Real-time embedded hologram calculation for augmented reality glasses

Antonin Gilles

Real-time embedded hologram calculation for augmented reality glasses

Antonin Gilles*
Advanced Media Content Laboratory
Research & Technology Institute b<>com
Rennes, France

Abstract

Thanks to its ability to provide accurate 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-consuming process which results in prohibiting head-motion-to-photon latency, especially when using embedded calculation hardware. In this paper, we present a real-time hologram calculation method implemented on a NVIDIA Jetson AGX Xavier embedded platform. Our method is 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. Experimental results show that the proposed method is able to compute 2K1K color holograms at more than 50 frames per second, enabling its use in augmented reality applications.

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

*Author can be reached antonin.gilles@b-com.com.

1 Introduction

Computer-Generated Holography (CGH) [1] is a technique to synthesize and reproduce the light wave scattered by a real or synthetic scene, giving the illusion to the viewers that it is physically present in front of them. By synthesizing the complete light wave, CGH provides all the Human Visual System depth cues, including focus, creating the most authentic and natural three-dimensional (3D) illusion to the naked eye. Thanks to its attractive features in terms of 3D visualization, CGH is considered as a promising display technology for Augmented Reality (AR) headsets, solving the focus issues of conventional stereoscopic Head-Mounted Displays (HMD) [2].

CGH synthesis techniques usually sample scenes by a set of 3D primitives, such as points [1], planar layers [3], or tilted polygons [4], and compute the hologram as the sum of complex light waves scattered by each of them. To enable its use in AR glasses, holograms must be computed in real-time using compact embedded calculation hardware and achieve a head-motion-to-photon latency under 20 milliseconds [5]. This is still very challenging because of the large amount of data to process.

Over the last decades, several authors have proposed acceleration methods to reduce the hologram calculation time, including look-up tables [6–8] and wavefront recording planes [9, 10] for the point-based approach, color-space conversion [11, 12] and hybrid methods [13, 14] for the layer-based approach, and

analytic formulas [15–17] for the polygon-based approach. However, only a few works proposed compact embedded hardware implementations, required for AR applications. In [18, 19], the authors built a special-purpose chip of size (28 cm × 13 cm) based on a Field Programmable Gate Unit (FPGA) to compute monochrome holograms of resolution (800×600) with a frame rate of 6 Hz for a scene containing 400 points. In [20], the authors developed a compact holographic computer using a Xilinx Zynq UltraScale+ MPSoC [21] with an ARM CPU and a FPGA on a single chip. They were able to compute full-HD monochrome holograms at 15 Hz for a scene containing 6500 points. Finally, in [22], a layer-based method was implemented on a FPGA to compute full-HD color holograms at 15 Hz.

Despite their compact size, these embedded systems are not able to compute holograms in less than 60 ms, which is insufficient for AR applications. One reason is that their computational complexity is always proportional to the number of points or layers. To overcome this issue, a real-time hologram calculation approach, in which the light waves scattered by each scene object are pre-computed offline and stored in a Look-Up Table (LUT), has been proposed [23]. In this method, during user navigation, the light waves corresponding to the viewer's position and orientation are simply extracted from the LUT in real-time to compute the hologram. Whereas in [6–8] only the spherical light waves emitted by reference points are stored in the LUT, in [23] the full light fields scattered by scene objects are precomputed, with proper texture and shading. Thanks to the pre-computation step, the hologram synthesis time only depends on its resolution and not on the sampling of the scene, enabling the fast calculation of detailed scenes. However, the suitability of this method for compact embedded hardware implementations has never been studied.

In this paper, we present the first implementation of [23] on a NVIDIA Jetson AGX Xavier embedded platform [24], and compare its performance with a layer-based method implemented on the same hardware. Thanks to implementation specifics, we demonstrate that our method is suitable for embedded hardware calculation, showing promising computation time and power consumption. This work paves the way for augmented reality applications. The following of this paper is organized as follows: Section 2 presents the proposed method, Section 3 gives a detailed description of the embedded implementation, and experimental results are analyzed in Section 4.

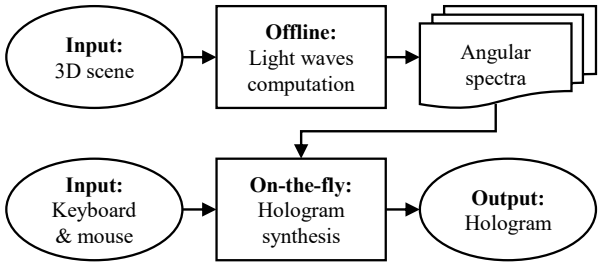


Figure 1: Overall block-diagram of the proposed method.

2 Proposed method

This section gives an overview of the method, whose detailed algorithm description can be found in [23]. Figure 1 shows the overall block-diagram of the method, consisting of two steps: an offline precomputation step and an on-the-fly hologram synthesis step, described in the following.

2.1 Offline light waves computation

During the offline calculation step, each scene object is sampled as a point-cloud. Then, its light wave is pre-computed as an angular spectrum distribution [25]. The angular spectrum corresponds to the plane wave decomposition of the light field scattered in every direction. For a given object n composed of M points, it is given by the sum of plane waves emitted by individual points, such that

$$S_{n,\lambda}(x,y,z) = \sum_{k=1}^{M} A_k \exp(j\phi_k) \times \exp\left(-j\frac{2\pi}{\lambda}\left(f_x x_k + f_y y_k + f_z z_k\right)\right). \quad (1)$$

In (1), A_k is the amplitude of point k, $\phi_k \in [0, 2\pi[$ is its phase, set to a random value to render a diffuse scene, (x_k, y_k, z_k) is its location, λ is the light wavelength, and

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

is the plane wave propagation vector.

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 object's center and whose axes are parallel to the world coordinates axes. The angular spectrum distribution of each object 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 $\frac{1}{2E}$, where E is the object's spatial extent. Finally, they are stored in a LUT to be used during hologram synthesis.

2.2 On-the-fly hologram synthesis

During the hologram synthesis step, the hologram H is computed in real-time according to the user's keyboard and mouse navigation. We call $\mathcal{R}_h = (O_h; \overrightarrow{\mathbf{x}_h}, \overrightarrow{\mathbf{y}_h}, \overrightarrow{\mathbf{z}_h})$ the local coordinates system of the hologram, whose origin O_h is located at the center of the hologram and whose axes $(\overrightarrow{\mathbf{x}_h}, \overrightarrow{\mathbf{y}_h}, \overrightarrow{\mathbf{z}_h})$ correspond to the horizontal, vertical and optical axes of the hologram, respectively.

First, the hologram angular spectrum $\hat{H}$ is computed by extracting from the LUT the light waves scattered by scene objects towards the hologram, such that

$$\hat{H}_\lambda\left(f_{hx}, f_{hy}\right) = \sum_{n=1}^{N} S_{n,\lambda}\left(\frac{f_x}{\lambda f_{\max}}, \frac{f_y}{\lambda f_{\max}}, \frac{f_z}{\lambda f_{\max}}\right) \times \exp\left(-j2\pi\left(s_x f_{hx} + s_y f_{hy} + s_z f_{hz}\right)\right), \tag{3}$$

where N is the number of scene objects, (f_x, f_y, f_z) are the hologram frequency coordinates expressed in $\mathcal{R}_n$, (s_x, s_y, s_z) are the coordinates of object n expressed in $\mathcal{R}_h$, and $f_{\max}$ and f_{hz} are given by

$$\begin{cases} f_{\max} = \max\left(f_x, f_y, f_z\right) \\ f_{hz} = \sqrt{\lambda^{-2} - f_{hx}^2 - f_{hy}^2} \end{cases} \tag{4}$$

Finally, the hologram is obtained by computing the inverse Fourier Transform of its angular spectrum, such that

$$H_\lambda(x, y) = \mathcal{F}^{-1}\left\{\hat{H}_\lambda\right\}(x, y). \tag{5}$$

3 Embedded hardware implementation

In this section, we present the embedded hardware used for the hologram calculation and discuss implementation specifics.

3.1 NVIDIA Jetson AGX Xavier embedded platform

The NVIDIA Jetson AGX Xavier embedded platform [24] has been released in 2018 as a successor to the Jetson TX2 and is aimed at developing Artificial Intelligence robotics applications. It comprises an 8-core ARM v8.2 64-bit CPU, a 512-core Volta GPU and 16 GB of DDR4 memory on a single module of size (10.5 cm × 10.5 cm × 6.5 cm).

While having 20 times less CUDA cores than most recent desktop GPUs such as the NVIDIA 3080Ti, it comprises a large memory to load neural networks. It is therefore a perfect fit for this calculation approach, which requires a potentially large LUT to be stored and loaded into memory. Furthermore, its compact size is perfectly suitable for embedded AR applications. For an easier application deployment, we used the development kit operating Ubuntu 18.04 LTS 64-bit.

3.2 Embedded implementation specifics

In our implementation, the light waves computation and hologram synthesis are performed by the embedded GPU using the CUDA application programming interface (API). The ARM processor is only used to sample and load the input 3D scene and manage the keyboard and mouse inputs during user's navigation. Finally, the hologram is displayed through the HDMI 2.0 port using the OpenGL API.

During the offline light waves computation, the CPU samples the input scene as a 3D point cloud and loads it into the GPU memory. To accelerate the computation and compensate for the reduced number of CUDA cores compared to a desktop GPU, the angular spectrum calculation is performed using 16-bit precision operations, with one GPU thread per sample. Furthermore, the complex exponential in (1) is calculated using LUTs for the real and imaginary parts cosine and sine functions, respectively. It must be noted that to avoid precision errors or to reduce the calculation time for scenes containing a large number of objects, the offline computation could also be performed using 32-bit precision on a desktop GPU and loaded onto the embedded platform storage afterwards.

The pre-computed angular spectra are then stored into 16-bit layered cubemap textures, with one layer per color channel. CUDA cubemap textures are addressed using a layer index and three texture coordinates that are interpreted as a direction vector originating from the cube center and pointing to one face of the cube. They are optimized for 2D spatial locality and texel interpolation, so threads that read texture addresses that are close together achieve best performance.

During user's navigation, the CPU updates the hologram position and orientation according to the keyboard and mouse inputs. Then, the hologram angular spectrum is computed on the GPU by extracting the light waves from the cubemap textures in parallel, using one thread per pixel. Similarly to the offline step, the calculation of the complex exponential in (3) is accelerated with 16-bit precision operations and LUTs.

Lastly, the inverse Fourier Transform in (5) is computed using the CUDA cuFFT library by NVIDIA, which uses the Cooley-Tukey algorithm [26] 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. Since sizes are restricted to power of two only when using 16-bit precision transforms, the hologram angular spectrum samples are first converted to 32-bit precision values when using arbitrary resolutions. Finally, to optimize performance, the three color channels are computed simultaneously using batched transforms.

(a) *Woods* (b) *Piano*

Figure 2: Input 3D scenes used for the experiments.

4 Experimental results

4.1 Input 3D scenes and hologram parameters

For the experiments, we used two 3D scenes shown in Figure 2, both containing five objects: *Woods*, composed by a mushroom house model surrounded by four trees, and *Piano*, comprising a grand piano and its bench as well as a table with two chairs. The angular spectrum memory occupation per channel ranges from 319 MB up to 583 MB for *Woods*, and from 253 MB to 463 MB for *Piano*. In total, the entire *Woods* and *Piano* scenes occupy 1362 MB and 1082 MB, respectively, which is less than 9% of the Jetson AGX Xavier available memory. It must be noted that the memory occupation is divided by two compared to [23] thanks to the use of 16-bit precision data.

To assess the proposed method in terms of hologram calculation times and visual quality of the numerical reconstructions, we compared it with a layer-based method [27] implemented on the same embedded platform using four sliced layers and 16-bit precision operations. In our experiments, we used a hologram resolution of 2048×1024 with a pixel pitch of 1.0 μm for *Woods* and 2.0 μm for *Piano*. The wavelengths were set to 640 nm, 532 nm and 473 nm for the Red, Green and Blue channels, respectively.

Table 1: Hologram synthesis time using the layer-based and proposed methods, depending on the number of objects N.

N	Layer-based (16-bit)	Proposed (32-bit)	Proposed (16-bit)
1	123.3 ms (8.1 Hz)	8.6 ms (116.4 Hz)	7.2 ms (138.7 Hz)
2	123.4 ms (8.1 Hz)	11.6 ms (86.1 Hz)	10.5 ms (95.2 Hz)
3	123.4 ms (8.1 Hz)	14.7 ms (68.2 Hz)	13.7 ms (73.2 Hz)
4	123.3 ms (8.1 Hz)	17.6 ms (56.7 Hz)	16.7 ms (60.0 Hz)
5	123.3 ms (8.1 Hz)	20.6 ms (48.5 Hz)	19.9 ms (50.3 Hz)

4.2 Calculation time and power consumption

Table 1 shows the average calculation times and frame rates of the layer-based and proposed methods, depending on the number of scene objects N. We differentiate the calculation time of our proposed method using 32-bit and 16-bit precision operations. First, we notice that the calculation time of the layer-based method does not depend on the number of scene objects. Indeed, the computational complexity of this approach is only proportional to the number of sliced layers. Using this method, the computation takes about 123.3 ms. This is incompatible with AR applications, where head-motion-to-photon latency under 20 ms is required [5].

On the other hand, the frame rate of our proposed method is inversely proportional to the number of scene objects. However, this method is much faster than the layer-based approach: using 32-bit precision operations, the calculation time ranges from 80.6 ms to 20.6 ms, corresponding to frame rates between 116.4 Hz and 48.5 Hz. Moreover, using 16-bit precision operations, the frame rate is further increased, ranging from 138.7 Hz to 50.3 Hz. The acceleration is therefore more significant when using fewer scene objects: the calculation time is reduced by more than 16% for one object, whereas it is only reduced by 4% when using five objects. This is due to the fact that the Fourier Transform in (5) benefits more from using 16-bit precision operations than complex multiplications in (3). Since the Fourier Transform is only computed once, the benefit is proportionally higher for scenes containing less objects.

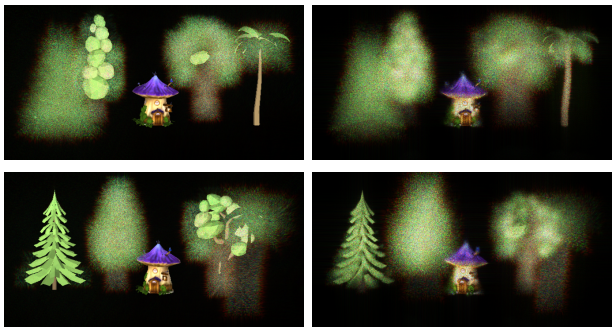
To measure the power consumption of our method, we used the `jtop` command of the jetson-stats monitoring package for NVIDIA Jetson embedded platforms [24]. This tool reported a total CPU and GPU power consumption of 27.5 W when computing holograms at the maximum achievable frame rate, which is substantially higher than commercially available stereoscopic AR headsets such as Hololens 2 [28]. Nevertheless, limiting the maximum frame rate to 50 Hz reduces the power consumption of the 16-bit version of our method to 6.3 W, 11.4 W, 18.3 W and 22.5 W for 1, 2, 3 and 4 objects in the scene, respectively.

Overall, these experimental results demonstrate that the proposed embedded system and method can be used in holographic AR applications. Nevertheless, to further reduce the calculation time and power consumption for binocular AR glasses, in a future work we plan to take into account temporal and inter-pupil hologram redundancies.

4.3 Numerical reconstructions

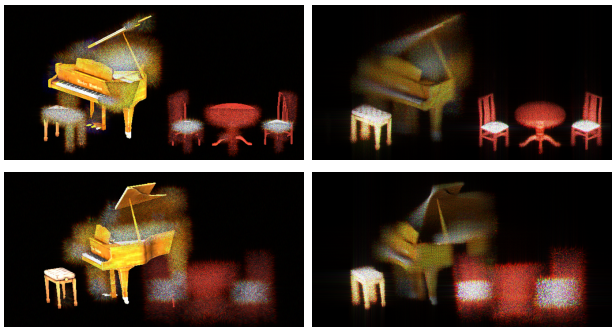
Figures 3 and 4 show the numerical reconstructions of holograms computed from *Woods* and *Piano* for two different viewpoints using the layer-based and proposed methods with 16-bit precision operations. The numerical reconstructions are focused on the mushroom house door in Figure 3 and on the piano bench in Figure 4.

As shown in Figure 3, while the mushroom house is partially blurred in the numerical reconstructions of the proposed method, it appears sharp using the layer-based method. We observe the same behaviour



(a) Layer-based method (b) Proposed method

Figure 3: Numerical reconstructions of holograms computed from *Woods* using 16-bit precision operations.



(a) Layer-based method (b) Proposed method

Figure 4: Numerical reconstructions of holograms computed from *Piano* using 16-bit precision operations.

with the piano bench in the bottom line of Figure 4. This is due to the fact that using only four layers, these objects are entirely located within one slice of the scene, appearing as flat 2D shapes. Consequently, their geometry is completely discarded using the layer-based method. On the other hand, since the poplar, oak and palm trees span across two different slices in the upper line of Figure 3a, the separation between consecutive layers is clearly visible, creating strong visual artifacts. A similar effect is also visible in the piano, table and chairs in the upper line of Figure 4a. These issues can be avoided using a larger number of scene layers (above 50), at the cost of a prohibiting calculation time of several seconds per frame.

By contrast, as shown in Figures 3b and 4b, the 3D scenes geometry and colors are accurately reproduced with a continuous depth of focus using the proposed method. Moreover, even using 16-bit precision operations, it does not produce any visible artifact in the reconstructions. Nevertheless, the proposed method does not handle occlusions between individual objects. As a consequence, while the oak tree is located behind the palm tree in the bottom line of Figure 3b, it is not properly occluded, and both objects appear as being semi-transparent. To overcome this issue, we will investigate light shielding techniques in a future work.

5 Conclusion

In this paper we implemented a real-time hologram calculation method on a NVIDIA Jetson AGX Xavier embedded module. In our system, the light field scattered by each scene object is individually precomputed offline and stored in a LUT. Then, during user's navigation, 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 point-based or layer-based embedded hardware implementations, the computation time of our method does not depend on the sampling of the scene, enabling the real-time calculation of detailed scenes.

Thanks to several implementation optimizations specifically designed for this embedded hardware, experimental results demonstrate that our proposed system is able to compute 2K1K color holograms at more than 50 frames per second. This calculation frame rate, together with the compact size of the module, demonstrate the suitability of our system for AR applications. In future work, we plan to further optimize the proposed method for binocular AR glasses by taking into account temporal and inter-pupil hologram redundancies. We will also investigate light shielding techniques between individual objects.

Acknowledgment

All the 3D models used in this paper were downloaded from the Unity Asset Store (https://assetstore.unity.com).

References

[1] B. R. Brown and A. W. Lohmann. Complex Spatial Filtering with Binary Masks. *Appl. Opt.*, 5(6):967–969, June 1966.

[2] Andrew Maimone, Andreas Georgiou, and Joel S. Kollin. Holographic Near-eye Displays for Virtual and Augmented Reality. *ACM Trans. Graph.*, 36(4):85:1–85:16, July 2017.

[3] Antonin Gilles and Patrick Gioia. Real-time layer-based computer-generated hologram calculation for the Fourier transform optical system. *Appl. Opt., AO*, 57(29):8508–8517, October 2018.

[4] Detlef Leseberg and Christian Frère. Computer-generated holograms of 3-D objects composed of tilted planar segments. *Appl. Opt.*, 27(14):3020–3024, July 1988.

[5] Randall E. Bailey, Jarvis James Arthur Iii, and Steven P. Williams. Latency requirements for head-worn display S/EVS applications. In *Enhanced and Synthetic Vision 2004*, volume 5424, pages 98–109. International Society for Optics and Photonics, August 2004.

[6] Mark E. Lucente. Interactive computation of holograms using a look-up table. *J. Electron. Imaging*, 2(1):28–34, January 1993.

[7] Seung-Cheol Kim and Eun-Soo Kim. Effective generation of digital holograms of three-dimensional objects using a novel look-up table method. *Appl. Opt.*, 47(19):D55–D62, July 2008.

[8] Hui Wei, Guanghong Gong, and Ni Li. Improved look-up table method of computer-generated holograms. *Appl. Opt., AO*, 55(32):9255–9264, November 2016.

[9] Tomoyoshi Shimobaba, Nobuyuki Masuda, and Tomoyoshi Ito. Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane. *Opt. Lett.*, 34(20):3133–3135, October 2009.

[10] Naotaka Hasegawa, Tomoyoshi Shimobaba, Takashi Kakue, and Tomoyoshi Ito. Acceleration of hologram generation by optimizing the arrangement of wavefront recording planes. *Appl. Opt., AO*, 56(1):A97–A103, January 2017.

[11] Tomoyoshi Shimobaba, Yuki Nagahama, Takashi Kakue, Naoki Takada, Naohisa Okada, Yutaka Endo, Ryuji Hirayama, Daisuke Hiyama, and Tomoyoshi Ito. Calculation reduction method for color digital holography and computer-generated hologram using color space conversion. *Opt. Eng*, 53(2):024108–024108, February 2014.

[12] Tomoyoshi Shimobaba, Michał Makowski, Yuki Nagahama, Yutaka Endo, Ryuji Hirayama, Daisuke Hiyama, Satoki Hasegawa, Marie Sano, Takashi Kakue, Minoru Oikawa, Takashige Sugie, Naoki Takada, and Tomoyoshi Ito. Color computer-generated hologram generation using the random phase-free method and color space conversion. *Appl. Opt., AO*, 55(15):4159–4165, May 2016.

[13] Antonin Gilles, Patrick Gioia, Rémi Cozot, 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*, volume Proc. SPIE 9599, pages 95990J–95990J–14, September 2015.

[14] Antonin Gilles, Patrick Gioia, Rémi Cozot, and Luce Morin. Hybrid approach for fast occlusion processing in computer-generated hologram calculation. *Appl. Opt., AO*, 55(20):5459–5470, July 2016.

[15] Lukas Ahrenberg, Philip Benzie, Marcus Magnor, and John Watson. Computer generated holograms from three dimensional meshes using

an analytic light transport model. *Appl. Opt.*, 47(10):1567–1574, April 2008.

[16] Jae-Hyeung Park, Seong-Bok Kim, Han-Ju Yeom, Hee-Jae Kim, HuiJun Zhang, BoNi Li, Yeong-Min Ji, Sang-Hoo Kim, and Seok-Bum Ko. Continuous shading and its fast update in fully analytic triangular-mesh-based computer generated hologram. *Optics Express*, 23(26):33893, December 2015.

[17] Mehdi Askari, Seong-Bok Kim, Kwang-Soo Shin, Seok-Bum Ko, Sang-Hoo Kim, Dae-Youl Park, Yeon-Gyeong Ju, and Jae-Hyeung Park. Occlusion handling using angular spectrum convolution in fully analytical mesh based computer generated hologram. *Opt. Express, OE*, 25(21):25867–25878, October 2017.

[18] Tomoyoshi Ito and Tomoyoshi Shimobaba. One-unit system for electroholography by use of a special-purpose computational chip with a high-resolution liquid-crystal display toward a three-dimensional television. *Opt. Express, OE*, 12(9):1788–1793, May 2004. Publisher: Optical Society of America.

[19] Tomoyoshi Shimobaba, Atsushi Shiraki, Nobuyuki Masuda, and Tomoyoshi Ito. Electroholographic display unit for three-dimensional display by use of special-purpose computational chip for holography and reflective LCD panel. *Opt. Express, OE*, 13(11):4196–4201, May 2005. Publisher: Optical Society of America.

[20] Yota Yamamoto, Nobuyuki Masuda, Ryuji Hirayama, Hirotaka Nakayama, Takashi Kakue, Tomoyoshi Shimobaba, and Tomoyoshi Ito. Special-purpose computer for electroholography in embedded systems. *OSA Continuum, OSAC*, 2(4):1166–1173, April 2019. Publisher: Optical Society of America.

[21] Zynq UltraScale+ MPSoC.

[22] Hojung Kim, Yongkyu Kim, Hyunwook Ji, Hyunsik Park, Jungkwuen An, Hoon Song, Yun Tae Kim, Hong-Seok Lee, and Kichul Kim. A Single-Chip FPGA Holographic Video Processor. *IEEE Transactions on Industrial Electronics*, 66(3):2066–2073, March 2019. Conference Name: IEEE Transactions on Industrial Electronics.

[23] Antonin Gilles and Patrick Gioia. Real-time computer-generated hologram calculation using pre-computed angular spectra. In *Optics, Photonics and Digital Technologies for Imaging Applications VI*, volume 11353, page 1135304. International Society for Optics and Photonics, April 2020.

[24] Jetson AGX Xavier Developer Kit, July 2018.

[25] Joseph W. Goodman. *Introduction to Fourier Optics*. Roberts and Company Publishers, Englewood, Colo, 3rd edition, 2005.

[26] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19(90):297–301, 1965.

[27] Yan Zhao, Liangcai Cao, Hao Zhang, Dezhao Kong, and Guofan Jin. Accurate calculation of computer-generated holograms using angular-spectrum layer-oriented method. *Optics Express*, 23(20):25440, October 2015.

[28] HoloLens 2—Overview, Features, and Specs | Microsoft HoloLens.