Fast processing method to generate gigabyte computer generated holography for three-dimensional dynamic holographic display

Yingxi Zhang (张迎曦), Juan Liu (刘 娟), Xin Li (李 昕), and Yongtian Wang (王涌天)*

Beijing Engineering Research Center for Mixed Reality and Advanced Display, School of Optoelectronics, Beijing Institute of Technology, Beijing 100081, China

**corresponding author: wyt@bit.edu.cn*

Received November 24, 2015; accepted January 8, 2016; posted online February 26, 2016

Two different methods from graphic processing unit (GPU) and central processing unit (CPU) are proposed to suitably optimize look-up table algorithms of computer generated holography (CGH). The numerical simulations and experimental results show that we can reconstruct a good quality object. The computation of CGH for a three-dimensional (3D) dynamic holographic display can also be sped up by programming with our proposed method. It can optimize both file loading and the inline calculation process. The phase-only CGH with gigabyte data for reconstructing 10 MB object samplings is generated. In addition, the proposed method effectively reduced time costs of loading and writing offline tables on a CPU. It is believed the proposed method can provide high speed and huge data CGH for 3D dynamic holographic displays in the near future.

OCIS codes: 090.1760, 090.1995, 090.2870, 090.5694.

doi: 10.3788/COL201614.030901.

Holographic display is the most promising technology among display techniques. With the development of holography, achievements have been made in different aspects. Elimination of a zero-order beam[1,2], handling of occlusion issues[3], increasedviewing angle[4], etc. have made great contributions to three-dimensional (3D) holographic display. However, the low speed of holographic calculation is still a bottleneck for holographic display, which hinders the real-time reconstruction of 3D objects.

To solve the problem, there have been many different methods to accelerate the generation of a real-time hologram during the past few decades. Integral photography has been used in a capture and reconstruction system that can reconstruct a 3D live scene generated by fast Fourier transform (FFT) at 12 frames/s[5]. The ray-tracing approach[6] is a basic computer generated hologram (CGH) computation method. 3D objects are discretized into the point cloud, and every point of this cloud is considered a point light source. A hologram can be calculated by the sum interference fringes of each point. We can reconstruct quite high quality 3D objects in this way after a very long calculation time. Some of the calculation steps can be transferred to the offline process, which can reduce computational work inline and leads to the use of the look-up table (LUT) method[7], a data structure that stores precalculated results. Using it to trade space for time can optimize the hologram generation speed of the coherent ray trace (CRT) method. Because of the rise of this idea, some new methods have been proposed. A split look-up table (S-LUT) algorithm[8] splits the LUTs into horizontal and vertical vectors by using the Fresnel approximation, but its offline tables grow fast with increasing hologram size. Furthermore, a compressed look-up table (C-LUT)[9] splits the z modulation factor using Fraunhofer diffraction to compress the offline table. It is a fast and memory reduced method and suitable for calculating large size holograms with limited store space, but the reconstructed object from the C-LUT algorithm cannot reach expected quality.

Except for the algorithms mentioned above, there are some optimized methods that utilize the characteristics of programming languages or high performance hardware devices. In recent years, mixed programming has been proposed that accelerates the generation speed of CGH[10]. Although combining the advantages of different programming languages is a good idea, some researchers are concerned more about high performance computing hardware[11,12]. Advanced computing hardware contributes much to high speed calculation. Graphic processing units (GPUs), as the special chip for bulk data handling, have been designed properly for numerical computation. A GPU's parallel architecture makes it much more effective than a central processing unit (CPU), and a GPU gives a superb performance in the processing of floating type data[13]. The special mechanisms of the GPU can be used to compute a large-pixel-count hologram[14]. Currently, the GPU is used commonly to improve the computation speed of a data independent algorithm. As GPU computation performance improves, the size of the hologram increases. Most recently, a 9000×9000 pixel hologram can be generated using an FFT-based method[15]. These algorithms can be accelerated in this way, but the coding process is complex, and data communication speed is not ideal for realizing dynamic 3D holographic display.

In this Letter, we propose two effective ways to accelerate the computation algorithm of a CGH based on the

attributes of high performance computation hardware, GPU and CPU. We use dynamic parallelism[16–18] and file mapping techniques[19] based on the characteristics of high performance computing devices to implement two different LUT methods. The size of hologram that can be calculated has increased to 1 Gbyte with our method. We can complete the whole computation job in about 120 s, and the reconstructed 3D image quality is good enough for the human eye. Multiple GPUs are put into use as foundational hardware facilities, and one CPU is responsible for logical controlling. We use the compute unified device architecture (CUDA) as a programming platform promoted by NVIDIA. Moreover, we improve the loading speed of offline tables on the CPU by using the file mapping technique.

In order to increase the speed of the CRT method, the LUT method stores whole offline computation results. Computation devices have to load tables into computer's memory when generating a hologram inline. The S-LUT algorithm using Fresnel approximation has decreased the inline computation load without sacrificing the quality of the reconstructed image. It splits the horizontal light modulation factor

$$H(x'_p - x_j, z_j) = \exp\left\{-ik\sqrt{(x'_p - x_j)^2 + (d - z_j)^2}\right\}, \quad (1)$$

and vertical light modulation factor

$$V(y'_q - y_j, z_j) = \exp\left\{-ik\sqrt{(y'_q - y_j)^2 + (d - z_j)^2}\right\}. \quad (2)$$

Both are computed offline. The inline algorithm of S-LUT is separated into two steps:

$$H_s(x'_p) = \sum_{j=0}^{p_n} a_j H(x'_p - x_j, z_j), \quad (3)$$

$$I(x'_p, y'_q) = \mathbf{V} \times \mathbf{H_s}. \quad (4)$$

In these equations, d is the distance between the object plane and the hologram. (x'_p, y'_q) represents the point location on the hologram plane. (x, y, z) is the object point coordinate, and z_j is the coordinate of every slice of the object. V and H_S are modulation vectors that consist of vertical and horizontal light modulation factors, respectively.

S-LUT is a fast and useful algorithm to generate a hologram in a proper size. Its consumption of storage space increases quickly with increasing hologram size. The offline tables cannot even be stored in video memory at one time when we want to compute a huge size hologram. C-LUT can preferably solve this problem. Compared with S-LUT, C-LUT takes advantage of the Fraunhofer diffraction to achieve the further approximation. Based on this, it defines the z light modulation factor

$$L(z', z_{j_z}) = \exp\left\{-ik\frac{(x'^2_p + y'^2_q)}{2d}\right\}. \quad (5)$$

Because of it, there are some changes to the **V** and **H** factor

$$H(x'_p, x_j) = \exp\left\{-ik\frac{x'_p x_j}{d}\right\}, \quad (6)$$

$$V(y'_q, y_j) = \exp\left\{-ik\frac{y'_q y_j}{d}\right\}. \quad (7)$$

C-LUT can save offline storage into one slice so that it is possible to generate a much larger hologram with less memory. It also takes much less time to load the offline files. The same as S-LUT's, C-LUT's inline process has two steps:

$$H_s(x'_p) = \sum_{j=0}^{p_n} a_j H(x'_p, x_j), \quad (8)$$

$$I(x'_p, y'_q) = \sum_{j_z=0}^{n_z-1} [\mathbf{V} \times \mathbf{H_s}] \cdot L_{j_z}, \quad (9)$$

where n_z represents the number of slices on the z axis, and L is a matrix.

CUDA is a general parallel computing architecture that can implement the complex and time-consuming computation efficiently and can appropriately support the highly parallel structure of the LUT algorithm. The newest version of CUDA provides a characteristic called dynamic parallelism, which can accommodate the size of blocks and threads in terms of the computation complexity. This new property makes the nested programming much simpler than before and can accelerate the LUT method to some extent. It has been used extensively in many other fields. This technique allows GPU to make a judgment of the results of the computation directly without transferring the data back to the CPU. Obviously, dynamic parallelism can reduce the time spent on the data communication between the GPU and CPU. Based on this mechanism, the computing process is shown in Fig. 1.

In our method, we allocate the blocks and threads as described above and implement the S-LUT and C-LUT algorithms in this way. It divides the whole hologram into pieces and optionally adjusts the calculation work to the current computing resources. Dynamic parallelism reduces the complexity of the nested program and converts it into a more controllable form. We can modify the size of every sub-hologram by barely changing the code. In order to take advantage of the dynamic parallelism mechanism of CUDA, we consider the light intensity information of each object point as a 1×1 matrix and multiply it with its H vector of modulation factor. Therefore, the accumulation of the H light modulation factor in the same row as the object can be computed as a kind of matrix manipulation. With the change of accumulation, we can use the dynamic parallelism to split the H factor into many

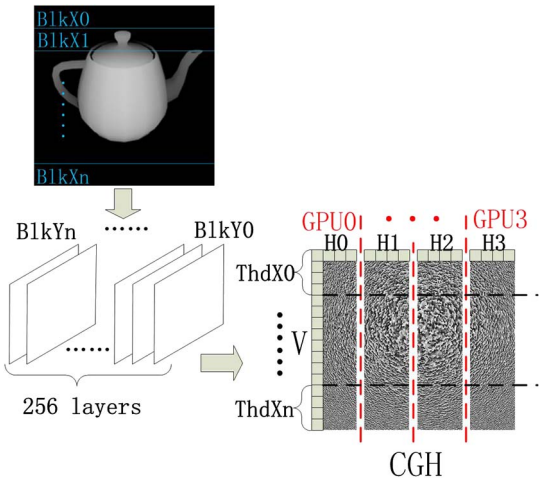

Fig. 1. Diagram of the generation of holograms by dynamic parallelism. "Blk" and "Thd" are the abbreviations of block and thread, respectively.

segments and then sum up each part together concurrently. It does finish a job to accelerate accumulation procedure. The inline process flow chart is shown in Fig. 2. File mapping and the dynamic parallelism technique are used in host and device, respectively. The device is responsible for the implementation of S-LUT and C-LUT. The complex data we process consists of two single-precision data formats.

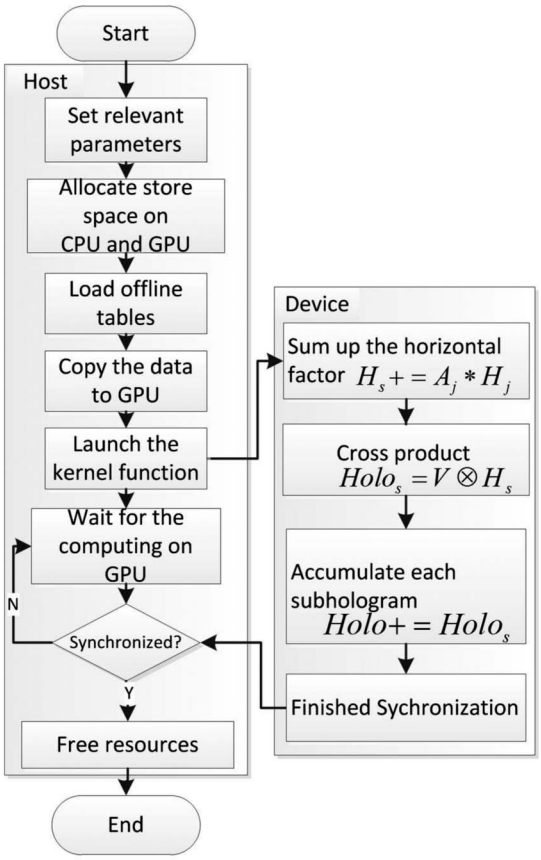

Fig. 2. Flow chart of inline computation.

For the computation of holograms with sizes larger than 16384×16384 based on the S-LUT algorithm, offline tables are so large for the limited video memory on GPU that the video memory has to be refreshed many times. This kind of manipulation consumes extra expenditures of time, which can be avoided by increasing video memory. The results are shown in Fig. 3.

In Ref. [8], the author generated a hologram accumulated by over 10000 object points at 1024×768 resolution, which took 500 ms. We can calculate a hologram with the same object points and a size of 1920×1080 in ~250 ms with the optimization of dynamic parallelism. In our experiment, the granularity level of blocks and threads also affect the speed of calculation. We modify the part of the hologram computed in a GPU into smaller sub-holograms, which means making the parallel computation have a finer level of granularity. It can improve the processing speed to some extent, but it won't be improved all the time because of thread creation and synchronization costs.

The LUT algorithm trades storage space for inline run time overhead so whole offline tables have to be loaded before starting to compute inline. It costs us huge amounts of time when we use the file stream functions to load the offline tables of a large hologram. The inputting and outputting of the file are both very time consuming. The memory mapping file technique is a memory management method under the control of an operation system. It grants an application permission to access the files in the disk through a memory pointer. In other words, this technique establishes the connection between the whole or part of the file in the hard drive and the fixed area of the virtual address space of the process. In this way, we can access a file directly and avoid both the file stream I/O operation and file buffer. It is especially efficient for loading some huge size files. We set the process of writing and reading offline table files as a graphic example in Fig. 4.

As shown in Fig. 4, first we establish a mapping between the logic memory and the hard disk before we try to read or write the light modulation factor H and V offline tables on the disk. The virtual logic address expressed by the file pointer is one-to-one mapping to the physical address now. Second, when the program wants to obtain the data of the offline table files, the operation system will invoke a page fault interrupt to call for the offline holographic data because there is no related data in logic memory. Third,

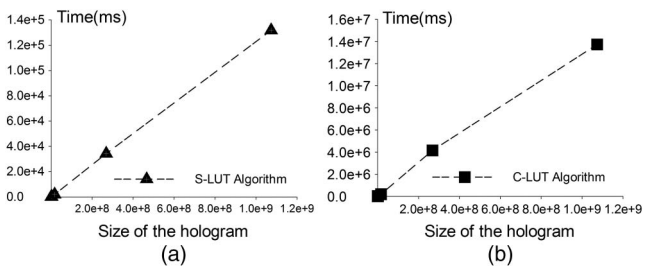

Fig. 3. Results of time consumption on (a) S-LUT and (b) C-LUT.

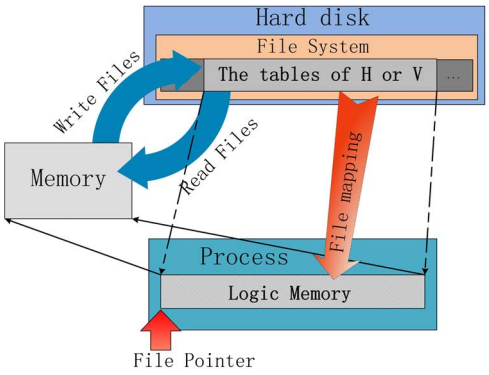

Fig. 4. Schematic diagram of file mapping.

these data are loaded into memory and then obtained by the process. This file reading and writing mechanism avoids the regular buffer copies, so we can load the data at one time rather than reading the offline table data twice from this process with other processes in this way, which means it can support multi-process programming.

Our experiments are accomplished by a computer with the specifications shown in Table 1. Our program is running on one CPU and four GPUs. Some parameters we used are listed in Table 2.

Diffraction distance is 600 mm, and pixel size is 8 μm. Based on these parameters, we compare the time costs for different offline table processing cases. The comparison of file mapping and regular file stream techniques are shown in Figs. 5 and 6.

To compare the relationship of hologram size and file loading time consumption, Figs. 5 and 6 show that the file mapping technique can reduce the time costs when we read or write data of offline tables from disk to the memory. With the increasing size of the hologram, the file mapping technique can keep relatively gentle and slow time expenditure. For different cases, the file processing time consumptions of S-LUT are ∼40–250 times more than C-LUT because of the different sizes of the processing data. The average time of writing offline holographic data for the file stream method in the S-LUT case is about 14 min, which is nearly 56 times larger than the file mapping method. For the reading case of S-LUT, the time cost of the file stream is 11 min on average. Compared with it, the average cost time of file mapping is <4 s.

The numerical and optical reconstructions of 1920 × 1080 are also completed by different algorithms. The simulations and optical experiments are shown in Fig. 7. We use the phase component loaded on the SLM to reconstruct the 3D object dynamically.

Our program is flexible for different sizes and is capable of generating huge CGH without quality loss. We can generate a 1 Gbyte hologram with more than 10^7 object points in about 120 s with our proposed method. The occlusion processing method in our program can speed up the calculation. The calculation precision and speed are in good agreement with the theoretical predictions. Furthermore,

Table 1. Specifications of PC

Item	Model	Details
CPU	Intel Xeon E5-2670	1 CPU, 8 cores 2.6 GHz
GPU	Tesla K20Xm	4 GPUs, 2 Gbyte memory each
Memory	DDR3	666.5 MHz, 192 Gbyte
OS	Windows 7	64 bit with SP1
IDE	Visual Studio 2010	32 bit
GPU API	NVIDIA	CUDA v5.5

Table 2. Parameters of CGH Computation

Item	Value
Height of object space	200 pixels
Width of object space	200 pixels
Depth of object space	256 pixels
Horizontal sampling interval of object space	80 μm
Vertical sampling interval of object space	80 μm
Wavelength of laser in reconstruction	532 nm

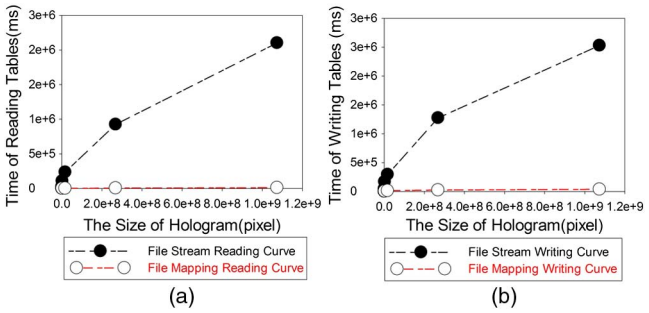

Fig. 5. Processing time comparison of (a)reading file process and (b)writing file process on S-LUT.

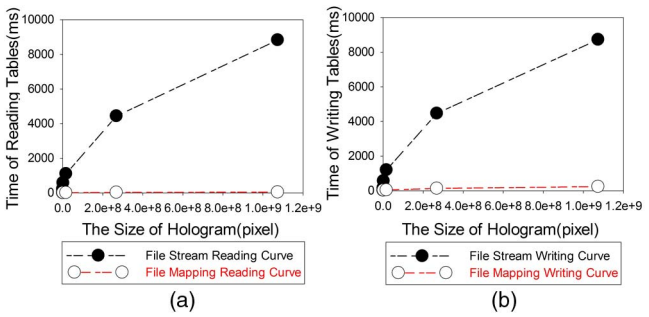

Fig. 6. Processing time comparison of (a) reading file process and (b) writing file process on C-LUT.

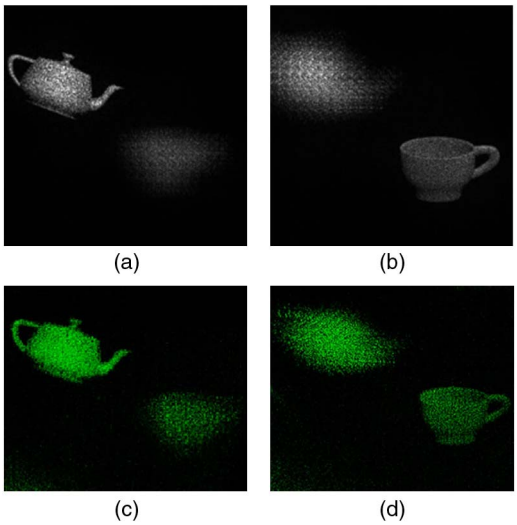

Fig. 7. Reconstructed 3D objects with different depth by S-LUT: (a) and (c) focus on the teacup, and (b) and (d) focus on the teapot, where (a) and (b) are simulated results, and (c) and (d) are recorded in optical experiments.

the dynamic parallelism technique cuts down the code complexity, which makes the program much easier to understand.

In conclusion, we propose one method to speed up the generation of CGH by GPUs and CPU based on S-LUT. Dynamic parallelism will lead to simpler programming and easier management of thread granularity. In addition, it can also reduce the inline time consumption of both LUT algorithms to some extent. The file mapping technique saves more than 100 times than file I/O stream method in average. The method can deal with huge data of up to 10^{16} total, which is a useful tool to realize the real-time 3D holographic display of the future. We can use part of the information of the huge hologram to reconstruct the object with a high quality by different encoding methods[20]. It could be applied to big data processing for various applications such as 3D data processing, cloud computation, etc.

This work was supported by the National High Technology Research and Development Program of China (No. 2015AA015905), the National Basic Research Program of China (Nos. 2013CB328801 and 2013CB328806), and the National Natural Science Founding of China (Nos. 61420106014 and 61235002).

References

1. H. Zhang, J. Xie, J. Liu, and Y. Wang, Appl. Opt. **48**, 5834 (2009).
2. H. Zhang, Q. Tan, and G. Jin, Opt. Eng. **51**, 075801 (2012).
3. H. Zhang, N. Collings, J. Chen, B. A. Crossland, D. Chu, and J. Xie, Opt. Eng. **50**, 074003 (2011).
4. J. Jia, Y. Wang, J. Liu, X. Li, and Y. Pan, Proc. SPIE **8557**, 85570B (2012).
5. Y. Ichihashi, R. Oi, T. Senoh, K. Yamamoto, and T. Kurita, Opt. Express **20**, 21645 (2012).
6. A. D. Stein, Z. Wang, and J. S. Leigh, Jr., Comput. Phys. **6**, 389 (1992).
7. M. Lucente, J. Electron. Imaging **2**, 28 (1993).
8. Y. Pan, X. Xu, S. Solanki, X. Liang, R. B. A. Tanjung, C. Tan, and T. Chong, Opt. Express **17**, 18543 (2009).
9. J. Jia, Y. Wang, J. Liu, X. Li, Y. Pan, Z. Sun, B. Zhang, Q. Zhao, and W. Jiang, Appl. Opt. **52**, 1404 (2013).
10. Y. Zhang, P. Wang, H. Chen, Y. Xu, W. Chen, and W. X. Chin, Chin. Opt. Lett. **12**, 030902 (2014).
11. J. Jia, Y. Wang, J. Liu, X. Li, and J. Xie, Laser Optoelectron. Prog. **49**, 050002 (2012).
12. T. Shimobaba, T. Ito, N. Masuda, Y. Ichihashi, and N. Takada, Opt. Express **18**, 9955 (2010).
13. L. Ahrenberg, P. Benzie, M. Magnor, and J. Watson, Opt. Express **14**, 7636 (2006).
14. Y. Pan, X. Xu, and X. Liang, Appl. Opt. **52**, 6562 (2013).
15. B. J. Jackin, H. Miyata, T. Ohkawa, K. Ootsu, T. Yokota, Y. Hayasaki, T. Yatagai, and T. Baba, Opt. Lett. **39**, 6867 (2014).
16. D. Merrill and A. Grimshaw, Parallel Process Lett. **21**, 245 (2011).
17. J. Dong, F. Wang, and B. Yuan, Lect. Notes Comput. Sci. **8206**, 409 (2013).
18. J. Wang and S. Yalamanchili, in *Proceedings of the IEEE International Symposium on Workload Characterization* (2014).
19. Z. Sun, Comput. Knowl. Technol. **9**, 4363 (2013).
20. G. Zheng, H. Muhlenbernd, M. Kenney, G. Li, T. Zentgraf, and S. Zhang, Nat. Nanotechnol. **10**, 308 (2015).