International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 10 (2019), pp. 1787-1792
© International Research Publication House. http://www.irphouse.com

Synchronization Method for CGH Generation on Graphic Processing Unit(GPU)

Joongjin Kook[1*]

[1]Assistnat Professor, School of Information Security Engineering, Sangmyung University, Korea.

**Corresponding Author*
ORCID: 0000-0002-0033-388X

Abstract

The computation for the generation of the digital hologram increases exponentially with the size of the point clouds comprising a 3D model. Thus, recently, the parallel processing using CUDA or OpenCL libraries based on GPUs, not based on CPUs can reduce the time required to generate CGH. CUDA kernel for GPU-based parallel processing needs to organize threads, blocks, and grids in consideration of the number of GPU cores and its memory size, and needs to consist of applications based on multi-threads implementing threads by core. However, multi-threads-based computation has concurrency problems, so with the concurrency problems unsolved, the computational results can not be guaranteed. In this paper, we propose concurrency problems that may occur in the CGH generation process and a software architecture to solve them, and examine the trade-off relationship between consistency and latency that may occur at this time.

Keywords: Hologram, CGH, GPU, Concurrency, Synchronization

I. INTRODUCTION

Digital holography technology can be divided into technology for obtaining holographic images and technology for displaying the holographic images. There are two methods of acquiring digital holographic images, which are obtaining 3D information of a real object and obtaining images from computer graphics. However, the technique of directly obtaining holographic images from a real object has technical limitations because it applies an interference phenomenon.

Recently, due to the rapid development of the computer hardware specifications and computing technology, CGH(Computer-Generated Hologram) technology has enabled to create various hologram contents applicable to digital holographic displays. The technology for directly obtaining images from real objects is mainly holographic image acquisition techniques using Optical Scanning Holography and Phase Shifting Interferometers [1]. The most basic and essential in the digital holographic image display technology is SLM(Spatial Light Modulator), which is capable of expressing the amplitude or phase of light, and forms a three-dimensional holographic image as a basic image display device. [2].

CGH allows you to record and to reconstruct the wavelengths of light for a 3D object. However, the resolution of CGH is the resolution of wavelength-order, and SLM (Spatial Light Modulator) having large area and high resolution is required, which leads to a limitation in developing a digital holography-based 3D display system. The enormous amounts of computation required to calculate CGH is also a problem to overcome.

Due to the developments of GPU, applications have recently been implemented by utilizing GPU-based CUDA, OpenCL, etc. in order to reduce the overhead of computations to generate CGH. However, software based on CUDA or OpenCL basically consists of multiple threads for the parallel processing, which inevitably makes a reference to the same resource in the computation process.

In this paper, we diagnose the concurrency problems that can occur in the parallel processing to generate CGH based on CUDA. We examine the impacts of these problems on the computational results, and consider the synchronization techniques necessary to overcome these problems, the limitations of the synchronization, and how to improve them.

In this paper, we modularize the CGH generation process step by step for GPU-based computation for CGH generation and construct individual CUDA kernel for modules that are capable of independent computation. In addition, in order to support multi-GPUs, in an environment with multiple GPUs, we design the entire point clouds divided by the number of GPUs and distributedly allocated to each GPU, enabling computation. We compare the computational speed between on synchronization and on non-synchronization in this environment, and the impacts of non-synchronization on the computation.

II. RELATED WORK

The typical method for digital hologram generation is R-S Integral (Rayleigh-Sommerfeld Integral), and Fresnel hologram and Fourier hologram are approximate methods based on it.

Since 1992, Chiba University in Japan has developed HORN, a hologram generator based on FPGA. HORN-6, which was developed in 2008, has four high-performance FPGAs built in and performs Lookup Table-based hologram generation computations in order to improve the computational speed. However, FPGA-based hologram generation has disadvantages: its implementation process is very complicated and takes a long time to develop it [3,4].

In 2007, Nvidia released a new GPU architecture called CUDA, and SDK (Software Development Kit), and have studied many ways to accelerate computations based on GPUs.

CUDA supports the parallel processing of data based on SIMT (Single Instruction Multiple Threads). As for CUDA, a set of threads to be processed in the parallelism is composed of grids. As shown in Fig. 2, a grid consists of a number of blocks, and a block consists of threads. The core of CUDA is parallelized in processing individual thread by core.

CUDASW ++ has proposed a library to accelerate the computational speed based on CUDA even in the low-spec GPU environment [5]. In this paper, we compared the performance of the database search performance based on the Smith-Waterman algorithm by using multiple-spec GPUs, a single GPU, and multiple GPUs.

For task-based and data-based parallel computing, which was released in 2008, the research for OPCL-based hologram generation has been conducted [6].

When GPUs are used to generate CGH, the design of memory structures for the parallelization of computations is required. As shown in Fig. 1, CUDA is composed of grids, blocks, and threads hierarchically, and is parallelly processed. The proper parallelism should be performed for the optimal computation considering warp units indicating the number of threads that can be processed simultaneously.

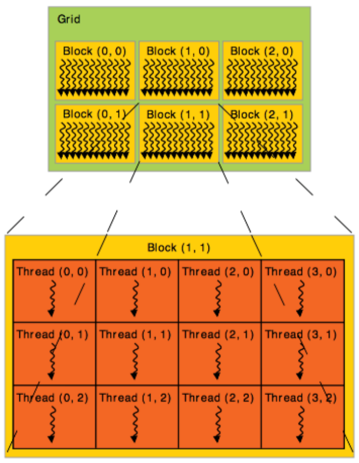

Fig 1. Structures of grids, blocks, and threads of CUDA

The formula for calculating CGH for a 3D object consisting of N points can be expressed as follows.

$$I(x_h, y_h) = \sum_i^N A_i \cos\left(\frac{2\pi}{\lambda}\left(\frac{(px_h - px_i)^2 + (py_h - py_i)^2}{2z_i}\right)\right) \quad (1)$$

In Equation (1), $I(x_h, y_h)$ is the intensity of light in CGH, and (x_h, y_h) and (x_j, y_j, z_j) represent the coordinates of CGH and 3D object. A_i represents the intensity of light for the 3D object, λ represents the wavelength of the reference light source, and in $p_i = \pi p^2 / (\lambda z_i)$, p represents a sampling interval in the CGH plane. The computational complexity of Equation (1) expressed in Big-O notation is $O(NN_xN_y)$, where N_xN_y represents the number of samples in x axis/ y axis in CGH [7].

Coherent Holographic Stereogram is one of the digital hologram generation algorithms, of which computational speed is much faster than that of R-S Integral-based algorithm. The main features for the computational acceleration are as follows.

·Divide the digital hologram plane into smaller segments.

·Reduce the amount of computations of the spatial frequency calculation process for each segment in the hologram plane from each point composing of a 3D object.

·Record the complex amplitude distribution for each point in the frequency plane of the divided hologram and process each divided hologram frequency plane as FFT.

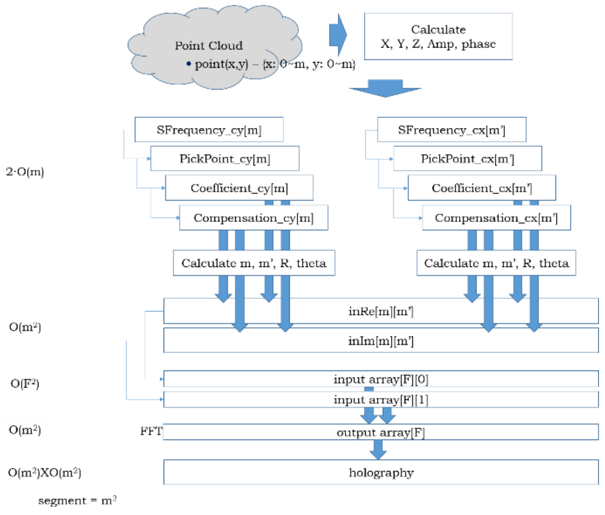

Fig 2. CGH computation process for hologram generation

CS-based hologram generation algorithms include PAS (Phase-added stereograms), CPAS (Compensated phase-added stereogram), APAS (Accurate phase-added stereogram), ACPAS (Accurate compensated phase-added stereogram), and FPAS (Fast phase-added stereogram) [8] [9].

ACPAS is an approximate algorithm that combines the characteristics of APAS and CPAS, but its accuracy and the quality of the reconstructed image are very similar to the R-S based reconstructed images. Thus, it is suitable for a holographic display device with high-speed and high-definition.

As the CS-based hologram generation algorithm has been developed in various ways, the computational speed has been improved compared to the conventional R-S method. However, the faster computational speed must be supported for the digital hologram generation that requires high-definition and real-time.

III. DIGITAL HOLOGRAM GENERATION SCHEME

III.I CGH Computation

The configuration and the computational process of the point clouds for CGH generation can be shown in Fig 3.

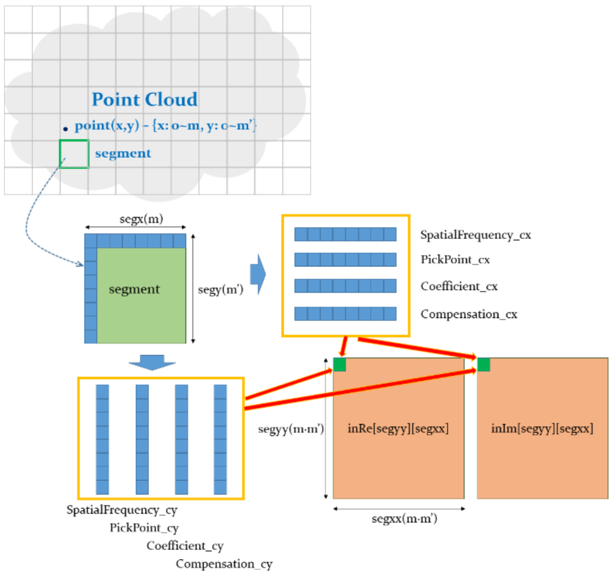

Fig 3. Point clouds and CGH computation procedure

The accumulation computations in the two-dimensional complex plane are performed repeatedly by grouping each point of the point clouds into segment, by operating the equation (1) for each point included in each segment, and by calculating the interference pattern.

In order to perform CGH computation based on CPU, iterative computations must be performed for each point of the point clouds as shown in Fig 4.

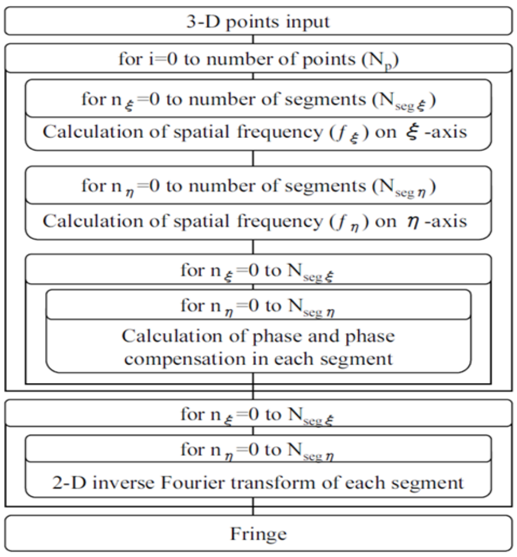
3-D points input
for i=0 to number of points (N_p)
for n_ξ=0 to number of segments ($N_{seg\,\xi}$)
Calculation of spatial frequency (f_ξ) on ξ-axis
for n_η=0 to number of segments ($N_{seg\,\eta}$)
Calculation of spatial frequency (f_η) on η-axis
for n_ξ=0 to $N_{seg\,\xi}$
for n_η=0 to $N_{seg\,\eta}$
Calculation of phase and phase compensation in each segment
for n_ξ=0 to $N_{seg\,\xi}$
for n_η=0 to $N_{seg\,\eta}$
2-D inverse Fourier transform of each segment
Fringe

Fig 4. CPU-based CGH generation procedure

On the other hand, in case of the parallelism based on GPU, as shown in Fig. 5, CUDA-based parallel processing should be performed by designating threads and blocks for each point and by forming a grid.

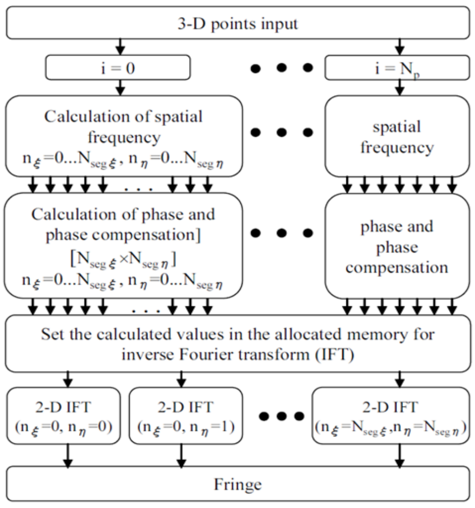
3-D points input
i = 0
i = N_p
Calculation of spatial frequency
n_ξ=0…$N_{seg\,\xi}$, n_η=0…$N_{seg\,\eta}$
spatial frequency
Calculation of phase and phase compensation]
[$N_{seg\,\xi}$×$N_{seg\,\eta}$]
n_ξ=0…$N_{seg\,\xi}$, n_η=0…$N_{seg\,\eta}$
phase and phase compensation
Set the calculated values in the allocated memory for inverse Fourier transform (IFT)
2-D IFT (n_ξ=0, n_η=0)
2-D IFT (n_ξ=0, n_η=1)
2-D IFT (n_ξ=$N_{seg\,\xi}$,n_η=$N_{seg\,\eta}$)
Fringe

Fig 5. GPU-based CGH generation procedure

The CUDA-based parallelism requires data exchange between the system memory and GPU memory. After dividing the point clouds into segments and loading it into system memory, this data must be transferred to the GPU's memory for the parallel processing. As for CUDA, the computation is performed by the kernel for the parallel processing, and the computational result is sent back to the system memory. Fig 6 shows the basic data exchange process between the system memory and GPU memory.

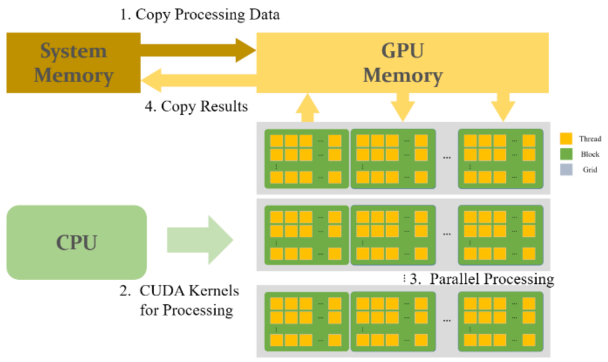

Fig 6. Data transfer and computation process between the system memory and GPU memory

III.II Parallelization of GPU Computation

To accelerate hologram generation based on multiple GPUs, conceptually you should divide the point clouds into sections as many as the number of GPUs, and make each GPU responsible for the divided sections, enabling parallel computation, as shown in Fig 7. At this time, each GPU performs the computation in its own memory individually, but a procedure for merging the computational results performed in each GPU is eventually needed.

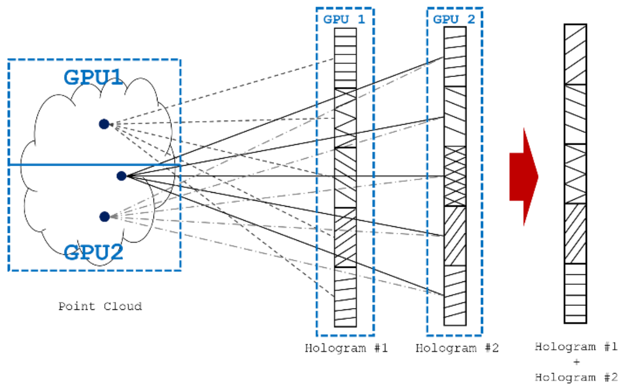

Fig 7. Partitioning computation and Merging for GPU-based CGH Generation

This parallelism requires merging individual memory allocations for each GPU and the final results of computations performed on each GPU, thereby the number of memory references is increased in proportion to the number of GPUs.

Multiple thread-based parallelism using GPUs causes a race condition among multiple threads along with an increase in the number of memory references. As shown in Fig. 8, the spatial frequency, phase and compensation computations for each point can be performed independently. But, the process of sequentially recording the real valuse/ imaginary values in the memory for the two-dimensional plane and the process of accumulating the results of FFT computation are performed in global memory, making it impossible the parallel process. Therefore, the computational results performed in each thread are accumulated in global memory as shown in Fig 8.

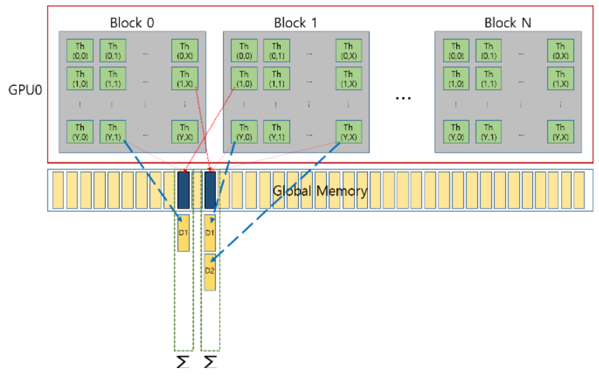

Fig 8. Global memory reference for a single GPU-based CGH computation

If simultaneous access by multiple threads is attempted while the computational results performed on the GPU is cumulative addition in global memory, the data corruption may occur and the final result may be inconsistent. Thus, additional methods such as semaphores, mutexes, or memory barriers should be used to ensure the mutual exclusion for global memory. However, these memory protection measures may result in a decrease in the computational speed.

III.III Concurrency and Synchronization

Semaphores, mutexes, atomic variables, etc. may be used to solve the concurrency problems, and different methods are applied depending on the types of concurrency problems. These methods of solving the concurrency problems commonly set up a critical section that should not be referenced at the same time, and perform lock /unlock operations before/ after this area. The lock operation is used to check whether there is a thread already entering the critical section before entering the section, and, if available, is used to obtain the right to enter the section. If a thread has already entered, the thread requesting the lock operation is blocked. The unlock operation, on the other hand, is used as a means to notify blocked threads that the lock has been released before the thread entering the critical section has left the section after the computation.

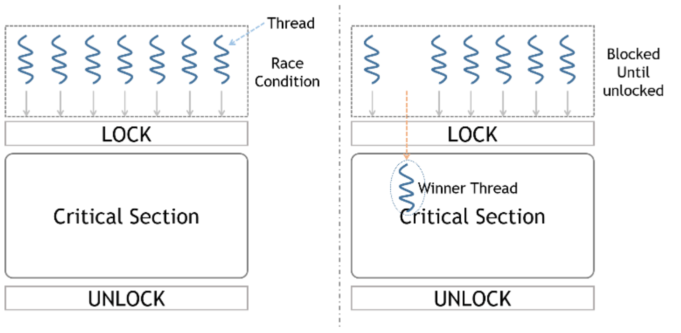

Fig 9. Global memory reference for multi-GPU based CGH computations

Fig 9 shows the race condition that threads face in entering the critical section. Threads winning the race acquire a key and enter a critical section to perform computations based on shared resources, while threads that fail to acquire a key are blocked and wait until a key is obtained again.

After all, the critical section is an exclusive access section, which reduces the performance of the parallelism. Therefore, in order to maximize the advantages of parallelism, the computations should be performed when the critical section does not exist if possible. Or, even if the critical section exists, the computations should be performed in this area only for a very short time.

In this paper, the kernel function was implemented to perform atomic operation for double precision type of mistakes in order to guarantee mutually exclusive computation and to solve the concurrency problems in CUDA environment.

V. EXPERIMENTS AND RESULTS

In this paper, in order to verify the consistency problem of computational results due to concurrency problems and the degradation of performance due to the device for synchronization in the process of generating GPU-based CGH, we compared their computational speed and their degree of corruption of the computational results among the point clouds consisting of 100, 1000, 10000, 40000, 100000, and 150000.

The configuration of the experiment environment is shown in Table 1.

Table 1. Experimental environments

H/W	Model	Quantity
CPU	Intel Core i7 6700HQ 2.6GHz	1
Memory	DDR4 8G PC4-21300	2(16GB)
GPU	Nvidia GTX960M	1

We compared the computational time for point clouds consisting of 100-160,000 on synchronization and on non-synchronization. The results are shown in Fig 10.

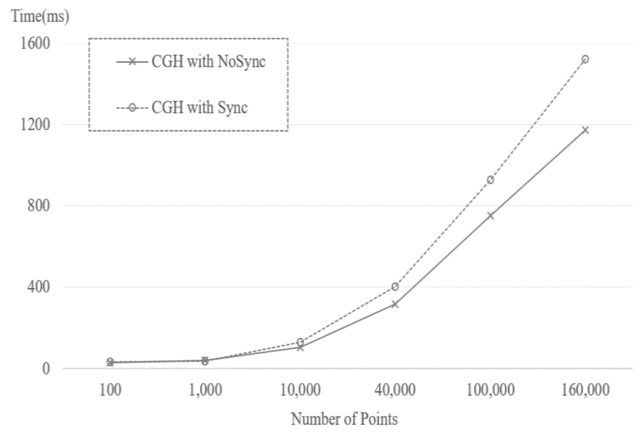

Fig 10. CGH Generation Time (ms)

On non-synchronization, the CGH generation time is shorter, but when CGH is generated for the same point cloud, the results are different each time as shown in Fig. 11.

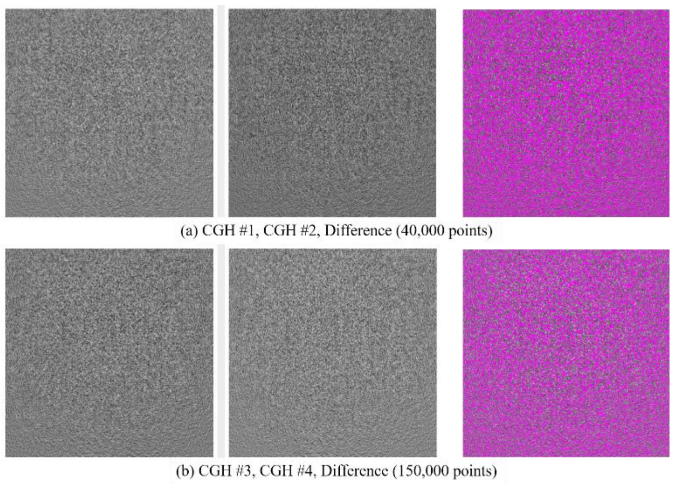
(a) CGH #1, CGH #2, Difference (40,000 points)
(b) CGH #3, CGH #4, Difference (150,000 points)

Fig 11. CGH Generation Result of Non-synchronization

The time required for synchronization increases as the number of points increases. From 10,000 points, about 20% of the CGH generation time is spent on the synchronization, as shown in Fig. 12.

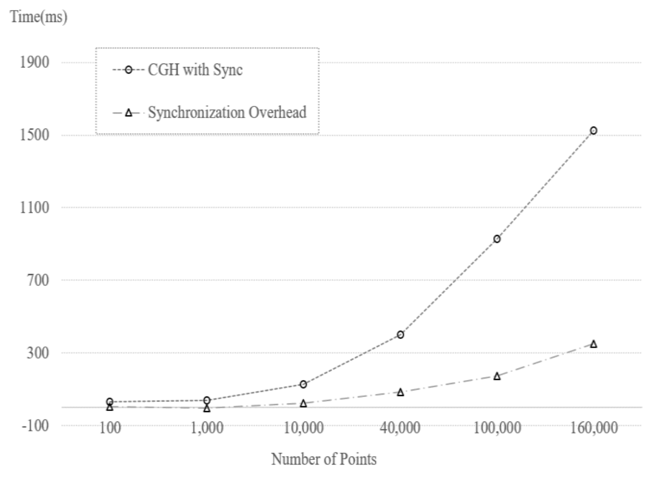

Fig 12. Synchronization Overhead for CGH Generation

VI. CONCLUSION

In this paper, we diagnosed the concurrency problem that occurs when GPU-based parallel processing is performed for CGH generation, and applied atomic operation for the purpose of the synchronization to guarantee the consistent computational results. However, we confirmed that as the size of the point cloud increases, the time required for synchronization increases by more than 20%. Therefore, there is a need for a new method to reduce the synchronization overhead in order to obtain high-quality real-time CGH.

REFERENCES

[1] Peter, W.M.T, Ting-Chung, P., 2016, “Review on the State-of-the-Art Technologies for Acquisition and Display of Digital Holograms,”IEEE Transactions on Industrial Informatics, 12(3), pp.886-901.

[2] Yongjun, L., Jinwoong, K., 2018,“Digital Holography Researches and Technologies for VR, AR, and MR,” Institute for Information & communication Technology Planning & evaluation, Weekly Technology Trend-AR, VR, MR, pp. 1-13.

[3] Tomoyoshi, S., Nobuyuki, M., Takashige, S., Satoru, H., Shinobu, T., Tomoyoshi, I., 2000,“Special-purpose computer for holography HORN-3 with PLD technology,”Computer Physics Communications, 130(1–2), pp. 75-82.

[4] Yasuyuki, I., Hirotaka, N., Tomoyoshi, I., Nobuyuki, M., Tomoyoshi, S., Atsushi, S, Takashige, S., 2009, “HORN-6 special-purpose clustered computing system for electroholography,” Optics Express, 17(16), pp. 13895-13903.

[5] Yongchao, L., Douglas, L.M., Bertil, S. “CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units,” BMC Research Notes, BioMed Central, May 2009,

[6] Tomoyoshi, S., Tomoyoshi, I., Nobuyuki, M., Yasuyuki, I., Naoki, T. 2010,“Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL,”Optics Express 9955, 18(10), pp. 9955-9960.

[7] Toyohiko, Y., 1976,“Stereoscopic approach to 3-D display using computer-generated holograms,” Applied Optics, 15, pp. 2722-2729.

[8] Masahiro, Y., Hideshi, H., Toshio, H., Nagaaki, O., 1993,“Phase-added stereogram: calculation of hologram using computer graphic technique,” Proceedings Volume 1914, Practical Holography VII: Imaging and Materials, pp. 25-33.

[9] Hoonjong, K., Elena, S., Hiroshi, Y., 2016,“Fast phase-added stereogram algorithm for generation of photorealistic 3D content,” Applied Optics, 55(3), pp. A135-A143.