Real-time computer-generated hologram calculation using pre-computed angular spectra

Antonin Gilles, Patrick Gioia

Real-time computer-generated hologram calculation using pre-computed angular spectra

Antonin Gilles[a] and Patrick Gioia[a,b]

[a]Research and Technology Institute b<>com, Cesson-Sévigné, France
[b]Orange Labs, Cesson-Sévigné, France

ABSTRACT

Due to its ability to reproduce the correct focus cues, holography is considered as a promising display technology for Augmented Reality glasses. However, since it contains a large amount of data, the calculation of a hologram is a time-demanding process, resulting in prohibiting head-motion-to-photon latency. In this paper, we propose a real-time hologram calculation method based on two modules: an offline pre-computation module and an on-the-fly hologram synthesis module. In the offline calculation module, the omnidirectional light field scattered by each scene object is individually pre-computed and stored in a Look-Up Table (LUT). Then, in the hologram synthesis module, the light waves corresponding to the viewer's position and orientation are extracted from the LUT in real-time to compute the hologram. Contrarily to previously proposed methods, our approach handles several independent scene objects and arbitrary user positions and orientations. Experimental results show that the proposed method is able to compute full-HD holograms at more than 256 frames per second, enabling its use in Augmented Reality applications.

Keywords: Real-time Hologram Calculation, Computer-Generated Holography, 3D Imaging

1. INTRODUCTION

Holography is often considered as the most promising 3D visualization technology since it creates the most natural depth illusion to the viewer. Indeed, it provides all the Human Visual System (HVS) depth cues without the need for special viewing devices and without causing eye-strain.[1] To create the depth illusion, a hologram diffracts an illuminating light beam to create the light wave that would be transmitted or reflected by a given scene.[2] As a consequence, the viewer perceives the scene as if it was physically present in front of him.

Holograms can be either optically acquired by interfering two coherent laser beams in a dark room,[3] or numerically synthesized using Computer-Generated Hologram (CGH) calculation algorithms.[4] Thanks to its attractive features in terms of 3D visualization, CGH may find application in the field of Augmented Reality (AR) glasses to solve the focus issues of conventional stereoscopic Head-Mounted Displays (HMD).[5] However, due to the large amount of data to process, real-time hologram calculation is still a very challenging topic, especially for AR applications where head-motion-to-photon latency under 20 milliseconds is required.[6]

To synthesize a hologram, most state-of-the-art methods sample the 3D scene geometry by a set of primitives and compute the hologram as the sum of light waves scattered by each primitive in the hologram plane. Commonly used primitives include points,[7] planar layers[8,9] and tilted polygons.[10] To reduce the hologram calculation time, several algorithms have been proposed, including using look-up tables,[11–13] wavefront recording planes[14,15] and redundancy removal techniques[16,17] for the point-source approach, color-space conversion,[18,19] fraction methods[20] and hybrid methods[21,22] for the layer-based approach, and using pre-computed light waves[23] or analytic formulas[24–26] for the polygon-based approach. Since they sample the scene using 3D primitives, the computational complexity of the aforementioned methods is always dependent on the number of primitives. As a consequence, computing holograms of highly detailed scenes is more time-consuming than for simple-shaped objects.

To overcome this issue, a fast calculation method based on the 3D Fourier spectrum of a voxel-based object has been proposed.[27] This method comprises two steps: an offline pre-computation step and an on-the-fly

Authors can be reached at: {antonin.gilles, patrick.gioia}@b-com.com

hologram calculation step. During the pre-computation step, the 3D spectrum of the scene is computed using a 3D Fast Fourier Transform (FFT) algorithm. Then, during the hologram calculation step, the 2D spectrum of the diffracted wavefront in a given radial direction is extracted from the 3D spectrum and inverse Fourier transformed to yield the hologram. The authors later enhanced this method with a hidden-surface removal algorithm[28] and applied it to the calculation of cylindrical holograms.[29] Since the calculation time of the second step only depends on the hologram resolution and not on the sampling of the scene, this approach enables the fast calculation of highly detailed scenes.

However, this approach still presents two limitations. First, it is limited to the calculation of holograms from scenes containing a single voxel-based object. More importantly, the hologram cannot have any arbitrary position and orientation: its optical axis must intersect the center of the scene. These two limitations are incompatible with Augmented Reality applications. In this paper, we propose a real-time calculation method based on a pre-computed Angular Spectrum per scene object. Similarly to,[27] the proposed method consists of an offline and on-the-fly calculation steps. In the offline calculation step, the Angular Spectrum of each scene object is individually pre-computed on a uniformly-sampled cube and stored in a Look-Up Table (LUT). In the hologram synthesis step, the light waves corresponding to the viewer's position and orientation are extracted from the LUT in real-time to compute the hologram. Contrarily to previously proposed methods, our approach handles several independent scene objects and arbitrary user positions and orientations.

In the rest of the paper the 3D space will be identified to $\mathbb{R}^3$ by means of an arbitrary world coordinate system. The following of this paper is organized as follows: Section 2 presents the proposed method and Section 3 gives a detailed description of the Graphics Processing Unit (GPU) implementation. Finally, experimental results are analyzed in Section 4.

2. PROPOSED METHOD

2.1 Overview

Figure 1 shows the overall block-diagram of the proposed method, which consists of two modules: an offline pre-computation module and an on-the-fly hologram synthesis module.

In the offline calculation module, the omnidirectional angular spectrum of each scene object is individually pre-computed and stored in a Look-Up Table (LUT). The omnidirectional angular spectrum corresponds to the plane wave decomposition of the light field scattered by an object in every direction. Each plane wave is represented by a three-dimensional frequency vector corresponding to its direction of propagation and by its complex amplitude. During the offline pre-computation, the self-occlusions are approximated for a few viewpoints only. In the hologram synthesis module, the light waves corresponding to the viewer's position and orientation are extracted from the LUT in real-time to compute the hologram. The calculation time of this step does not depend on the complexity or the sampling of the scene. This makes it possible to maintain a constant framerate while ensuring a good visual quality. In the following, these two modules are described.

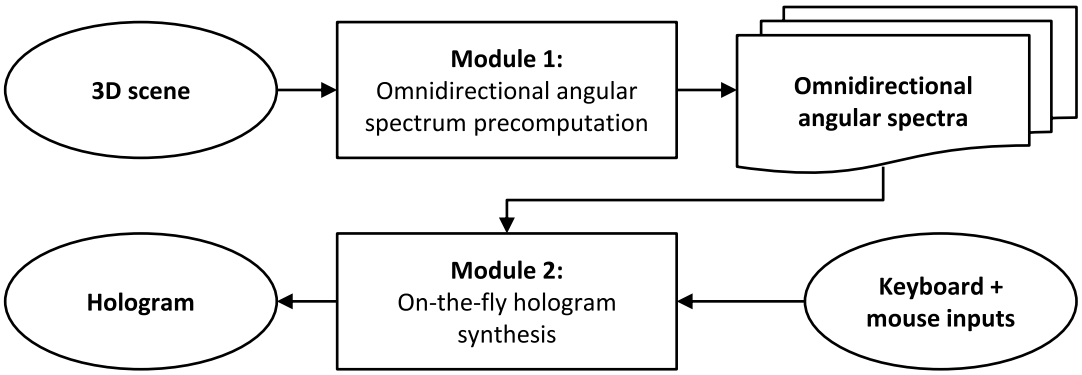


Figure 1: Overall block-diagram of the proposed method.

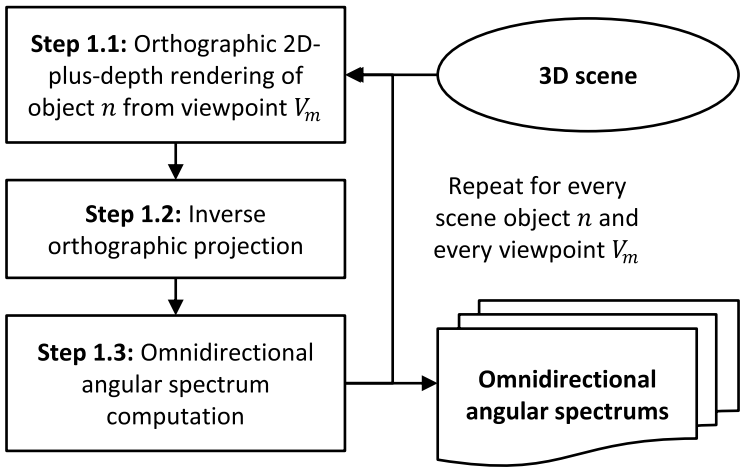


Figure 2: Block-diagram of the offline pre-computation module.

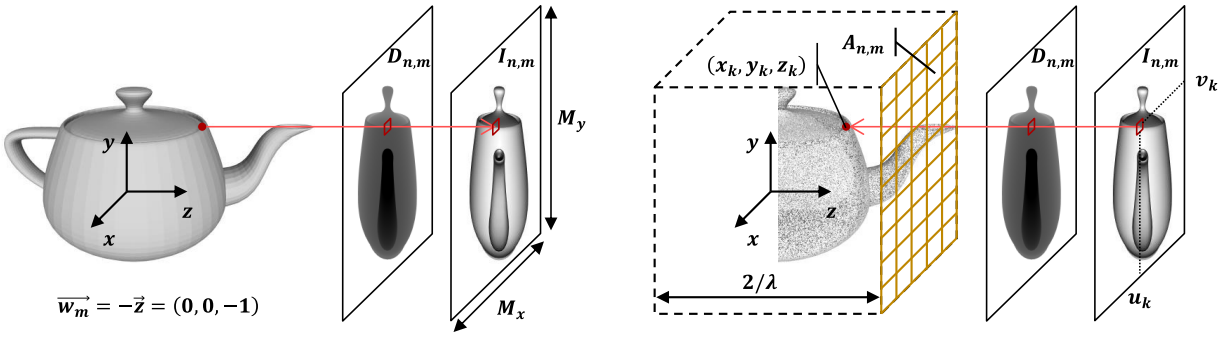
2.2 Module 1: Offline pre-computation

The block diagram of the offline pre-computation module is given in Figure 2. It consists of three steps.

First, for each object $n \in \{1, \ldots, N\}$ in the scene, a number of viewing directions $M \in \mathbb{N}$ from which the self-occlusions will be approximated is defined. M can be set depending on the geometry of each scene object: an object containing non-convex surfaces will require a larger number of viewing directions than an simple-shaped object. For each viewing direction $\overrightarrow{\mathbf{w}_m} \in \mathbb{R}^3$, with $m \in \{1, \ldots, M\}$, a 2D-plus-depth orthographic projection of object n is synthesized, as shown in Figure 3a. During this 2D-plus-depth rendering, the object self-occlusions are removed thanks to z-buffering.[30] We call $I_{n,m}$ and $D_{n,m}$ the rendered intensity and depth images, respectively.

The second step of the offline pre-computation module is to reconstruct the 3D object geometry from the 2D-plus-depth images, as shown in Figure 3b. Let $\mathcal{R}_n = (O_n; \vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}})$ be the local coordinates system of object n, whose origin $O_n = (x_n, y_n, z_n)$ is located at the center of object n and whose axes are parallel to the world coordinates axes. Since $D_{n,m}$ is encoded as an 8-bits gray level image, each pixel (u_k, v_k) can be projected back to a 3D point k of coordinates

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \Delta \left(u_k - \frac{M_x}{2} \right) \overrightarrow{\mathbf{u}_m} + \Delta \left(v_k - \frac{M_y}{2} \right) \overrightarrow{\mathbf{v}_m} + \left(\frac{d_k}{255} (z_{\max} - z_{\min}) + z_{\min} \right) \overrightarrow{\mathbf{w}_m}, \tag{1}$$



(a) (b)

Figure 3: Computing steps of the offline calculation module: (a) orthographic 2D-plus-depth rendering, (b) inverse orthographic projection and omnidirectional angular spectrum computation.

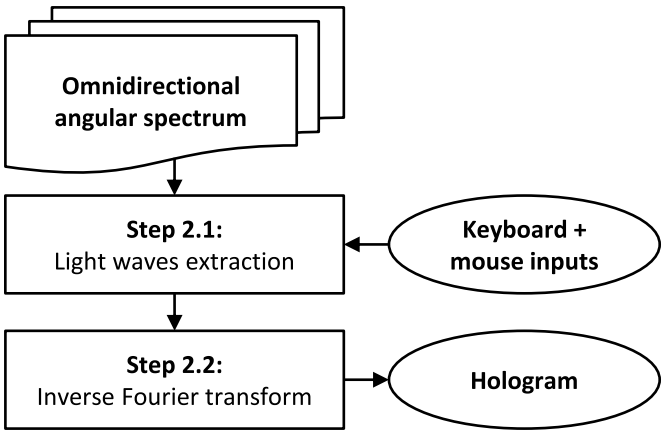


Figure 4: Block-diagram of the hologram synthesis module.

where (M_x, M_y) is the resolution of $I_{n,m}$ and $D_{n,m}$, Δ is the pixel pitch size, $z_{\min}$ and $z_{\max}$ are the minimal and maximal distances recorded in $D_{n,m}$, $d_k = D_{n,m}(u_k, v_k)$, and $\left(\overrightarrow{\mathbf{u}_m}, \overrightarrow{\mathbf{v}_m}, \overrightarrow{\mathbf{w}_m}\right)$ are the camera coordinates axes.

Finally, once the geometry of object n has been reconstructed from the 2D-plus-depth projection m, the corresponding omnidirectional angular spectrum is computed on the surface of a cube whose center matches O_n, with a side length of $\frac{2}{\lambda}$ and a sampling pitch of $\frac{1}{2S}$, where S is the object's spatial extent. The procedure is shown in Figure 3b. The omnidirectional angular spectrum is given by the sum of plane waves scattered by individual points in the point-cloud, such that

$$A_{n,m}(x,y,z) = \sum_{k=1}^{M_x M_y} \sqrt{I_{n,m}(u_k, v_k)}\exp(j\phi_k) \times \exp\left(-j\frac{2\pi}{\lambda}\left(f_x x_k + f_y y_k + f_z z_k\right)\right), \tag{2}$$

where $\phi_k \in [0, 2\pi[$ is the phase of point k, set to a random value to render a diffuse scene, and

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \frac{1}{\sqrt{x^2+y^2+z^2}} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3}$$

is the plane wave propagation vector. If $M > 1$, then for each viewing direction $\overrightarrow{\mathbf{w}_m}$ we only compute the omnidirectional angular spectrum at coordinates $\mathbf{x} = (x, y, z)$ such that $\overrightarrow{\mathbf{w}_m} \cdot \mathbf{x} > 0$. In that way, the occlusions are properly taken into account. These steps are computed for each object $n \in \{1, \ldots, N\}$, and the omnidirectional angular spectrum is accumulated for all the viewing directions $m \in \{1, \ldots, M\}$, such that

$$A_n(x,y,z) = \sum_{m=1}^{M} A_{n,m}(x,y,z). \tag{4}$$

2.3 Module 2: On-the-fly hologram synthesis

Once the omnidirectional angular spectrum of each scene object has been computed, the hologram is computed in real-time during the user navigation. Let $\mathcal{R}_h = (O_h; \overrightarrow{\mathbf{x}_h}, \overrightarrow{\mathbf{y}_h}, \overrightarrow{\mathbf{z}_h})$ be the local coordinates system of hologram H, whose origin $O_h = (x_0, y_0, z_0)$ is located at the center of the hologram and whose axes defined by

$$\overrightarrow{\mathbf{x}_h} = \begin{pmatrix} x_{h0} \\ x_{h1} \\ x_{h2} \end{pmatrix}, \quad \overrightarrow{\mathbf{y}_h} = \begin{pmatrix} y_{h0} \\ y_{h1} \\ y_{h2} \end{pmatrix}, \quad \overrightarrow{\mathbf{z}_h} = \begin{pmatrix} z_{h0} \\ z_{h1} \\ z_{h2} \end{pmatrix} \tag{5}$$

correspond to the horizontal, vertical and optical axes of the hologram, respectively. The block diagram of the hologram synthesis module is given in Figure 4. It consists of two steps.

The first step is to build the angular spectrum of the hologram $\hat{H} = \mathcal{F}\{H\}$. To this end, the light waves scattered by scene objects towards the hologram are extracted from their omnidirectional angular spectrum, such that

$$\hat{H}\left(f_{hx}, f_{hy}\right) = \sum_{n=1}^{N} A_n \left(\frac{f_x}{\lambda f_{\max}}, \frac{f_y}{\lambda f_{\max}}, \frac{f_z}{\lambda f_{\max}}\right) \exp\left(-j2\pi\left(s_x f_{hx} + s_y f_{hy} + s_z f_{hz}\right)\right), \tag{6}$$

where

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{bmatrix} x_{h0} & y_{h0} & z_{h0} \\ x_{h1} & y_{h1} & z_{h1} \\ x_{h2} & y_{h2} & z_{h2} \end{bmatrix} \begin{pmatrix} f_{hx} \\ f_{hy} \\ f_{hz} \end{pmatrix} \tag{7}$$

are the hologram frequency coordinates expressed in $\mathcal{R}_n$,

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = \begin{bmatrix} x_{h0} & x_{h1} & x_{h2} \\ y_{h0} & y_{h1} & y_{h2} \\ z_{h0} & z_{h1} & z_{h2} \end{bmatrix} \begin{pmatrix} x_i - x_0 \\ y_i - y_0 \\ z_i - z_0 \end{pmatrix} \tag{8}$$

are the coordinates of object n expressed in $\mathcal{R}_h$, and $f_{\max}$ and f_{hz} are given by

$$\begin{cases} f_{\max} = \max\left(f_x, f_y, f_z\right) \\ f_{hz} = \sqrt{\lambda^{-2} - f_{hx}^2 - f_{hy}^2} \end{cases} \tag{9}$$

Once the angular spectrum of the hologram has been computed, the last step is to compute its inverse Fourier Transform to obtain the hologram, such that

$$H(x, y) = \mathcal{F}^{-1}\left\{\hat{H}\right\}(x, y). \tag{10}$$

3. GRAPHICS PROCESSING UNIT IMPLEMENTATION

3.1 CUDA thread organization and memory model

The proposed method was implemented using the Unity game engine on a PC system employing an Intel Core i7-4930K CPU operating at 3.40 GHz, a main memory of 16 GB, three GPUs NVIDIA GeForce GTX 780Ti, and an operating system of Microsoft Windows 8. To compute the CGH patterns for the three colors simultaneously, we used one CPU thread and one GPU per color. In the implementation, all the CGH computation is done by the GPUs using the CUDA application programming interface. The CPU threads are only used to load the input 3D scene and launch CUDA kernels. Finally, to achieve best performance on the GPU, all the computations are performed using single precision.

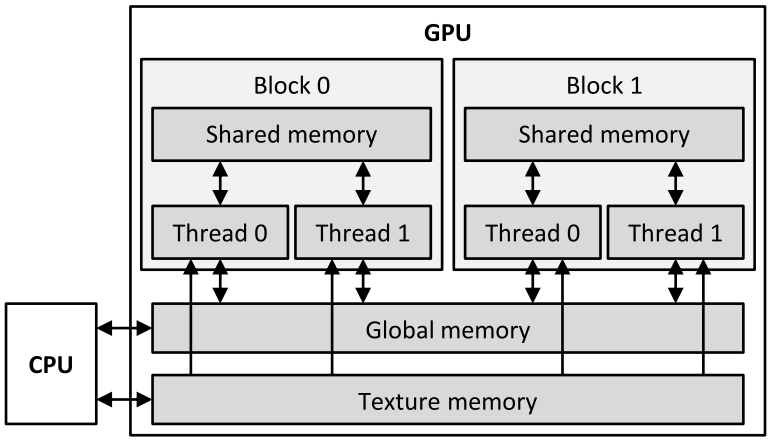


Figure 5: CUDA thread organization and memory model

Figure 5 shows the CUDA thread organization and memory model used by the GPU. The parallel code portions which are executed on the GPU are called kernels. Each kernel is composed by several threads, which are organized in blocks. Threads within a single block can exchange data through the on-chip shared memory, which has a short-latency but limited capacity. The CPU and GPU threads exchange data through the global memory, which has a large capacity but long latency and limited bandwidth. Additionally, the GPU threads can also access to a read-only texture memory, which is cached on-chip. The texture cache is optimized for 2D spatial locality, so threads that read texture addresses that are close together will achieve best performance.

3.2 Implementation as a native plugin for Unity

Figure 6 shows the overall block diagram of our implementation, which consists of two parts: the Unity engine game code, written in C# and Cg, and the native plugin code, written in C++ and CUDA.

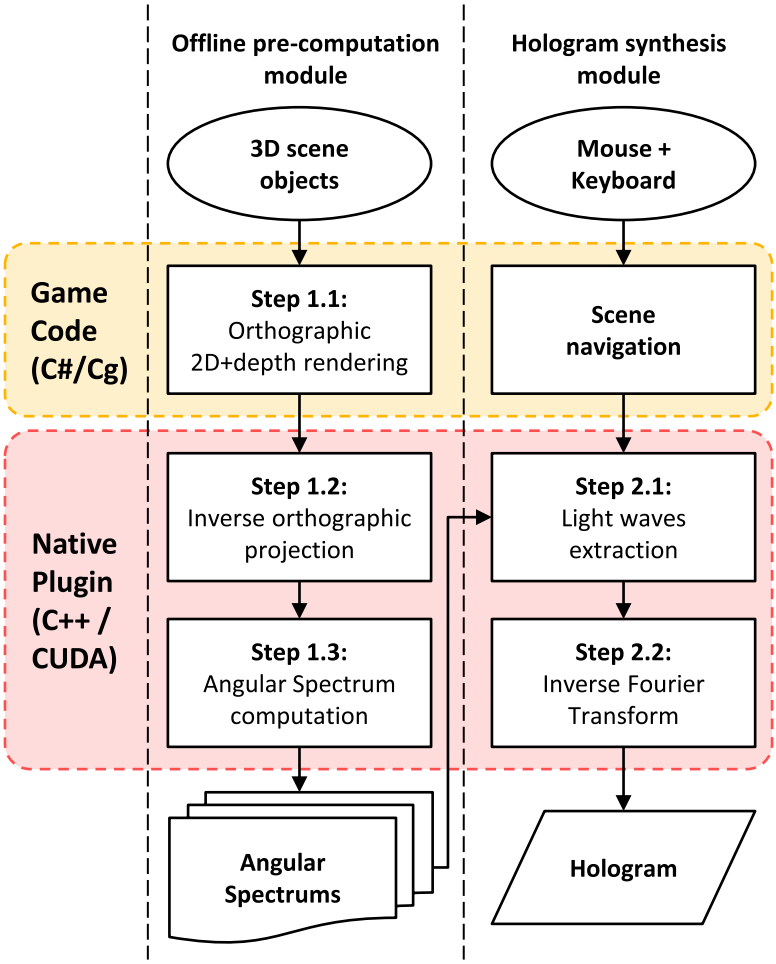


Figure 6: Block diagram of our proposed implementation.

In the offline pre-computation module and for each 3D object in the scene, the game code synthesizes orthographic 2D+depth projections from several viewpoints using a moving virtual camera with a resolution of (M_x, M_y). The rendered 2D+depth images are then sent to the native plugin code, which performs the inverse orthographic projections and Angular Spectrum computation. The resulting spectra are then stored in CUDA structures called cubemap textures, which are optimized for 2D spatial locality and texel interpolation. Cubemap textures are addressed using three texture coordinates x, y, and z that are interpreted as a direction vector emanating from the center of the cube and pointing to one face of the cube.

In the hologram synthesis module, the game code manages the user's keyboard and mouse navigation. A script, written in C#, updates the hologram position and orientation according to the keyboard and mouse inputs and calls the plugin code to compute the hologram. In the plugin code, the light waves corresponding to the hologram position and orientation are extracted from the cubemap textures in parallel, using one GPU thread per pixel. The hologram is finally obtained by computing the inverse Fourier Transform of the resulting light field using the CUDA cuFFT library by NVIDIA. This library uses the Cooley-Tukey algorithm[31] to optimize the performance of any transform size that can be factored as $2^a 3^b 5^c 7^d$, where a, b, c and d are non-negative integers.

4. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed method, we compared it with a GPU implementation of the layer-based approach described in,[8] using four sliced layers. Table 1 shows the two different hologram configurations used in the experiments. The hologram is sampled on a regular 2D grid of resolution 1920×1080 (full-HD) in the first configuration and 3840×2160 (4k2k) in the second, with a pixel pitch of 1.0μm. The wavelengths are set to 640nm, 532nm and 473nm for the Red, Green and Blue channels, respectively.

Parameter	Value
Hologram resolutions	(1920×1080) and (3840×2160)
Pixel pitch	1.0μm
Red wavelength	640nm
Green wavelength	532nm
Blue wavelength	473nm

Table 1: Hologram parameters used for the experiments

4.1 Hologram parameters and input 3D scenes

Figure 7 shows the five different test scenes used for the experiments. These scenes contain a 3D *Mushroom House* model surrounded by a set of zero to four trees: *Oak Tree*, *Poplar Tree*, *Fir Tree* and *Palm Tree*, whose omnidirectional angular spectrum memory occupations are shown in Table 2. As shown in this table, the pre-computed angular spectra require from 83 MB up to 266 MB per scene object and color. As a consequence, the total memory occupation of each 3D scene ranges from 83 MB to 1164 MB for each color, as shown in Table 3.

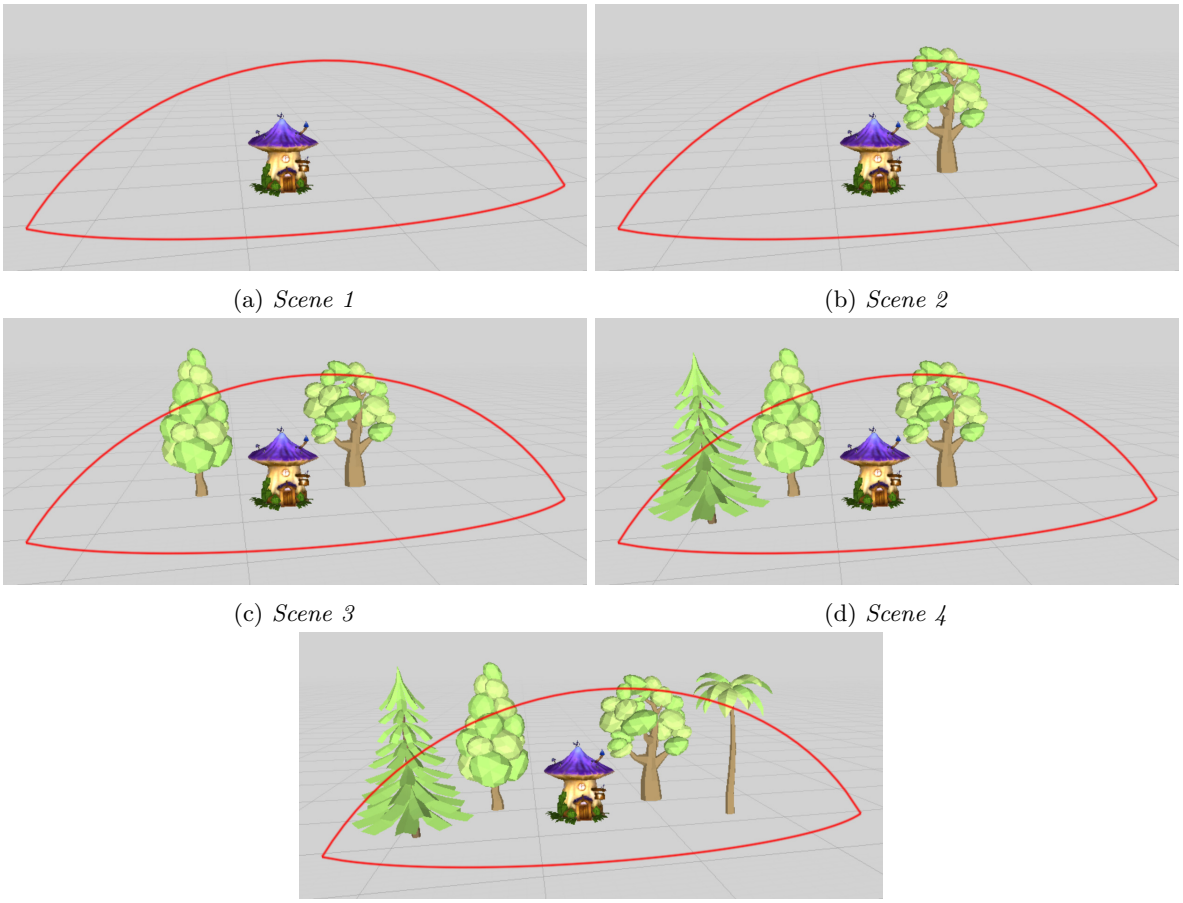
Object	Red (MB)	Green (MB)	Blue (MB)
Mushroom House	83	120	152
Oak Tree	126	182	231
Poplar Tree	151	218	277
Fir Tree	145	211	266
Palm Tree	130	188	238

Table 2: Pre-computed angular spectrum memory occupation of the different scene objects

Scene	Red (MB)	Green (MB)	Blue (MB)
Scene 1	83	120	152
Scene 2	209	302	383
Scene 3	360	520	660
Scene 4	505	731	926
Scene 5	635	919	1164

Table 3: Total memory occupation of the five different scenes

Most current consumer GPUs have more than 3 GB of global memory, so the proposed method does not require the use of a professional GPU. However, since the memory occupation depends on the spatial extent of each individual object, their size is limited by the available global memory on the GPU. In a future work, we will attempt to reduce the angular spectra memory occupation using Gabor wavelet-based compression.[32]



(a) *Scene 1* (b) *Scene 2*

(c) *Scene 3* (d) *Scene 4*

(e) *Scene 5*

Figure 7: Input 3D scenes used for the experiments. In this figures, the hologram position path is shown in red.

4.2 Analysis of the calculation time

To analyze the hologram calculation framerate, we differentiate the offline and on-the-fly computation steps. The omnidirectional angular spectrum calculation times of the different objects is shown in Table 4. As shown in this table, the pre-computation step takes about one minute for each scene object, depending on their spatial extent. This calculation time is incompatible with real-time computation. However, since this step does not depend on the hologram resolution and pixel pitch, the omnidirectional angular spectrum of each scene object can be used to compute many different holograms. This step is therefore performed only once per object and can be done offline on a CPU or GPU cluster.

Table 5 shows the calculation times and framerates of the layer-based and proposed methods for full-HD and 4k2k holograms, respectively. For a given resolution, the hologram calculation time of the layer-based method remains constant. This is due to the fact that the computational complexity of this approach depends only on

Object	**Pre-computation time**
Mushroom House	33.7 s
Oak Tree	54.1 s
Poplar Tree	62.8 s
Fir Tree	60.4 s
Palm Tree	54.8 s

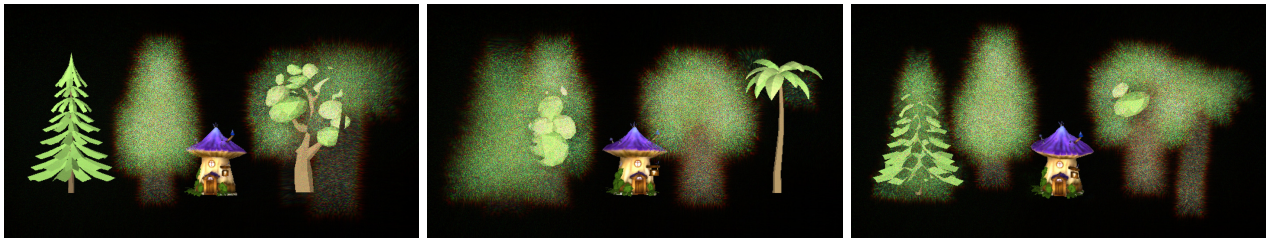
Table 4: Omnidirectional angular spectrum calculation time for the different scene objects

Scene	Full-HD holograms		4k2k holograms	
	Layer-based	*Proposed*	*Layer-based*	*Proposed*
Scene 1	65.6 ms (15.2 Hz)	2.1 ms (476 Hz)	249.8 ms (4.00 Hz)	6.8 ms (147 Hz)
Scene 2	65.2 ms (15.3 Hz)	2.5 ms (400 Hz)	249.6 ms (4.01 Hz)	8.5 ms (118 Hz)
Scene 3	65.0 ms (15.4 Hz)	3.0 ms (333 Hz)	249.4 ms (4.01 Hz)	10.3 ms (97.1 Hz)
Scene 4	65.4 ms (15.3 Hz)	3.5 ms (286 Hz)	249.6 ms (4.01 Hz)	12.1 ms (82.6 Hz)
Scene 5	65.5 ms (15.3 Hz)	3.9 ms (256 Hz)	249.7 ms (4.00 Hz)	13.9 ms (71.9 Hz)

Table 5: Calculation times and framerates for full-HD and 4k2k holograms using the layer-based and proposed methods.

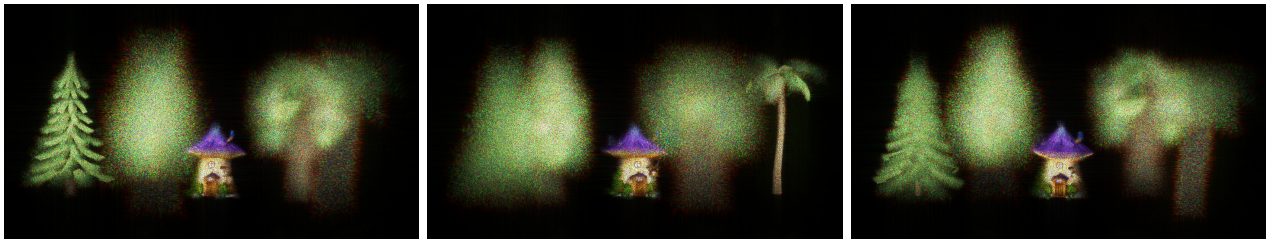
the number of sliced layers, which is set to four for every scene. Using this method, the computation takes about 65ms and 249.5ms for full-HD and 4k2k holograms of resolution, respectively. This is incompatible with AR applications, where head-motion-to-photon latency under 20 milliseconds is required.[6]

As shown in Table 5, the calculation time of the proposed method is much lower than the layer-based approach, but increases linearly with the number of scene objects. For a full-HD hologram, it takes from 2.1ms to 3.9ms, leading to framerates between 476Hz and 256Hz. For a 4k2k hologram, it varies from 6.8ms to 13.9ms, leading to framerates between 147Hz and 71.9Hz. These experimental results demonstrate that the proposed method can be used in holographic AR applications.



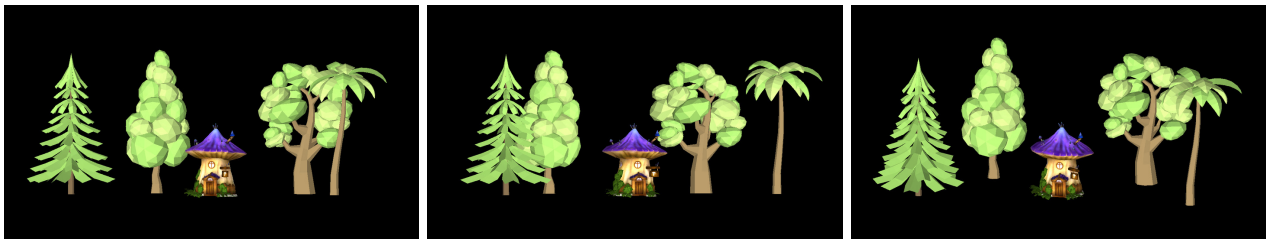
(a) Frame 26 (b) Frame 115 (c) Frame 244

Figure 8: Numerical reconstructions of 4k2k holograms computed from *Scene 5* using the layer-based method.



(a) Frame 26 (b) Frame 115 (c) Frame 244

Figure 9: Numerical reconstructions of 4k2k holograms computed from *Scene 5* using the proposed method.



(a) Frame 26 (b) Frame 115 (c) Frame 244

Figure 10: (Video1) Computer Graphics images synthesized from the same viewpoints. http://dx.doi.org/doi.number.goes.here

4.3 Numerical reconstructions

To assess the proposed method in terms of visual quality, we used a hologram moving along a semi-circle path shown in red in Figure 7, with the optical axis oriented towards the *Mushroom House* center. Figures 8 and 9 show the numerical reconstructions of 4k2k holograms computed from *Scene 5* for different viewpoints using the layer-based method and proposed method, respectively. The numerical reconstructions are focused on the *Mushroom House* door. For the sake of comparison, Computer Graphics images synthesized from the same viewpoints are shown in Figure 10.

As shown in Figure 8, the *Mushroom House* appears all in focus in the numerical reconstructions of holograms computed using the layer-based method. This is due to the fact that since we used only four sliced layers, the *Mushroom House* is entirely located within one slice. As a consequence, the 3D geometry of individual objects is completely discarded using this method, appearing as flat 2D shapes in optical reconstructions. Moreover, as shown in this figure, when objects span across two different slices, the separation between consecutive layers is clearly visible, creating strong visual artifacts in the numerical reconstructions. To overcome these issues, the number of layers should be set to at least 50. However, in that case, the calculation time increases to several seconds per frame for a 4k2k hologram, preventing this method from being used for real-time hologram synthesis.

On the contrary, as shown in Figure 9, the proposed method accurately reproduces the 3D scene geometry and colors with a continuous depth of focus. Moreover, while the optical axis of the hologram is always oriented towards the *Mushroom House* center, the trees surrounding the house are accurately reproduced as well, showing that the hologram can have any arbitrary position and orientation relative to scene objects. Nevertheless, the proposed method does not handle occlusions between independent objects, limiting the realism of the displayed images. To overcome this issue, light shielding techniques will be investigated in a future work.

5. CONCLUSION

In this paper we proposed, implemented and tested a novel approach for real-time calculation of holograms based on the omnidirectional Angular Spectrum decomposition of a 3D scene. Unlike previously proposed methods, our approach provides accurate geometry and color reproduction for several independent scene objects and arbitrary user positions and orientations. Experimental results demonstrate the computation of full-HD holograms at more than 256 frames per second and 4k2k holograms at 72 frames per second, paving the way for Augmented Reality applications.

Despite its many advantages, our method could benefit from several further investigations. First, while self-occlusions within each object are accurately reproduced, it does not take into account occlusions between independent scene objects, urging the need for light shielding techniques specifically dedicated to this approach. Another limitation lies on the fact that since the omnidirectional Angular Spectra memory occupation depends on the spatial extent of each individual object, their size is limited by the available global memory on the GPU. In a future work, we will investigate data compression techniques to reduce this memory usage.

Finally, a possible further investigation could target the optimal derivation of initial directions for the Angular Spectrum representation. This could take into account the geometry of the objects and in particular their convexity. These directions could also be denser in areas with high specularity. Altogether, an automatic shape/material adaptive derivation of the Angular Spectrum directions could lead to visually optimize the method.

ACKNOWLEDGMENTS

The *Mushroom House*, *Oak Tree*, *Poplar Tree*, *Fir Tree* and *Palm Tree* 3D models were downloaded from the Unity Asset Store (https://assetstore.unity.com).

This work has been achieved within the Research and Technology Institute b<>com, dedicated to digital technologies. It has been funded by the French government through the National Research Agency (ANR) Investment referenced ANR-A0-AIRT-07.

REFERENCES

[1] Schnars, U. and Jüptner, W., [*Digital Holography: Digital Hologram Recording, Numerical Reconstruction, and Related Techniques*], Springer Science & Business Media (Dec. 2005).

[2] Goodman, J. W., [*Introduction to Fourier Optics*], Roberts and Company Publishers, Englewood, Colo, 3rd ed. (2005).

[3] Leith, E. N. and Upatnieks, J., "Wavefront Reconstruction with Diffused Illumination and Three-Dimensional Objects," *Journal of the Optical Society of America* **54**, 1295–1301 (Nov. 1964).

[4] Lohmann, A. W. and Paris, D. P., "Binary Fraunhofer Holograms, Generated by Computer," *Applied Optics* **6**, 1739–1748 (Oct. 1967).

[5] Maimone, A., Georgiou, A., and Kollin, J. S., "Holographic Near-eye Displays for Virtual and Augmented Reality," *ACM Trans. Graph.* **36**, 85:1–85:16 (July 2017).

[6] Bailey, R. E., Iii, J. J. A., and Williams, S. P., "Latency requirements for head-worn display S/EVS applications," in [*Enhanced and Synthetic Vision 2004*], **5424**, 98–109, International Society for Optics and Photonics (Aug. 2004).

[7] Brown, B. R. and Lohmann, A. W., "Complex Spatial Filtering with Binary Masks," *Applied Optics* **5**, 967–969 (June 1966).

[8] Zhao, Y., Cao, L., Zhang, H., Kong, D., and Jin, G., "Accurate calculation of computer-generated holograms using angular-spectrum layer-oriented method," *Optics Express* **23**, 25440 (Oct. 2015).

[9] Gilles, A. and Gioia, P., "Real-time layer-based computer-generated hologram calculation for the Fourier transform optical system," *Applied Optics* **57**, 8508–8517 (Oct. 2018).

[10] Leseberg, D. and Frère, C., "Computer-generated holograms of 3-D objects composed of tilted planar segments," *Applied Optics* **27**, 3020–3024 (July 1988).

[11] Lucente, M. E., "Interactive computation of holograms using a look-up table," *Journal of Electronic Imaging* **2**, 28–34 (Jan. 1993).

[12] Kim, S.-C. and Kim, E.-S., "Effective generation of digital holograms of three-dimensional objects using a novel look-up table method," *Applied Optics* **47**, D55–D62 (July 2008).

[13] Wei, H., Gong, G., and Li, N., "Improved look-up table method of computer-generated holograms," *Applied Optics* **55**, 9255–9264 (Nov. 2016).

[14] Shimobaba, T., Masuda, N., and Ito, T., "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane," *Optics Letters* **34**, 3133–3135 (Oct. 2009).

[15] Hasegawa, N., Shimobaba, T., Kakue, T., and Ito, T., "Acceleration of hologram generation by optimizing the arrangement of wavefront recording planes," *Applied Optics* **56**, A97–A103 (Jan. 2017).

[16] Plesniak, W., "Incremental update of computer-generated holograms," *Applied Optics* **42**, 1560–1571 (June 2003).

[17] Kwon, M.-W., Kim, S.-C., and Kim, E.-S., "Three-directional motion-compensation mask-based novel look-up table on graphics processing units for video-rate generation of digital holographic videos of three-dimensional scenes," *Applied Optics* **55**, A22 (Jan. 2016).

[18] Shimobaba, T., Nagahama, Y., Kakue, T., Takada, N., Okada, N., Endo, Y., Hirayama, R., Hiyama, D., and Ito, T., "Calculation reduction method for color digital holography and computer-generated hologram using color space conversion," *Optical Engineering* **53**, 024108–024108 (Feb. 2014).

[19] Shimobaba, T., Makowski, M., Nagahama, Y., Endo, Y., Hirayama, R., Hiyama, D., Hasegawa, S., Sano, M., Kakue, T., Oikawa, M., Sugie, T., Takada, N., and Ito, T., "Color computer-generated hologram generation using the random phase-free method and color space conversion," *Applied Optics* **55**, 4159–4165 (May 2016).

[20] Chen, J.-S. and Chu, D. P., "Improved layer-based method for rapid hologram generation and real-time interactive holographic display applications," *Optics Express* **23**, 18143–18155 (July 2015).

[21] Gilles, A., Gioia, P., Cozot, R., and Luce Morin, "Fast generation of complex modulation video holograms using temporal redundancy compression and hybrid point-source/wave-field approaches," in [*Applications of Digital Image Processing XXXVIII*], **Proc. SPIE 9599**, 95990J–95990J–14 (Sept. 2015).

[22] Gilles, A., Gioia, P., Cozot, R., and Morin, L., "Hybrid approach for fast occlusion processing in computer-generated hologram calculation," *Applied Optics* **55**, 5459–5470 (July 2016).

[23] Yang, F., Kaczorowski, A., and Wilkinson, T. D., "Fast precalculated triangular mesh algorithm for 3D binary computer-generated holograms," *Applied Optics* **53**, 8261–8267 (Dec. 2014).

[24] Ahrenberg, L., Benzie, P., Magnor, M., and Watson, J., "Computer generated holograms from three dimensional meshes using an analytic light transport model," *Applied Optics* **47**, 1567–1574 (Apr. 2008).

[25] Park, J.-H., Kim, S.-B., Yeom, H.-J., Kim, H.-J., Zhang, H., Li, B., Ji, Y.-M., Kim, S.-H., and Ko, S.-B., "Continuous shading and its fast update in fully analytic triangular-mesh-based computer generated hologram," *Optics Express* **23**, 33893 (Dec. 2015).

[26] Askari, M., Kim, S.-B., Shin, K.-S., Ko, S.-B., Kim, S.-H., Park, D.-Y., Ju, Y.-G., and Park, J.-H., "Occlusion handling using angular spectrum convolution in fully analytical mesh based computer generated hologram," *Optics Express* **25**, 25867–25878 (Oct. 2017).

[27] Sando, Y., Barada, D., and Yatagai, T., "Fast calculation of computer-generated holograms based on 3-D Fourier spectrum for omnidirectional diffraction from a 3-D voxel-based object," *Optics Express* **20**, 20962–20969 (Sept. 2012).

[28] Sando, Y., Barada, D., and Yatagai, T., "Hidden surface removal of computer-generated holograms for arbitrary diffraction directions," *Applied Optics* **52**, 4871 (July 2013).

[29] Sando, Y., Barada, D., Jackin, B. J., and Yatagai, T., "Fast calculation method for computer-generated cylindrical holograms based on the three-dimensional Fourier spectrum," *Optics Letters* **38**, 5172–5175 (Dec. 2013).

[30] Straßer, W., *Schnelle Kurven- und Flächendarstellung auf grafischen Sichtgeräten*, thesis, Technischen Universität Berlin (Sept. 1974).

[31] Cooley, J. W. and Tukey, J. W., "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation* **19**(90), 297–301 (1965).

[32] El Rhammad, A., Gioia, P., Gilles, A., Cagnazzo, M., and Pesquet-Popescu, B., "Color digital hologram compression based on matching pursuit," *Applied Optics* **57**, 4930–4942 (June 2018).