Phase added sub-stereograms for accelerating computer generated holography

Blinder, David; Schelkens, Peter

Published in: Optics Express

DOI: https://doi.org/10.1364/OE.388881

Publication date: 2020

License: Other

Document Version: Final published version

Citation for published version (APA):
Blinder, D., & Schelkens, P. (2020). Phase added sub-stereograms for accelerating computer generated holography. Optics Express, 28(11), 16924-16934. https://doi.org/10.1364/OE.388881
Phase added sub-stereograms for accelerating computer generated holography

DAVID BLINDER¹,²* and PETER SCHELKENS¹,²

¹Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), Pleinlaan 2, B-1050 Brussel, Belgium
²IMEC, Kapeldreef 75, B-3001 Leuven, Belgium
*dblinder@etrovub.be

Abstract: Phase-added stereograms are a form of sparse computer generated holograms, subdividing the hologram in small Fourier transformed blocks and updating a single coefficient per block and per point-spread function. Unfortunately, these algorithms’ computational performance is often bottlenecked by the relatively high memory requirements. We propose a technique to partition the 3D point cloud into cells using time-frequency analysis, grouping the affected coefficients into subsets that improve caching and minimize memory requirements. This results in significant acceleration of phase added stereogram algorithms without affecting render quality, enabling real-time CGH for driving holographic displays for more complex and detailed scenes than previously possible. We report a 30-fold speedup over the base implementation, achieving real-time speeds of 80ms per million points per megapixel on a single GPU.

© 2020 Optical Society of America under the terms of the OSA Open Access Publishing Agreement

1. Introduction

Computer generated holography (CGH) comprises a set of techniques to numerically simulate diffraction of light for various applications in holography and beyond. CGH is computationally challenging because of the nature of diffraction, since every 3D scene element can potentially affect all hologram pixels. It is especially important for holographic displays which require high-resolution holograms to be calculated at video rates.

Holographic displays are often cited as the ultimate 3D display technology [1], because in principle holograms can display virtual objects which are indistinguishable from real objects: they account for all human visual cues, such as parallax, occlusion, stereopsis, eye-focusing and no exhibit accommodation-vergence conflict. Many types of CGH algorithms exist [1,2]. Depending on the CGH method, features such as computational speed, visual realism, supported graphical effects and scene geometry might be traded off against each other. They can be classified as point cloud methods [3–8], layer-based methods [9–11], polygon methods [12–15], RGB+depth methods [16], ray-based methods [17–19] and sparse CGH [20–22].

This paper will focus on “phase-added stereograms” (PAS) [23–26]. They are a particular form of sparse CGH, relying on the fact that point spread functions (PSF) can be accurately approximated by only a few coefficients in a well-chosen transform domain. That way, the number of needed coefficient updates will be much lower than the hologram pixel count, speeding up calculations considerably while assuring minimal quality loss. The modus operandi of PAS is to divide the hologram into equally sized blocks and approximating the PSF in every block by a single plane wave. This corresponds to adding a single Fourier coefficient per block. Most of the effort pertaining to optimizing PAS has been dedicated to get the most accurate hologram approximation with the smallest number of coefficients. However, calculation speed is not only determined by the calculation complexity, but by bound memory throughput as well. In the PAS framework, the needed memory bandwidth per coefficient is the limiting factor over the required computational cost per coefficient. This will limit the the number of independently working
threads, the usability of the cache and preclude several forms of specialized implementation with low memory or complexity requirements, such as many types of FPGAs.

To address this challenge, we propose a new algorithm for computing PAS more efficiently. By partitioning the point cloud into our newly proposed cell construction, we guarantee that all points within a cell only affect a tiny number of coefficients per PAS block, called a “sub-stereogram”. This makes the algorithm cache-friendly and can substantially speeds up most PAS variants.

The paper is structured as follows: in Section 2 we first derive the optimal way for computing PAS coefficients in the Fresnel approximation regime. Then, in Section 3 we model the cell geometries by means of time-frequency analysis, characterize the tiling topology in 2D and 3D space, and model an efficient algorithm for binning the points into the different cells. In section 4, we experimentally validate the proposed solution quantitatively and qualitatively. Finally, we conclude in section 5.

2. Phase-added stereograms modelled with dictionary elements

In point based CGH we approximate objects by a discrete set of 3D points, which create a linear combination of point-spread functions (PSF). The Fresnel approximation of a single PSF $P$, laterally displaced with a translation $(\delta, \epsilon)$ and at a depth $z$ from the hologram plane is given by

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

(1)

where $i$ is the imaginary unit, $\lambda$ is the wavelength and $a \in \mathbb{C}$ is the point amplitude.

This expression can be used to calculate a hologram $H$ by summing over a collection of $Q$ points $P_j, j \in \{1, \ldots, Q\}$, each with their respective coordinates $(\delta_j, \epsilon_j, z_j)$ and amplitudes $a_j$:

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

(2)

Directly computing this sum is very computationally demanding, updating every hologram pixel $Q$ times. To accelerate computations, the CGH can be computed in a transform domain where PSFs are sparse, such that only a few coefficients need to be updated to achieve a good approximation of the reference PSF.

2.1. Phase-added stereogram as a dictionary model

We will focus on phase-added stereograms, which subdivide the holograms into blocks of $B \times B$ pixels, each described by their local Fourier domain. This was chosen because PSFs have been shown to be sparse in short time Fourier transform domains, more so than e.g. wavelets [21]. Every block is represented by a $S \times S$ coefficients corresponding to plane waves pieces with different frequencies, cf. Figure 1. When $S > B$, we have a variant of the accurate PAS method [25], effectively using an overcomplete dictionary $D_{mn}$:

$$D_{mn}(x, y) = \exp\left(\frac{2\pi i}{S}(mx + ny)\right); \quad x, y \in \{0, \ldots, B - 1\}; \quad m, n \in \{-S/2, \ldots, S/2 - 1\}$$

(3)

namely, every block is represented by a linear combination of $D_{mn}$, where $m$ and $n$ indicate the horizontal and vertical frequency, respectively. For every PSF, the goal is to locally approximate a PSF piece as closely as possible in every block by choosing the best matching $D_{mn}$ and assigning the right coefficient value. As $S$ gets larger, we sample Fourier space more finely and get more $D_{mn}$ to choose from, so we can better approximate the carrier frequency of the PSF in any given block, increasing the quality of the PAS.
This problem amounts to minimizing the energy difference between the target quadratic chirp (i.e., the PSF) as expressed in (1) and the planar wave within the block boundaries $[-A, +A]$ by solving for the right frequencies $m, n$ and phase delay $\phi$:

$$\arg \min_{\phi} \int_{-A}^{+A} \left| \exp \left( \frac{\pi i}{\lambda z} [(x - \delta_j)^2 + (y - \epsilon_j)^2] \right) - \exp (2\pi i(mx + ny + \phi)) \right|^2 \, dx \, dy$$

(4)

with half block size $A = \frac{Bp}{2}$. If we want to solve for other blocks not centered at the origin, we can translate the system equivalently by modifying $\delta$ and $\epsilon$, so this is a generic problem formulation. Note that we aim to minimize the difference between two phase-only functions, i.e., functions $f: \mathbb{R} \rightarrow \mathbb{C}$ where $|f(x, y)| = 1, \forall x, y \in \mathbb{R}$. Thus, we can use the Euclidean distance between two points on the unit complex circle to express the distance as a function of their phase difference:

$$|\exp(i\phi_1) - \exp(i\phi_2)|^2 = \sin (\phi_1 - \phi_2)^2 + (1 - \cos (\phi_1 - \phi_2))^2 = 2 (1 - \cos (\phi_1 - \phi_2))$$

(5)

making (4) equivalent to

$$\arg \min_{\phi} \int_{-A}^{+A} 1 - \cos \left( \frac{\pi}{\lambda z} [(x - \delta_j)^2 + (y - \epsilon_j)^2] - 2\pi(mx + ny + \phi) \right) \, dx \, dy$$

(6)

For well-chosen $m, n$, we can assume that the phase difference between the plane wave and the target PSF chirp is small over the block region. We can therefore use the Taylor approximation $\cos t \approx 1 - \frac{t^2}{2}$ valid for small values of $|t|$. This makes (6) equivalent to solving

$$\frac{\partial}{\partial \phi} \int_{-A}^{+A} \left( \frac{\pi}{\lambda z} [(x - \delta_j)^2 + (y - \epsilon_j)^2] - 2\pi(mx + ny + \phi) \right)^2 \, dx \, dy = 0$$

(7)

resulting in the sought phase delay (and omitting the index $j$)

$$\varphi = \frac{1}{2\lambda z} \left( \frac{2}{3}A^2 + \delta^2 + \epsilon^2 \right)$$

(8)

where the term containing $A^2$ can be ignored, because it will cause the same phase delay across all blocks. For a general block centered at position $(M_x, M_y)$, we get:

$$m = \left[ \frac{(M_x - \delta)pS}{\lambda z} \right], \quad n = \left[ \frac{(M_y - \epsilon)pS}{\lambda z} \right], \quad \varphi = \frac{(M_x - \delta)^2 + (M_y - \epsilon)^2}{2\lambda z} - \frac{B}{2S} (m + n)$$

(9)
where \([\lfloor . \rfloor]\) is the rounding operator. Repeating this process for every point in the point cloud and summing all computed coefficients will result in the Fourier domain representation of the hologram. We obtain the final PAS after applying a single IFFT per block and cropping the \(B \times B\) top-left corner of samples, as shown in Fig. 1. Because of the cropping and frequency oversampling of the FFT we need the added term \(\frac{B^2}{2}\) to obtain a correct phase for \(\varphi\) in (9).

3. Modelling sub-stereograms

There are a total of \(\frac{S^2}{B^2}N^2\) coefficients for a hologram of \(N \times N\) pixels. Since this large data structure does not fit in local caches, direct random access to update these coefficients is relatively slow. It is not straightforward to use multiple compute threads per PAS block, because threads interfere with each other requiring atomic access to variables. Hence, multi-level caching is needed for improved throughput and large amounts of memory are required, precluding its use in simpler hardware topologies such as many FPGAs. Even if such complex hardware is available, the maximum computation gain that can be obtained will still be inferior to the proposed solution; this has been analyzed in detail in [27].

The goal is to partition the point cloud into smaller groups called cells, each only affecting a small subset of the total number of coefficients. That way, these small number of coefficients can be cached for significantly improved computation throughput. Every thread is responsible for its own PAS block, so that it will never interfere with coefficients accessed by other threads (cf. Fig. 2). In the conventional approach, a block typically does not fit in cache and memory access patterns are unbounded and different across blocks, thus requiring storage in global memory. This will bottleneck the calculations. In the proposed method, only a "sub-stereogram" is active per cell. Once the sub-stereogram is fully computed, only then the data is transferred to global memory, which will considerably reduce the required global memory accesses from once per point to once per cell.

3.1. Lozenge cells

Suppose we only want to update a \(K \times K\) sub-block of the dictionary coefficients per \(S \times S\) block where \(K < S\), cf. Figure 2(b), which we will designate as a "sub-stereogram". We want to answer the following question: what is the spatial region of which the contained 3D points will solely affect the considered sub-stereogram?

This can be derived with time-frequency analysis (a.k.a. space-frequency, phase space or Wigner space). It portrays \(n\)-dimensional signals with a \(2n\)-dimensional representation simultaneously describing time (or space) and frequency. For a one-dimensional PSF signal, this
space can be drawn as a chart, where the horizontal axis shows the evolution of the signal over space and across PAS blocks, and the vertical axis shows the frequency value. The instantaneous frequency $\nu(x)$ of a (one-dimensional) PSF $P(x)$ is given by

$$\nu(x) = \frac{1}{2\pi} \frac{\partial}{\partial x} \angle P(x) = \frac{x - \delta}{4z}$$

(10)

where $\angle$ is the complex argument (or angle). The curve $\nu(x)$ is a line with a slope inversely proportional to $z$ and displaced by a lateral translation $\delta$. Thus, this curve tells us what PAS Fourier coefficient should be updated for a 3D point at a given spatial coordinate $x$.

The size $K$ of a sub-block will dictate the maximal available bandwidth at any given position. Since the signal frequency is proportional to the incidence angle, the set of supported incidence angles will form a cone, as exemplified by the red cones in Fig. 3(b). We now consider a parallelogram-shaped region with a height given by $K$ shown in Fig. 3(a). By taking the intersection of all corresponding cones, we obtain a quadrangle looking like a distorted rhombus, which we will call a “lozenge cell”. Every point in 3D space will have a resulting time-frequency curve lying fully in the parallelogram if and only if it is found in the lozenge cell. Because the time-frequency curves of Fresnel modelled PSFs are lines, they are fully characterized by two points in the time-frequency diagram. We can therefore guarantee that a 3D point satisfies this condition if its PSF has an instantaneous frequency within two specific frequency bands at the extremities of the hologram.

![Fig. 3.](image)

**Fig. 3.** Example of a parallelogram-shaped region in the time-frequency domain (a), and how it maps to scene space (b). Note how points in one space map to lines in the other space and vice versa. The red bandwidth intervals in (a) map to cones in (b), whose intersection forms a lozenge cell. The union of all points in the lozenge cell map to the bundle of all possible lines lying fully in the parallelogram region in (b). Three example points are shown: the dark blue and green points are in the lozenge cell, while the orange point is outside, causing its corresponding time-frequency line to (partially) lie outside of the parallelogram.

Given the irregular shape of lozenge cells, a new question arises: how do we efficiently partition any 3D point cloud into the minimal number of cells?

### 3.2. Partitioning space into lozenge cells

We can seamlessly tile space with lozenge cells in the 2D case using the construction shown in Fig. 4(a). We subdivide the total supported hologram viewing angle into cones at both hologram edges depending on the ratio $\frac{k}{k}$, indexed by $i_\text{x}$ and $j_\text{x}$, respectively. Every cell is thus uniquely characterized by its coordinates $(i_\text{x}, j_\text{x})$. The lozenge cells are stacked in rows, whose index is
given by \( r_x = i_x + j_x \), consisting of \( r_x + 1 \) cells per row. Each cell in the same row also has the same minimum and maximum \( z \) values given by \( d[r_x] \) and \( d[r_x + 2] \), respectively, with

\[
d[r_x] = \frac{p^2 N}{\lambda(1 - r_x F)} .
\]

(11)

where \( F \) is the reciprocal of the number of chosen conic subdivision, which can be set to \( F = \frac{K}{3} \).

Note that the union of all cells, marked in blue in Fig. 4(a), will precisely form the “aliasing-free” cone [1]. Points outside of the cone will cause aliasing, inducing frequencies in the hologram surpassing the Shannon bound.

![Diagram of the 3D point cloud subdivision by combining two 2D lozenge cell partitionings.](image)

**Fig. 4.** Diagram of the 3D point cloud subdivision by combining two 2D lozenge cell partitionings. Every cell in the \( x-z \) plane is characterized by its coordinates \( i_x \) and \( j_x \) (a). The row index is given by the sum of the coordinates, as shown for the orange cells for the example \( r_x = 2 \). Every lozenge cell in (a) can have an intersection with a cell in (b), so long as they have an overlap along \( z \), as shown by the green cells.

We will now generalize the calculations to 3D space. Because the Fresnel PSF formulation is separable, every point now has to satisfy four conditions, one for each of the four edges of the hologram. For every hologram side, we obtain a wedge-shaped region whose edge is coincident with the hologram edge, and whose opening angle and orientation is given by the position and width of the active sub-block’s frequency band. The intersection of these four wedges results in an anisotropically distorted octahedron, as shown in Fig. 5.

Unfortunately, we cannot seamlessly tile 3D space with these shapes, because it is mathematically impossible to seamlessly tile space with octahedrons. We therefore will have some redundancy if we want to fully cover 3D space, i.e. some lozenge cells will inevitably overlap.

A first solution is to use the Cartesian product of the 2D lozenge cell tiling for \( x \) and \( y \); namely, identify and combine the cell indices for the \( xz \)-plane and \( yz \)-plane independently. However, this is redundant because many intersections will be empty: most lozenge cell are bounded in \( z \), given by the layer values in (11); if the \( z \)-ranges of the two 2D lozenge cells in both planes do not overlap, then no points can simultaneously lie in both cells at once.

We eliminate the empty cells by taking into account the observation that every 2D lozenge cell in the \( xz \)-plane extruded along the \( y \)-axis will only intersect 2D cells in up to 3 rows in the \( yz \)-plane, provided the same frequency band partitioning is used for both dimensions. This is illustrated by the green lozenge cells in Fig. 4(b).
To partition the point cloud, we need an efficient way to determine in what lozenge cell any given point \((\delta, \epsilon, z)\) lies. This can be computed as follows, using the floor operator \(\lfloor \cdot \rfloor\):

\[
\alpha: t \mapsto \lfloor \frac{1}{2F} - \frac{lp}{F\lambda z} \rfloor; \quad i_x = \alpha(\delta); \quad i_y = \alpha(\epsilon); \quad j_x = \alpha(Np - \delta); \quad j_y = \alpha(Np - \epsilon). \tag{12}
\]

We can linearize every valid tuple \((i_x, i_y, j_x, j_y)\) uniquely into a single number \(\ell\):

\[
\ell = r_x \cdot (r_x^2 + r_x + 1) + 3i_x \cdot (r_x + 1) + \frac{1}{2}r_y \cdot (r_y + 1) + i_y; \quad r_x = i_x + j_x; \quad r_y = i_y + j_y; \tag{13}
\]
simplifying the point storage structure into a linear array of point lists.

### 4. Experiments

The CGH can be viewed in an optical setup by encoding the computed pattern on a Spatial Light Modulator (SLM) or printed on e.g. a glass plate [28]. This is done by quantizing the amplitude and/or phase of the hologram depending on the modulation capabilities of the display device; most used SLMs in electro-holography are “phase-only” or “kinoform”. The object can be viewed by directly illuminating the display with a coherent light source, typically using a planar wave for the reference beam. More info on point-based CGH display setups can be found in [26,29].

We investigate two aspects of the proposed algorithm in this section: (1) the impact of the parameter selection on quality and speed, and (2) measurements of the gains in calculation speed on a detailed CGH example, including numerical reconstructions from several viewpoints.

#### 4.1. Algorithm parameter configuration

The main algorithm parameters to be chosen are the hologram subdivision block size \(B\), the coefficient block size \(S\) and the sub-block size \(K\) for the sub-stereogram computation.
The values of $B$ and $S$ are inherited from the accurate PAS method [25], and will affect the approximation accuracy. We measure the approximation error using the the normalized mean-square error (NMSE):

$$\text{NMSE}(P, \hat{P}) := \frac{\|P - \hat{P}\|_2}{\|P\|_2}$$

(14)

where $P$ is the reference PSF, $\hat{P}$ is the approximated PSF using the proposed PAS method and $\|\cdot\|_2$ is the Euclidean norm. We report the error in Table 1 for different values of $B$ and $S$, averaged over 100 random point instances. The accuracy of the PAS approximation will always improve when $B$ decreases or $S$ increases (when keeping the other parameter constant, respectively), because they are a superset of the possible PAS with higher $B$ or lower $S$. The best choice of $B$ and $S$ depend on the use case, i.e. what the quality requirements are for a given application. This principle is studied as well in [25,26]. The memory requirements are related by the total number of coefficients $\frac{S^2 B^2}{N^2}$, and the number of calculations for coefficient updates is proportional to $N^2 B^2$.

| $s$ | $B$   | $4$  | $8$  | $16$  | $32$  | $64$  |
|-----|-------|------|------|------|------|------|
|     |       |      |      |      |      |      |
|     | $B$   | 0.5780 | 0.4862 | 0.4837 | 0.5177 | 0.9608 |
|     | $2B$  | 0.1470 | 0.1369 | 0.1367 | 0.1802 | 0.8105 |
|     | $4B$  | 0.0383 | 0.0353 | 0.0376 | 0.0886 | 0.7707 |
|     | $8B$  | 0.0097 | 0.0090 | 0.0123 | 0.0646 | 0.7607 |

The optimal sub-block size $K$ will be device dependent. Larger $K$ means fewer and bigger lozenge cells, but requires more memory. Generally, the goal is to have the largest $K$ possible that still fits in cache. We chose $K = 4$ for our hardware configuration; more details on the CUDA implementation can be found in [27].

4.2. Measuring calculation speed

We calculated a hologram with a wavelength of $\lambda = 633$ nm and a pixel pitch of $\rho = 2$ $\mu$m. The resolution was $16384 \times 16384$ pixels, totalling to $2.56 \cdot 10^8$ pixels. The hologram was subdivided in blocks of size $B = 32$, with dictionary block sizes of $S = 64$. The sub-stereogram sub-block size was set to $K = 4$.

The scene was composed of the Bi-plane point cloud, consisting of 1 million points with associated intensities for the point color. The plane was centered laterally to match the hologram center, and displaced to be 20 cm from the hologram plane. The dimensions of the plane along the main axes are $33.8 \times 13.2 \times 23.5$ mm.

The algorithm ran on a machine with an Intel Xeon E5-2687W v4 processor 3Ghz, 256 GB of RAM, a NVIDIA TITAN RTX GPU with a Windows 10 OS. The algorithm was implemented in C++17 with CUDA 10.1 for the calculations done on the GPU, enabling CUDA compute capability 7.5.

The conventional PAS implementation took 711.7 s, while the proposed solution took 20.6 s, resulting in a 34.5-fold speedup. This amounts to average speeds of 80.5 ms per $10^6$ points per megapixel. The distribution of the points into the lozenge cells only took about 29 ms, so it has negligible overhead. The algorithms produce identical output (up to round-off errors due to the non-associativity of floating point addition) and thus have no quality difference.
If we take a typical high-end commercially available SLM with 4K resolution (3840×2160 pixels) and a point cloud of $5 \cdot 10^4$ points, we achieve a video rate of 30 frames per second. As a comparison, [29] reports 15 frames per second for a spatial light modulator of 1920×1080 pixels for an object represented by 6500 point cloud sources on a state-of-the-art FPGA hardware implementation of a CGH computer; this amounts to 4.95s per million points per megapixel. Our proposed system is thus 62 times faster; however, an important caveat is that both the hardware and the PSF approximation algorithms are different for this system.

Four different reconstructed views for the generated hologram are shown in Fig. 6, and compared with the brute-force ray-tracing implementation using (2) for the ground truth.

**Fig. 6.** Multiple numerical reconstructions of the hologram showing all combinations of left/right views, refocusing at the front/back of the plane, computed with ray-tracing or with PAS. Note that the proposed algorithm will produce exactly the same result as standard PAS.

5. **Conclusion**

We propose a novel algorithm to significantly accelerate PAS calculations without quality reduction. The algorithm is compatible with most variants of PAS that use the FFT. The paper
brings novel insight into how the partition of space maps to FFT coefficients, which may lead to faster CGH algorithms beyond PAS. The algorithm is especially suited for customized hardware solutions such as FPGA because of its much lower memory requirements and dependencies. We sped up PAS calculations with a factor of over 30. In doing so, we hope to bring high resolution real time video holography a step closer to realization.

Funding
Fonds Wetenschappelijk Onderzoek (12ZQ220N); Seventh Framework Programme (617779).

Acknowledgments
The Biplane point cloud model is courtesy of ScanLAB Projects.

Disclosures
The authors declare no conflicts of interest.

References
1. 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).
2. J.-H. Park, “Recent progress in computer-generated holography for three-dimensional scenes,” J. Inf. Disp. 18(1), 1–12 (2017).
3. M. E. Lucente, “Interactive computation of holograms using a look-up table,” J. Electron. Imaging 2(1), 28–34 (1993).
4. 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).
5. 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).
6. P. Tsang, W.-K. Cheung, T.-C. Poon, and C. Zhou, “Holographic video at 40 frames per second for 4-million object points,” Opt. Express 19(16), 15205–15211 (2011).
7. S. Jiao, Z. Zhuang, and W. Zou, “Fast computer generated hologram calculation with a mini look-up table incorporated with radial symmetric interpolation,” Opt. Express 25(1), 112–123 (2017).
8. Y. Yamamoto, H. Nakayama, N. Takada, T. Nishitsuji, T. Sugie, T. Kakue, T. Shimobaba, and T. Ito, “Large-scale electroholography by horn-8 from a point-cloud model with 400,000 points,” Opt. Express 26(26), 34259–34265 (2018).
9. 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).
10. A. Symeonidou, D. Blinder, and P. Schelkens, “Colour computer-generated holography for point clouds utilizing the Phong illumination model,” Opt. Express 26(8), 10282–10298 (2018).
11. A. Gilles and P. Gioia, “Real-time layer-based computer-generated hologram calculation for the Fourier transform optical system,” Appl. Opt. 57(29), 8508–8517 (2018).
12. H. Kim, J. Hahn, and B. Lee, “Mathematical modeling of triangle-mesh-modeled three-dimensional surface objects for digital holography,” Appl. Opt. 47(19), D117–D127 (2008).
13. K. Matsushima and S. Nakahara, “Extremely high-definition full-parallax computer-generated hologram created by the polygon-based method,” Appl. Opt. 48(34), H54–H63 (2009).
14. K. Matsushima, M. Nakamura, and S. Nakahara, “Silhouette method for hidden surface removal in computer holography and its acceleration using the switch-back technique,” Opt. Express 22(20), 24450–24465 (2014).
15. H. Nishi and K. Matsushima, “Rendering of specular curved objects in polygon-based computer holography,” Appl. Opt. 56(13), F37–F44 (2017).
16. T. Shimobaba, T. Kakue, and T. Ito, “Acceleration of color computer-generated hologram from three-dimensional scenes with texture and depth information,” (2014), p. 91170B.
17. T. Yatagai, “Stereoscopic approach to 3-d display using computer-generated holograms,” Appl. Opt. 15(11), 2722–2729 (1976).
18. K. Wakanami, H. Yamashita, and M. Yamaguchi, “Occlusion culling for computer generated hologram based on ray-wavefront conversion,” Opt. Express 21(19), 21811–21822 (2013).
19. H. Zhang, Y. Zhao, L. Cao, and G. Jin, “Fully computed holographic stereogram based algorithm for computer-generated holograms with accurate depth cues,” Opt. Express 23(4), 3901–3913 (2015).
20. T. Shimobaba and T. Ito, “Fast generation of computer-generated holograms using wavelet shrinkage,” Opt. Express 25(1), 77–87 (2017).
21. D. Blinder and P. Schelkens, “Accelerated computer generated holography using sparse bases in the stft domain,”
Opt. Express 26(2), 1461–1473 (2018).
22. D. Blinder, “Direct calculation of computer-generated holograms in sparse bases,” Opt. Express 27(16), 23124–23137
(2019).
23. 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).
24. H. Kang, T. Fujii, T. Yamaguchi, and H. Yoshikawa, “Compensated phase-added stereogram for real-time holographic
display,” Opt. Eng 46(9), 095802 (2007).
25. H. Kang, T. Yamaguchi, H. Yoshikawa, S.-C. Kim, and E.-S. Kim, “Acceleration method of computing a compensated
phase-added stereogram on a graphic processing unit,” Appl. Opt. 47(31), 5784–5789 (2008).
26. H. Kang, E. Stoykova, and H. Yoshikawa, “Fast phase-added stereogram algorithm for generation of photorealistic 3d
content,” Appl. Opt. 55(3), A135–A143 (2016).
27. 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 (SPIE, 2020),
pp. 1–10.
28. S. A. Benton and V. M. Bove Jr, Holographic imaging (John Wiley & Sons, 2008).
29. Y. Yamamoto, N. Masuda, R. Hirayama, H. Nakayama, T. Kakue, T. Shimobaba, and T. Ito, “Special-purpose
computer for electroholography in embedded systems,” OSA Continua 2(4), 1166–1173 (2019).