基于多图像处理单元的 Matlab 计算全息图快速算法

宁冉，李重光*，楼宇丽，桂进斌，宋庆和
昆明理工大学理学院，云南 昆明 650500

摘要 为实现全息图的准实时计算，提出了一种基于多图像处理单元(GPU)的 Matlab 计算机生成的全息图快速算法。该方法充分利用了 Matlab 编程的简易性和 GPU 的高计算性能，可有效节约全息图计算时间。模拟结果表明，所提方法的计算速度比传统计算方法提高了两个数量级。

关键词 全息；三维显示；准实时计算；图像处理单元
中图分类号 O438 **文献标识码** A **doi**：10.3788/LOP56.050901

Matlab Fast Algorithm of Computer Generated Holograms Based on Multi-Graphic Processing Unit

Ning Ran, Li Chongguang*, Lou Yuli, Gui Jinbin, Song Qinghe
College of Science, Kunming University of Science and Technology, Kunming, Yunnan 650500, *China*

Abstract In order to realize the quasi-real-time calculation of holograms, a fast algorithm of computer generated holograms based on a multi-graphic processing unit (GPU) in Matlab is proposed. This method makes full use of the simplicity of Matlab programming and the high computational performance of GPU, and thus the computing time of holograms is effectively saved. The simulation results show that the computing speed of the proposed method is about two orders of magnitude higher than those of the traditional calculation methods.

Key words holography; three-dimensional display; quasi-real-time calculation; graphic processing unit
OCIS codes 090.1760; 090.1970; 100.1160; 100.2000

1 引 言

全息三维(3D)显示技术可以提供三维场景的全部物理信息，并满足人眼观察的自然习惯，被认为是最具有发展潜力的三维显示技术。目前，全息图计算的耗时问题仍是全息三维显示技术应用的瓶颈，阻碍了该项技术的发展。

为了提高全息图的计算速度，研究人员提出了很多快速算法[1-4]。但是利用算法加速全息图的生成，存在局限性。将高性能硬件与算法相结合是当下提高全息图计算速度的有效手段，也是计算全息发展的主要趋势。目前高性能硬件加速全息图生成的方法可根据硬件分为现场可编程门阵列(FPGA)和图像处理单元(GPU)。Okada 等[5]使用可编程逻辑器件构建了专门用来加速全息图计算的硬件系统(HORN)，其最新系列 HORN-7 可以在 0.4 s 内生成含有 16000 个物点的两百万像素全息图。然而，全息专用计算机的开发周期耗时长且价格昂贵，而且需要很高的硬件设计能力，相比之下，GPU 具有高性能、价格便宜、开发周期短等优点，引起了研究人员的密切关注。1999 年，Ritter 等[6]提出了基于通用图形语言 OpenGL 对计算机图形硬件进行操作生成全息图的方法；2006 年，Ahrenber 等[7]采用 OpenGL 图形库和 OpenGL 着色器语言(GLSL)对 GPU 编程，加速全息图计算，可在 0.1 s 完成生成含有 1000 个物点分辨率为 960 pixel×600 pixel 的全息图。2010 年，Shimobaba 等[8]使用 AMD-HD5000 系列 GPU，在 OpenGL 架构下快速计算全

收稿日期：2018-08-09；**修回日期**：2018-09-06；**录用日期**：2018-09-12
基金项目：国家自然科学基金(61465005,61565011,61540075)、云南省级人培基金 (KKSY201407082)
* **E-mail**：Licg66@qq.com

息图。2012 年，Takada 等[9]利用多 GPU 集群系统实现计算机生成的全息图(CGH)，多 GPU 系统能够 0.055 s 内计算由 2048 个点组成的分辨率为 6400 pixel×3720 pixel 的三维物体的全息图。上述方法均利用低级语言对 GPU 进行编程，编程难度较大，耗时耗力，增加了全息学者的研究工作量。

在科学计算中，Matlab 是一个包含数值分析、矩阵运算、信号处理和图形显示的可视化软件[10]，不仅包含丰富的工具箱功能，而且编程简单，可以很好地解决研究中遇到的系统仿真和计算领域中的问题。为了解决 Matlab 存在计算效率较低的问题，MathWorks 利用 Parallel Computing Toolbox 提供对 NVIDIA GPU 的支持。本文提出了利用 Matlab 对多 GPU 进行编程，实现了计算全息图的并行计算，为研究人员提供了一种简单的、利用高性能 GPU 编程的方法。

2 计算全息计算原理

CGH 能够正确记录和重建三维对象的信息。在点源模型计算全息图中，三维物体被看作理想的漫散射体，将三维物体表面离散成 N 个点的集合，其中物体表面的每个点都可以视为独立的点光源，所发出的球面波可以均匀地照射在整个全息图平面上，通过计算各个点光源发出的球面波，跟踪其路径得到每一个采样点的衍射图样，如图 1 所示。

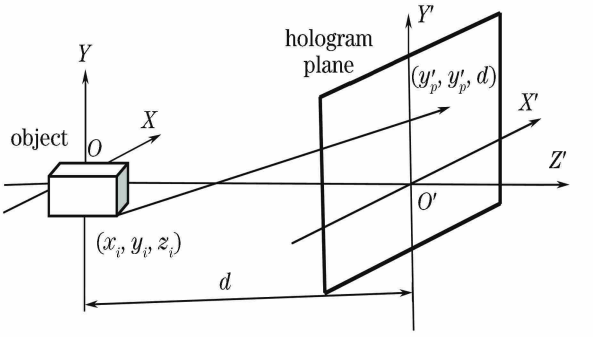

图 1 三维计算全息记录示意图

Fig. 1 Schematic of three-dimensional computer generated hologram recording

设三维物体的空间坐标为 XOY，全息图平面的空间坐标为 $X'O'Y'$，三维物体表面每个点光源坐标为(x_i, y_i, z_i)，可以用光线跟踪算法计算全息图平面上的复振幅分布：

$$U(x'_p, y'_p) = \sum_{i=1}^{N} \frac{A_i}{r_i} \exp[\mathrm{j}(kr_i + \varphi_i)], \quad (1)$$

式中 A_i 为第 i 点的光波振幅，φ_i 表示初始相位。三维物体是一个漫散射体，一般初始相位的取值为$(0, 2\pi)$之间变化的随机相位，$k = 2\pi/\lambda$ 表示波数，λ 是全息图参考光的波长，r_i 表示三维物体表面的第 i 点光源为(x_i, y_i, z_i)距全息图平面上像素点(x'_p, y'_p, d)的距离，即

$$r_i = \sqrt{(x'_p - x_i)^2 + (y'_p - y_i)^2 + z_0^2}, \quad (2)$$

式中 $z_0 = d - z_i$。在模拟计算过程中，三维物体尺寸相对于全息记录距离均较小，即$(x'_p - x_i) \ll z_0$ 和$(y'_p - y_i) \ll z_0$，因此(2)式在旁轴近似的条件下得到：

$$r_i = z_0 + \frac{(x'_p - x_i)^2}{2z_0} + \frac{(y'_p - y_i)^2}{2z_0}。 \quad (3)$$

本文采用平面波为参考光，其计算结果可用以类推参考光为球面波的情况，在 $Z = d$ 的全息平面上参考光的复振幅分布为

$$R(x', y') = A\exp[\mathrm{j}\phi(x', y')], \quad (4)$$

式中相位 $\phi(x', y') = (2\pi/\lambda)(x'\cos\theta + y'\cos\beta)$，$\theta$ 和 β 分别为参考光波矢与 X 轴和 Y 轴方向的夹角，A 为参考光的振幅。则全息面上的光强 $I(x', y')$ 分布为

$$I(x', y') = |U(x', y') + R(x', y')|^2, \quad (5)$$

在全息图的计算过程中，使用的是双极强度计算方法，双极强度计算方法是利用欧拉公式将光强分布公式进行化简，分为实数的余弦条纹和虚数的正弦条纹。在全息图计算时，该方法仅计算光强分布的余弦条纹，可提高全息图的计算速度[4]。因此在全息面上的干涉光强分布为

$$I(x', y') = C + \sum_{i=1}^{N} A_i \cos[kr_i + \varphi_i - \phi(x', y')], \quad (6)$$

式中 C 为常量，N 为构成三维物体物点数。

3 GPU 系统结构

随着计算机技术的发展，GPU 的通用计算已成为计算机学科的趋势。与中央处理器(CPU)不同，GPU 可用于计算密集型和高并行性的运算，因为它通过多个流处理器同时执行一个或多个业务。如图 2 所示，一个 GPU 具有一个全局内存、多个流式处理器(SM)、一个加载/存储单元[11]。其中，一个 SM 具有多个流处理器(SP)和共享内存，相同的流式处理器中的流处理器之间可以共享控制逻辑和指令缓存，而且 SP 可以并行执行多个线程。全局内存具有千兆字节(GB)的数据传输带宽。但是具有较长的读取延时，SM能够在4个时钟周期内访问

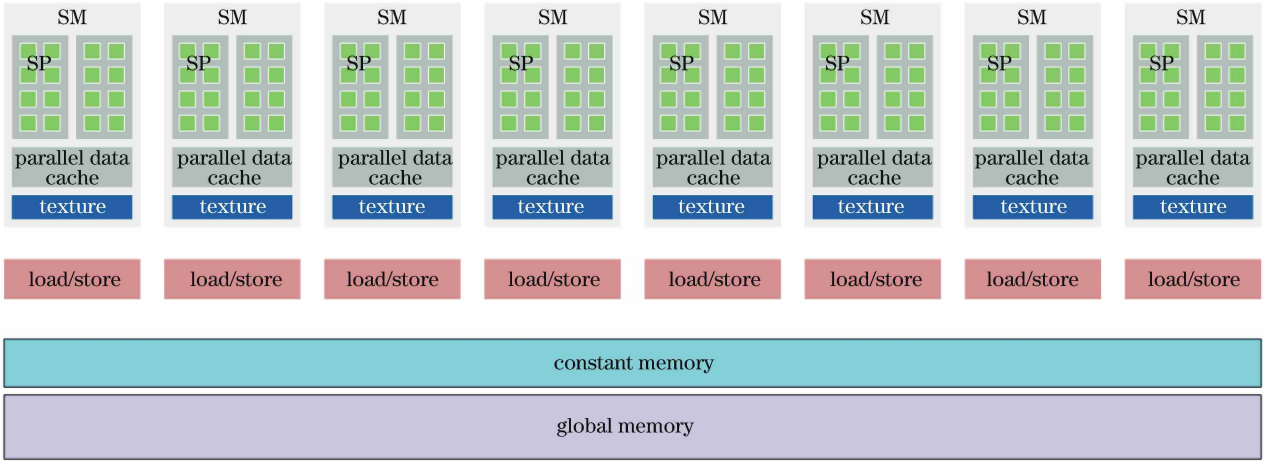

图 2 GPU 的结构

Fig. 2 Structure of GPU

共享缓存，访问全局缓存则需要 400～600 个时钟周期。为了提高 GPU 的计算速度，通常需要将数据从 GPU 的全局内存读取到每个 SM 的共享内存中再进行核函数的计算。

4 Matlab 多 GPU 并行算法

目前开发基于 Matlab GPU 的并行算法主要有三个工具箱，分别为 Jacket，GPUmat 和并行计算工具箱。本文所使用的工具箱为并行计算工具箱(PCT)，PCT 是 Matlab 公司自主研发的工具箱，其限制 NVIDA CUDA 支持的计算能力在 1.3 以上才可以使用，而且需要安装 R2010b 以上的 Matlab 版本。

4.1 并行算法

在 Matlab 中利用 PCT 调用 GPU 进行并行计算有两种方法。

1) 在 GPU 上执行重载的 Matlab 函数

在 GPU 上执行重载的 Matlab 函数，是较为简单的 Matlab 编程方式，编程者可以通过两种方法实现 Matlab 工作区间和 GPU 内存之间的数据传输。一种是由编程者定义数据传输的方式，另一种是由编程者直接使用 gpuArray()函数在 GPU 内存中创建数据所需的内存。后者可以有效减少 CPU 与 GPU 之间数据传输的运行时间。随着 GPU 并行计算的快速发展，在 Matlab 中，已经重载超过一百多个的 Matlab 函数支持 GPU 运算。例如快速傅里叶变换、快速傅里叶逆变换、矩阵的四则运算等。同时，编程者在调用函数进行运算时，可以自主选择在 GPU 内存上读取数据，或在 Matlab 工作区和 GPU 内存中的数据混合，从而使 Matlab GPU 编程的更具简易性和灵活性。

2) 自定义 GPU 运算的核函数

利用在 Matlab 中定义 GPU 核函数的方法，编程者可以根据自己的编程需求来定义 Matlab 函数，然后执行 arrayfun()函数，即在 GPU 上对数据进行并行计算的操作。而且，输入的数据可以是 CPU 内存中的，也可以是 GPU 内存中的。利用这种编程模式进行 GPU 运算，编程者可以根据需求自主编写在 GPU 设备端上运行的 Matlab 函数，大大提高了编程效率和计算性能。为了进一步优化计算性能，编程者需要控制 GPU 对 Matlab 上定义函数的调用和 CPU 与 GPU 之间的数据传输。

4.2 多 GPU 的调用

在 Matlab 上进行多 GPU 的调用，首先可以在命令窗口输入命令“gpuDeviceCount;”得到计算机中的 GPU 个数，然后输入命令“ gpuDevice(n);”来选定所要执行的第 n 个 GPU 设备。选定 GPU 设备后，接下来的 GPU 计算将会调用第 n 个 GPU。假如不调用命令“ gpuDevice(n);”，Matlab 会默认在第一个 GPU 设备上进行 GPU 计算。由于 Matlab 编程命令具有逐行执行的特点，在上次命令结束之前，下一条命令将不会执行。这是利用 Matlab 进行多 GPU 并行计算的重要问题。为了解决这个问题，方法一是启动多个 Matlab 客户端，然后利用命令“ gpuDevice(n);”逐个选取所需执行的 GPU 设备。由于该方法不便于编程和管理，选用另一方法，即启动 Matlab 的并行计算池，在命令窗口输入“parpool”命令，在 Matlab 客户端左下角的并

行计算图标将会闪烁，并显示 worker 的数量，然后在所需多 GPU 并行运算的程序中加入并行循环语句“parfor”命令，即可实现多 GPU 的并行计算。

例如，利用两个 GPU 设备进行并行计算，其编程方式如下所示。

```
N=gpuDeviceCount;          %计算机中 GPU 的个数
Parfor n_gpuDevice=1:N
gpuDevice(n_gpuDevice);    %选定在第 n_gpuDevice 设备上运行 GPU 运算
if n_gpuDevice ==1         %在第一个设备上运行
codes_1;                   %需要 GPU 并行计算的内容
elseif n_gpuDevice ==2     %在第二个设备上运行
codes_2;                   %需要 GPU 并行计算的内容
end
end
```

4.3 多 GPU 全息图的计算

基于点的计算全息图方法计算量庞大，利用多个 GPU 的高性能计算系统可快速完成 CGH 计算。本文计算全息图的尺寸为 1024 pixel×1024 pixel，GPU 个数为 2。为了充分利用多 GPU 的高计算性能，将原始的全息图划分成大小为 1024 pixel×512 pixel的两部分，每个 GPU 计算相应的全息图部分，如图 3 所示。

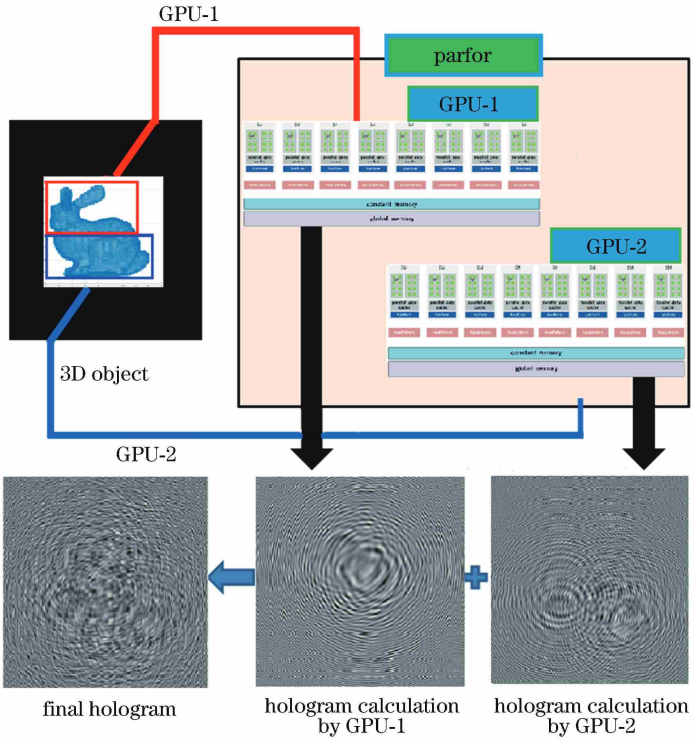

图 3 多 GPU 计算全息图

Fig. 3 Multi-GPU computer generated holograms

编程中，多 GPU 计算全息图的流程如图 4 所示。

5 实验验证

5.1 模拟实验

本文实验采用的硬件平台：处理器为 Intel Core i5-3470，运行内存为 8 GB，实验显卡为两块，型号是 NVIDIA GeForce GT 610，计算能力为 2.1，显存为 1 GB，见表 1 和 2，表 1 中 OS 表示操作系统。

模拟实验中利用 CGH 公式生成计算全息图，其结果如图 5 所示，其中 3D 对象为由 48632 个点构成的兔子。计算全息图的分辨率为 512 pixel×512 pixel 和 1024 pixel×1024 pixel，参考光波长为 $\lambda=0.000532$ mm，重建距离为 100 mm，像素间距离为 0.0061 mm。

表 3 给出了不同物体在 CPU、GPU、Mutil-core GPU 系统中生成计算全息图时间的对比。对比基于 CPU 和 GPU 的全息图计算时间数据可知，在分辨率为 512 pixel×512 pixel 时，基于 GPU 的速度约为基于CPU的50倍；在分辨率为1024pixel×

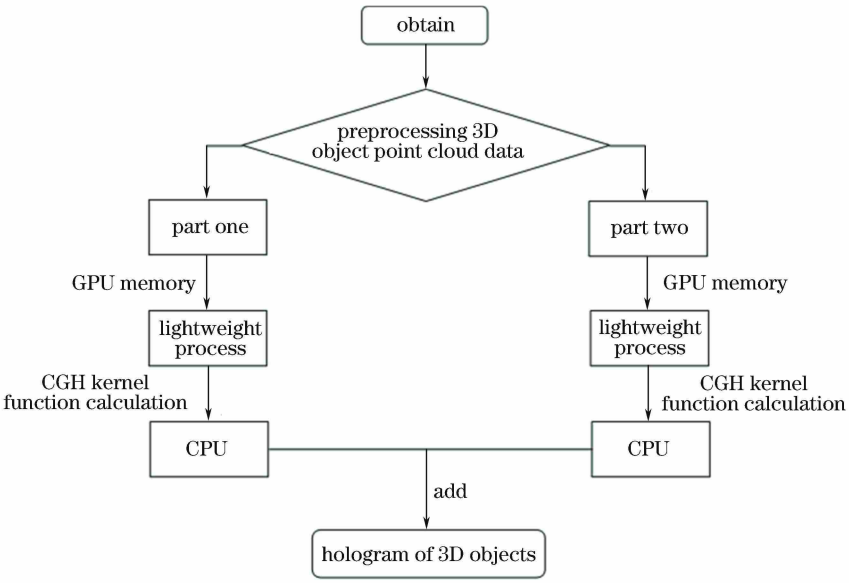

图 4 多 GPU 全息图计算流程

Fig. 4 Flow chart of multi-GPU hologram calculation

表 1 硬件环境

Table 1 Hardware configuration

Hardware	Model parameter
OS	Microsoft Windows 10 Pro(64-bit)
CPU	Intel Core i5-3470
GPU	NVIDIA GeForce GT 610

表 2 NVIDIA Quadro K4000 显卡参数

Table 2 Graphic card parameters of NVIDIA Quadro K4000

Attribute	Parameter
Core	48
Clock frequency /GHz	0.824
Memory /GB	1
Memory bandwidth /(GB·s^{-1})	14.4

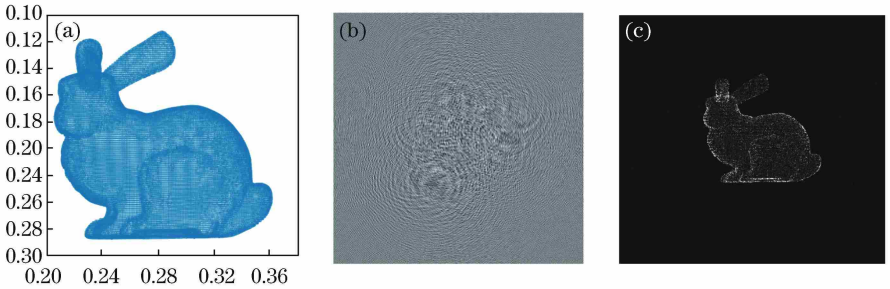

图 5 全息图的模拟及重建结果。(a)实验中的 3D 点云对象;(b)3D 对象的光强分布全息图;(c)计算全息图重建的 3D 对象

Fig. 5 Simulation and reconstruction results of hologram. (a) 3D point cloud objects in experiment; (b) light intensity distribution hologram for 3D object; (c) 3D object reconstructed from computer generated hologram

表 3 三种计算系统的 CGH 生成时间

Table 3 CGH operation time for three kinds of computation systems

Size of hologram /(pixel×pixel)	Rabbit		
	CPU /s	GPU /s	Multi-core GPU /s
512×512	1036.85	20.19	9.86
1024×1024	4017.50	53.95	27.06

1024 pixel 时,基于 GPU 的运行速度约为基于 CPU 的 80 倍。对比基于 Mutil-core GPU 的全息图计算时间同样可以看出,基于 Mutil-core GPU 的运行速度约为基于 GPU 的 2 倍。

5.2 光学重建

实验中搭建光学再现光路如图 6 所示,所得实验结果如图 7 所示,采用波长 $\lambda=0.000532$ mm 的绿光半导体激光器作为重现参考光,激光自左向右依次经过衰减片 P,扩束镜及准直透镜 L 后,投向分辨率为 1920 pixel × 1080 pixel、像元尺寸为 0.0064 mm的纯相位反射式空间光调制器(LCOS),经空间光调制器(SLM)反射,在其后空间,再现出物光波,获取三维物体的再现像。

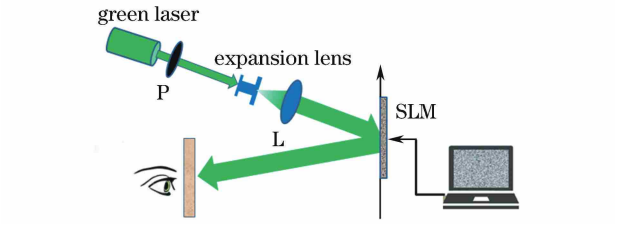

图 6 光学再现光路图

Fig. 6 Optical path for redisplay

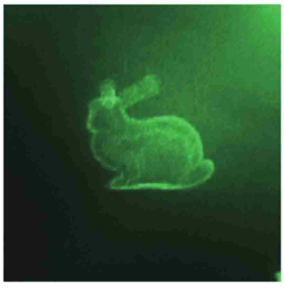

图 7 光学重建像

Fig. 7 Optical reconstruction image

从模拟及实验结果可以看出，基于多 GPU 并行计算全息图的方法不仅能够快速地计算全息图，而且三维模拟及光学再现成像质量较高，表明本文方法在全息三维显示技术具有可行性，是一种快速生成全息图的有效方法。

6 结 论

利用 Matlab 编程的简易性和 GPU 的高计算性能，实现了 Matlab 对多 GPU 编程的计算全息图快速算法。实验结果表明，在文中硬件配置下，单块 GPU 的计算全息图速度是 CPU 的 60 倍；在利用两块 GPU 卡时，计算全息图的速度是单块 GPU 计算速度的 2 倍。由于本文所用的 GPU 并不是主流的计算型卡，因此在计算速度上稍有不足。今后，将利用可升级连接界面(SLI)标准在全息图的计算系统中配置 4 块主流的 teasla 系列显卡，并采用遮挡剔除和更复杂的基元表示三维对象，使全息图的计算速度得到较大的提升，从而实现全息图准实时计算的目的。

参 考 文 献

[1] Matsushima K, Nakahara S. Region segmentation and parallel processing for creating large-scale CGHs in polygon source method[J]. Proceedings of SPIE, 2009, 7233: 72330E.

[2] Nishi H, Higashi K, Arima Y, *et al*. New techniques for wave-field rendering of polygon-based high-definition CGHs [J]. Proceedings of SPIE, 2011, 7957: 79571A.

[3] Matsushima K, Nakahara S. Extremely high-definition full-parallax computer-generated hologram created by the polygon-based method [J]. Applied Optics, 2009, 48(34): H54-H63.

[4] Lucente M E. Interactive computation of holograms using a look-up table [J]. Journal of Electronic Imaging, 1993, 2(1): 28-34.

[5] Okada N, Hirai D, Ichihashi Y, *et al*. Special-purpose computer HORN-7 with FPGA technology for phase modulation type electro-holography [C] // 19th International Display Workshops/Asia Display, 2012, 19: 1261-1264.

[6] Ritter A, Böttger J, Deussen O, *et al*. Hardware-based rendering of full-parallax synthetic holograms [J]. Applied Optics, 1999, 38(8): 1364-1369.

[7] Ahrenberg L, Benzie P, Magnor M, *et al*. Computer generated holography using parallel commodity graphics hardware [J]. Optics Express, 2006, 14(17): 7636-7641.

[8] Shimobaba T, Ito T, Masuda N, *et al*. Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL [J]. Optics Express, 2010, 18(10): 9955-9960.

[9] Takada N, Shimobaba T, Nakayama H, *et al*. Fast high-resolution computer-generated hologram computation using multiple graphics processing unit cluster system [J]. Applied Optics, 2012, 51(30): 7303-7307.

[10] Zhang B D, Xu S, Zhang F, *et al*. Accelerating Matlab code using GPU: a review of tools and strategies[C] // International Conference on Artificial Intelligence, Management Science and Electronic Commerce, 2011: 1875-1878.

[11] Suh J, Kim Y. Accelerating MATLAB with GPU computing [M]. Burlington: Morgan Kaufmann, 2013: 1-17.