. Figure 4. All cones cover a total angle equal to the maximum viewing angle $\theta_{max} = \lambda/(2d)$ where d is the pixel pitch and λ is the wavelength. Figure 4 shows intersections of these cones. Their intersection divides the plane into Lozenge cells, each corresponding to a localized Gabor (phase space) region. Finally, we map each point to its own Lozenge cell; an in-depth explanation on the mapping strategy of the Lozenge cells is provided in [18]. To efficiently assign each point to its Lozenge cell, we define the mapping:

$$\omega : t \mapsto \left\lfloor \frac{\lambda z}{2\pi} - \frac{td}{\pi} \right\rfloor; \quad i_x = \omega(\delta); \quad i_y = \omega(\epsilon); \quad j_x = \omega(Nd - \delta); \quad j_y = \omega(Nd - \epsilon). \tag{17}$$

Optics EXPRESS

This provides a tuple of indices for a point with coordinates (δ, ϵ, z). From this tuple, we can uniquely assign an index ℓ to every point based on its location, determined by:

$$\ell = r_x \cdot (r_x^2 + r_x + 1) + 3i_x \cdot (r_x + 1) + \frac{1}{2} i_y \cdot (r_y + 1) + i_y; \quad r_x = i_x + j_x; \quad r_y = i_y + j_y. \tag{18}$$

where r_x is the depth level for the x component, and r_y is the depth level for the y component.

3.2. Coefficient computation

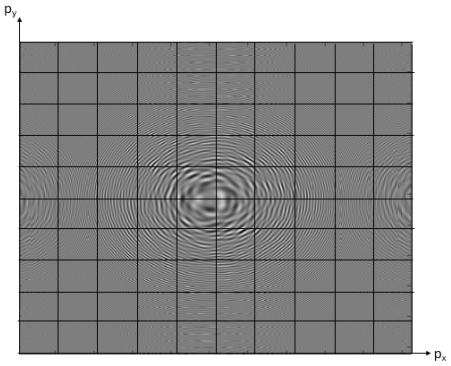
To efficiently implement the algorithm, we use GPU parallelization with C++/CUDA to calculate all coefficients. In CUDA, the GPU memory is modeled in a hierarchical memory structure. Registers are the fastest unit to write to, but they are limited in size and belong to individual threads, so they are not shareable with other threads. The cache serves as a storage pool for threads within a thread block in CUDA. All threads can access the cache of their own blocks. While slower than the registers, this shared storage is significantly faster than the global GPU memory that all blocks can access. For our implementation, we call on a kernel for every cell sequentially. In the kernel, we map the translations (p_x, p_y), which comprise one PAS block, to different thread blocks. In every thread block, threads are mapped to frequency modulations (m, n) Fig. 7. Each block will store its partial frequency data inside the cache, which is only possible because we know the affected frequency values inside a Lozenge cell will fall within the precalculated $L \times L$ bounds, where L is the cache size. Once a thread block is fully looped through, all points in that Lozenge cell will be processed, and the complete frequency spectrum for the Gabor PAS block with indexes (p_x, p_y) will be computed. We then write the data from this cache to the global memory, which contains the full 4D array with all blocks and frequencies. This writing operation is done once per kernel, significantly increasing the speedup of the algorithm.

Furthermore, we group our Gabor PAS blocks in batches to use the cache more efficiently. For each Lozenge cell, a single thread block needs an $(L \times L)$ sized array to store all frequency components; we chose $L = 8$. Since the storage capacity of the cache is higher than L^2 complex-valued floating-point coefficients, we can batch multiple Gabor PAS blocks to a single thread block. For a batch size b, the cache utilization increases to $b \times L \times L$. It is important to note that we can make a trade-off between the cache size, batch size, and number of coefficients per PSF. Experimentally, we found that $K = 4$ coefficients per PSF per direction yield a low error margin (<1%), where K is the selectable sparsity parameter. These coefficients were mapped to an $L \times L$-sized cache, for which we found a batch size of 96 to be nearing the memory limits the GPU could manage. Thus, $b = 96$ Gabor PAS blocks were batched and assigned per thread block.

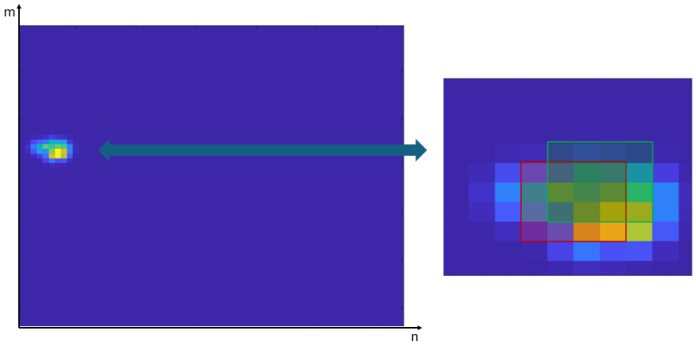
A CUDA kernel is launched per Lozenge cell to compute contributions from all its assigned points. Each kernel:

1. Initializes a local cache to store intermediate Gabor coefficients.
2. Retrieves the block and thread index.
3. Translates block index to specific translations (p_x, p_y) for the Gabor domain.
4. For each point p_i in the Lozenge cell:
    - We compute instantaneous frequencies (m, n) via Eq. (4).
    - Threads generate Gabor atoms for (x, y) components from translation and frequency indices. Each thread will generate a different frequency centered around (m, n).
    - We calculate the inner product between the PSF chirplet Eq. (12) and Gabor atoms Eq. (9). Exploiting separability, the 2D inner product factors into x and y components. Each inner product is calculated using Eq. (13).

Optics EXPRESS



(a) Blocks for each Gabor atom's translation.



(b) Frequency spectrum for a single (p_x, p_y) block.

Fig. 7. (a) shows a hologram grid for points clustered in 1 Lozenge cell; this grid consists of blocks that are represented by their block index (p_x, p_y) in the CUDA kernel. (b) shows the Gabor transform of a single block. 4 threads calculate 16 Gabor atoms using (p_x, p_y) for the translation component, and (m, n) for the frequency modulation. The inner product of these Gabor atoms with the PSF is mapped into the cache (right). The values corresponding to 2 separate PSFs are shown in green and red. After this is done for all PSFs, each block writes the cache to global memory (left).

- The results are accumulated in the cache.

5. We map the cached coefficients to global memory after processing all points.

The per-cell cache minimizes global memory access. Since all points in a cell update a limited region of the Gabor domain, caching enables coalesced writes. This contrasts with the PAS method, where concurrent writes to global memory degrade performance (Fig. 5). The size of this cache is a freely chosen parameter, but will typically be larger than the cache in the Lozenge cell method using a single plane approximation per block. This is because the Gabor domain doesn't approximate the PSF using one coefficient; as such, a Gabor transform will need a larger cache to represent a PSF accurately.

4. Experiments

The CGH can be visually analyzed using an optical setup with a spatial light modulator (SLM). The interference pattern corresponding to the CGH is encoded onto the SLM illuminated with a coherent reference beam. To calculate this interference pattern, we compare our proposed method with the Brute-force method (Eq. (2)), which serves as the ground truth, and with the accurate

Optics EXPRESS

PAS method, which is also accelerated with the Lozenge cell implementation. All methods were implemented in C++20 and CUDA 12.6.

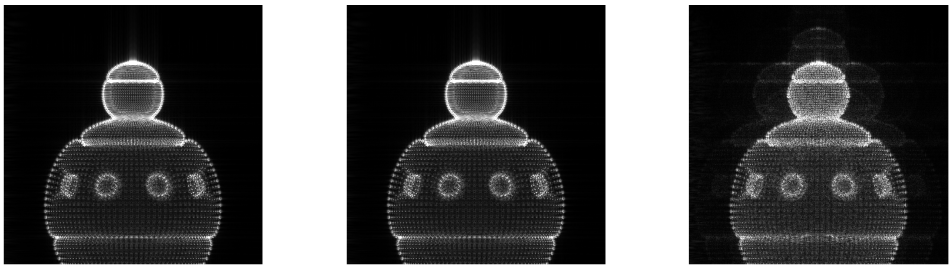
We tested the proposed algorithm for multiple point clouds. For the reference brute-force method, each pixel value was computed by a separate thread, looping over all points in the point cloud. We also calculated the diffraction in the Gabor space using the proposed method and compared the computational speed of the different methods. We use $K = 4$, that is, 4×4 coefficients per block per point, as opposed to one coefficient per block per point used in the accurate PAS method.

4.1. Accuracy analysis

The accuracy was compared with a GPU-optimized version of the APAS method [17] for a random point cloud of a hundred points. Since the CGH algorithms compute a linear superposition of PSFs, the shape of the point cloud should not have much impact on the accuracy measurements. We calculate the error using the normalized mean-square error (NMSE):

$$\mathrm{NMSE}(f,\hat{f}) := \frac{\|f-\hat{f}\|_2}{\|f\|_2}, \tag{19}$$

where f is the ground truth hologram, calculated using the brute-force method without any approximations, $\hat{f}$ is the target hologram whose quality we evaluate, and $||\cdot||_2$ is the Euclidean (L_2) norm. For a hologram calculation using 4×4 coefficients per PSF, the NMSE values can be found in Table 2 for 100 randomly sampled points, the chess point cloud, and the bi-plane point cloud. The APAS method was implemented using a stereogram block size of 16×16 and an oversampling factor of 4. By trading off some computational speed, we achieve a significant accuracy improvement of about four orders of magnitude. A cropped reconstruction of the chess piece is shown in Fig. 8. Approximating the point spread function (PSF) using a single plane wave leads to noticeable degradation in visual quality, introducing artifacts and loss of detail. In contrast, using a Gabor-based sparse representation yields a reconstruction that closely matches the reference result, effectively reducing artifacts.



(a) Reference (brute-force) (b) Proposed (Gabor-based) (c) APAS

Fig. 8. Cropped reconstructions for three methods. The APAS reconstruction exhibits strong ringing and ghosting artifacts, particularly around object boundaries. In contrast, the proposed Gabor-based approach significantly reduces these artifacts and closely matches the ground truth produced by brute-force CGH.

4.2. Speed analysis

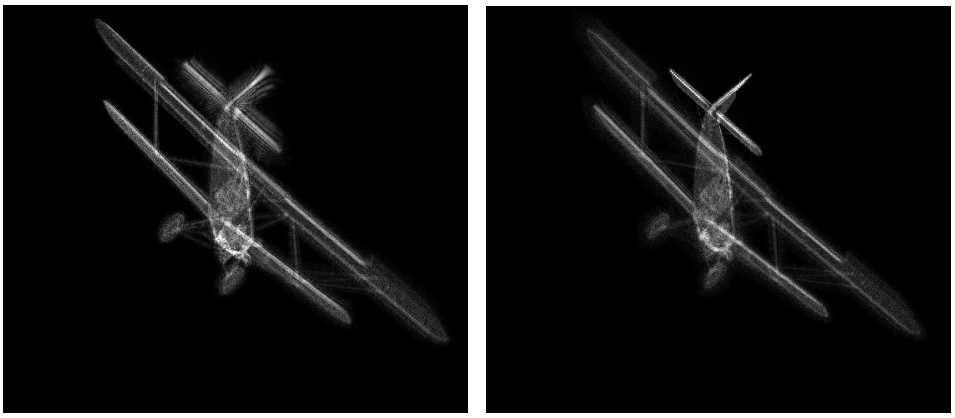
We calculated a hologram with a pixel pitch of $d = 4$ μm and wavelength of $\lambda = 633$ nm. The number of modulations was chosen to be 64, and the translation step size was 42 pixels, ensuring sufficient overlap between the windows at multiple Gabor PAS blocks. We did this for varying

Optics EXPRESS

Table 2. NMSE for APAS and proposed method.

Hologram	100 points	Chess pieces	Bi-plane
APAS	5.08×10^{-2}	4.45×10^{-2}	4.44×10^{-2}
Proposed method	7.52×10^{-6}	4.15×10^{-6}	1.37×10^{-6}

numbers of pixels, ranging from 2048×2048 to 16384×16384. For each configuration, the proposed method was run 10 times, and the results were averaged. The first point cloud is the bi-plane model, consisting of 454,595 points. We chose $K = 4$, keeping this constant throughout all experiments to ensure consistency. In doing so, we can accurately compare the speedup as a function of the resolution of the hologram. The algorithm ran on a machine with an RTX4060 GPU, a 3.80 GHz AMD Ryzen 5 7600 6-Core Processor, 32 GB of RAM, and a Windows 11 OS. To contextualize the performance, we compared our method against both the brute-force point-based CGH method and the wavefront recording plane (WRP) method. The WRP method was implemented using a single plane, placed at a distance of 1 mm from the closest point in the point cloud. As shown in Table 3, the proposed method demonstrates a substantial speedup compared to the brute-force baseline. For a hologram of resolution 16384×16384 pixels, the proposed method calculates the hologram 47 times faster than the optimized brute-force method. The resulting numerical reconstruction is shown in Fig. 9.



(a) Front focus (b) Back focus

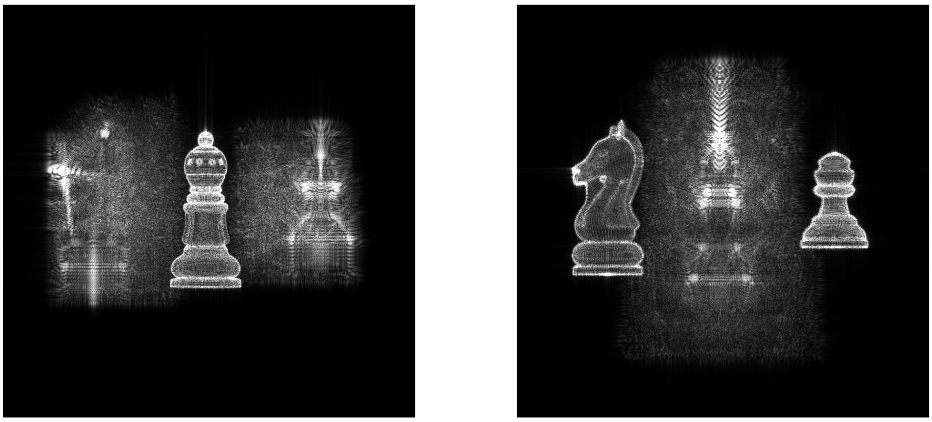
Fig. 9. Numerical reconstruction of the hologram showing the focus on the objects' front plane (a) and back plane. (b).

Table 3. Speedup factor comparison for the Bi-plane model, relative to the brute-force reference CGH algorithm. The APAS model uses a stereogram block size of 16×16 and an oversampling factor of 4, except for the configuration denoted by (*), which uses a $2\times$ oversampling factor due to memory limitations.

Hologram definition (x and y)	2048	4096	8192	16384
WRP method	×2.0	×2.1	×2.1	×2.1
Proposed method	×29	×36	×44	×47
APAS	×92	×135	×122	×153*

Optics EXPRESS

The second point cloud used comprised models of chess pieces, a model consisting of 69,340 points; the speedup results are summarised in Table 3 for the plane model and in Table 4 for the chess model. The resulting numerical reconstruction is shown in Fig. 10.



(a) Bishop (b) Knight and pawn

Fig. 10. Numerical reconstruction of the hologram showing the focus on the nearest object (a) and the furthest objects (b).

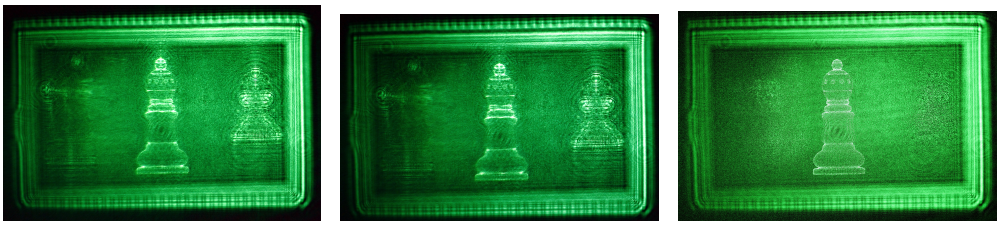
Table 4. Speedup factor comparison for the chess model, relative to the brute-force reference CGH algorithm. The APAS model uses a stereogram block size of 16 × 16 and an oversampling factor of 4, except for the configuration denoted by (*), which uses a 2× oversampling factor due to memory limitations.

Hologram definition (x and y)	2048	4096	8192	16384
WRP method	×1.5	×1.6	×1.6	×1.6
Proposed method	×25	×33	×43	×46
APAS	×42	×69	×39	×83*

4.3. Optical experiments

In the optical configuration for a proof-of-concept demonstration, a fiber-coupled laser diode (LD) with a central wavelength of 520 nm and a spectral bandwidth of 10 nm (FOLS-13-RGB-SM) was collimated slightly off-axis using an adjustable aspheric collimator (CFC5A-A, Thorlabs) and expanded approximately threefold using a beam expander (GBE03-A, Thorlabs). The beam was then reflected at approximately 45° by a beam splitter and fully illuminated a phase-only reflective spatial light modulator (SLM) (Holoeye, GAEA-2.1 Phase-Only LCOS-SLM; 3.74 μm pixel pitch, 4160 × 2464 resolution). The beam reflected from the SLM passed back through the beam splitter and a linear polarizer (LPVISE2X2, Thorlabs). An image sensor (Sony) was positioned after the polarizer to capture the holographic object at various focal distances, approximately 20 mm from the SLM. The brute-force method, proposed method, and APAS reconstructions can be found in Fig. 11.

Optics EXPRESS



(a) Brute-force (b) Proposed method (c) APAS

Fig. 11. Optical reconstructions using different methods: brute-force (a), proposed method (b), and APAS (c).

5. Conclusion

This paper presents a novel CGH algorithm integrating the Gabor transform with a caching strategy based on Lozenge cells. We improve on the PAS method by addressing its limitation of accuracy: blocking artifacts caused by approximating the PSF in a block with a single plane wave. We solve this by modeling the PSF as a chirplet and leveraging its sparsity in the Gabor transform space. The analytical formulation of chirplets enables efficient computation of the inner product of the Gabor transform for a PSF.

The proposed Lozenge cell tiling strategy further optimizes performance by localizing PSF contributions in phase space, minimizing global memory access through GPU cache utilization. We report a 47× speedup over brute-force methods for 16K holograms while keeping the NMSE about four orders of magnitude lower. The main drawback is the smaller speedup compared to the APAS method implemented using Lozenge cells and a single coefficient. There is also a higher implementation complexity in calculating diffraction using Gabor and the strategy to optimize memory bottlenecks as opposed to the more straightforward single coefficient calculations laid out in [18]. This work brings us closer to high-resolution, accurate, real-time video holography. Future improvements could be made in terms of speed by applying motion estimation algorithms to avoid calculating every new frame from the ground up. In terms of visual quality, occlusion is still not accounted for in the current implementation; in doing so, we could further increase the visual quality of the generated hologram frame.

Funding. Fonds Wetenschappelijk Onderzoek (G089424N, 12ZQ223N).

Disclosures. The authors declare no conflicts of interest.

Data availability. The 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.

References

1. D. Gabor, "A new microscopic principle," Nature **161**(4098), 777–778 (1948).
2. B. R. Brown and A. W. Lohmann, "Computer-generated binary holograms," IBM J. Res. Dev. **13**(2), 160–168 (1969).
3. D. Blinder, T. Birnbaum, T. Ito, *et al.*, "The state-of-the-art in computer generated holography for 3d display," (2022).
4. S. Yoshida, "High-speed full-color computer-generated holography using a digital micromirror device and fiber-coupled rgb laser diode," Appl. Opt. **63**(10), 2455–2461 (2024).
5. D. Pi, J. Liu, and Y. Wang, "Review of computer-generated hologram algorithms for color dynamic holographic three-dimensional display," (2022).
6. E. Sahin, E. Stoykova, J. Mäkinen, *et al.*, "Computer-generated holograms for 3d imaging: A survey," (2020).
7. M. H. Eybposh, V. R. Curtis, J. Rodríguez-Romaguera, *et al.*, "Advances in computer-generated holography for targeted neuronal modulation," Neurophotonics **9**(4), 041409 (2022).
8. C. Zhao, "Computer-generated hologram for optical testing: a review," (SPIE-Intl Soc Optical Eng, 2021), p. 18.
9. V. Petrov, A. Pogoda, V. Sementin, *et al.*, "Advances in digital holographic interferometry," J. Imaging **8**(7), 196 (2022).
10. X. Xie, X. Wang, C. Min, *et al.*, "Single-particle trapping and dynamic manipulation with holographic optical surface-wave tweezers," Photonics Res. **10**(1), 166–173 (2022).

Optics EXPRESS

11. F. Kaván, P. Psota, V. Lédl, *et al.*, "Multiple wavelength digital holography for freeform shape measurement and lens alignment," Appl. Opt. **62**(10), D138–D145 (2023).
12. R. H.-Y. Chen and T. D. Wilkinson, "Computer generated hologram from point cloud using graphics processor," Tech. rep. (2009).
13. Y. Zhang, H. Fan, F. Wang, *et al.*, "Polygon-based computer-generated holography: a review of fundamentals and recent progress [invited]," Appl. Opt. **61**(5), B363 (2022).
14. T. Ichikawa, T. Yoneyama, and Y. Sakamoto, "Cgh calculation with the ray tracing method for the fourier transform optical system," Opt. Express **21**(26), 32019 (2013).
15. T. Shimobaba and T. Ito, "Fast generation of computer-generated holograms using wavelet shrinkage," Opt. Express **25**(1), 77 (2017).
16. H. Kang, E. Stoykova, and H. Yoshikawa, "Fast phase-added stereogram algorithm for generation of photorealistic 3d content," Appl. Opt. **55**(3), A135 (2016).
17. D. Blinder and P. Schelkens, "Accelerating phase-added stereogram calculations by coefficient grouping for digital holography," in *Optics, Photonics and Digital Technologies for Imaging Applications VI*, vol. 11353 P. Schelkens and T. Kozacki, eds., International Society for Optics and Photonics (SPIE, 2020), p. 1135303.
18. D. Blinder and P. Schelkens, "Phase added sub-stereograms for accelerating computer generated holography," Opt. Express **28**(11), 16924 (2020).
19. H. Kang, T. Yamaguchi, H. Yoshikawa, *et al.*, "Acceleration method of computing a compensated phase-added stereogram on a graphic processing unit," Appl. Opt. **47**(31), 5784–5789 (2008).
20. D. Blinder, T. Birnbaum, and P. Schelkens, "Efficient numerical fresnel diffraction with gabor frames," Photonics Res. **13**(2), 330–339 (2025).