doi:10.3969/j.issn.1671-637X.2013.03.014

基于并行计算的全息图快速生成

付胜豪[1]， 王元庆[1]， 鲍绪良[1]， 范科峰[2]
（1.南京大学电子科学与工程学院,南京 210093； 2.中国电子技术标准化研究所,北京 100007）

摘 要：全息图的快速生成是当前计算机制全息的重点与难点,为提高全息图的生成速度,提出一种基于Cuda并行计算的全息图快速生成方案。首先利用OpenGL的深度缓存对空间物体进行离散取样,获得空间离散物点集,再对传统的查表算法进行优化处理,大大减小查找表的空间大小,离线制作查找表,存入GPU纹理内存,合理设计并行计算方案,将Cuda并行计算的方法应用于全息图的快速生成。实验表明该方法有效可行,并行计算可将全息图的生成速度提高40倍左右,同时,OpenGL的使用给交互式全息图的计算机生成提供一种研究思路。

关键词：计算机全息；并行计算；OpenGL；相息；菲涅耳衍射
中图分类号：V271.4；O438 **文献标志码**：A **文章编号**：1671-637X(2013)03-0061-04

Fast Hologram Generation Based on Parallel Computing

FU Shenghao[1], WANG Yuanqing[1], BAO Xuliang[1], FAN Kefeng[2]
(1. School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China;
2. China Electronics Standardization Institute, Beijing 100007, China)

Abstract: Fast hologram generation is the key for the study on Computer Generated Hologram (CGH). In order to accelerate the generation speed of CGH, a fast scheme based on Cuda parallel computing was proposed. Firstly, the objects were sampled by means of OpenGL depth buffer, and objects points set was acquired. Then with the optimization of traditional look-up table algorithm, storage space was reduced, a short look-up table was produced offline and stored in the GPU texture memory. Finally, a reasonable parallel computing program was designed to generate CGH. The experimental results showed that the algorithm was effective and feasible, and the speed was increased by about forty times with the parallel computing technology. At the same time, the use of OpenGL provided an idea for computer-generated interactive holograms.

Key words: Computer Generated Hologram (CGH); parallel computing; OpenGL; Kinoform; Fresnel diffraction

0 引言

随着全息技术以及计算机技术的发展,计算机制全息图逐渐取代了传统的光学全息,克服了光学全息对实验条件要求高的缺点,计算机制全息图可以灵活地生成各种所需的全息图[1]。迄今为止,人们已经探索出各种全息图的计算机生成方案,其中,计算相息图的方案以其衍射效率高[2]等优点尤其引人注意。然而,目前的相息图算法如迭代傅里叶算法[3-4]与模拟退火算法[5]等,都停留在对二维场景全息图制作的阶段。层加傅里叶算法能够有效制作三维场景全息图,而且每次可以对一个深度层面上的物点进行处理,大部分情况效率较高,然而缺乏灵活性,该算法对于深度层面较多且每层包含较少物点的空间物体效率较低。逐点计算的方法计算精度高,操作灵活,然而计算量大,十分耗时,无法达到实时计算的速度。鉴于此,一些加速的算法相继被提出,如针对逐点计算方法的查表算法[6]、主菲涅尔波带算法[7]等,针对傅里叶变换算法的FPGA硬件加速fft算法[8]等。这些算法或者加速效果不明显,或者对硬件要求较高。本文在查表法的基础上将Cuda并行计算用于全息图的生成,有效地加速了全息图的生成速度。同时在空间物点离散采样

收稿日期:2012-02-22 **修回日期**:2012-03-20
基金项目:国家自然基金重点项目资助(608320036);国家质检公益性行业科研专项经费资助项目(201110233)
作者简介:付胜豪(1988—),男,江苏徐州人,硕士生,研究方向为立体图像生成。

环节,利用 OpenGL 的深度缓存提取离散化空间物点数据,再结合 OpenGL 强大的交互性,实现交互式全息图的计算机生成,为交互式全息显示探索方向。

1 相息图

空间物体可以看作是一系列空间点的集合,计算各空间点到全息面上各像素点的距离,进而计算出各物点在该像素点的空间复振幅,叠加所有物点在该处的复振幅,取其相位信息,舍去振幅信息即可得到相息图[9]。物体各点发出的光传播一定距离后,在全息面上复振幅可表示为

$$u(x,y)=\sum_{n=0}^{N}a_n\exp(jkr_n+j\varphi_{r_n}) \tag{1}$$

式中:r_n 为物点 n 到全息面上 (x,y) 点的距离;a_n,φ_{r_n} 分别为第 n 个物点衍射光波的振幅和相位;N 为物点总数;k 为波数。

式(1)可以表示成

$$u(x,y)=A(x,y)\exp[j\varphi(x,y)] \tag{2}$$

式中,$A(x,y)$ 和 $\varphi(x,y)$ 分别表示整个物体的物光波在全息面 (x,y) 点的振幅和相位分步。

将振幅信息舍去,即令 $A(x,y)$ 为固定的值1,可得相息图,由 $\varphi(x,y)$ 调制纯相位空间光调制器,从而对入射光的相位信息进行调制,重建出原空间场景。

2 方案设计

2.1 OpenGL 读取空间物点信息

全息图的计算机生成,首先要获取空间物点的点集信息。本文采用 OpenGL 深度缓存来获取空间点集信息,首先利用 OpenGL 在虚拟坐标中绘制待显示的三维场景,再通过深度缓存读取各物点深度信息,通过坐标变换获取场景中各物点空间信息。另外,当进行交互操作,如空间物体移动时,能够即时更新空间点集的信息,重新计算全息图,实现交互式全息图的计算机生成。

OpenGL 获取空间物点信息的详细流程见图1。

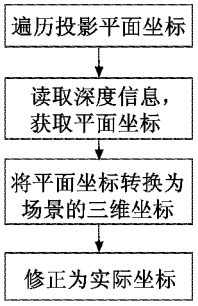


图1 OpenGL 获取物点空间信息流程图

Fig. 1 Flow chart of getting object points by OpenGL

首先通过 glReadPixels( ) 函数读取深度缓冲区的深度值;再利用函数 gluUnProject( ) 将平面坐标 (x,y,z) 转换为对应场景的三维坐标,这个坐标是相对于视点的,将其进行修正转换为实际三维空间坐标值;最后将获取的实际坐标近似规整到标准的空间物体网格坐标中,为下一步进行全息计算提供物点集数据。

2.2 改进查表算法

逐点计算物点到各全息面像素点的距离,进而计算相位,大量的计算量耗费在计算距离与相位上,为了加速全息图的生成速度,传统的查表算法是离线计算好所有可能用到的相位信息,并存入查找表,在线运行时通过查找获取相应的相位因子,减小运行时的计算负担,提高运行速度。传统查表算法需要存储的表格数据量庞大,以 $M\times N\times K$ 的空间场景为例(其中,K 为空间场景深度方向层数),需要的存储空间为 $M\times N\times K$。为了利用 Cuda 并行计算加速全息图的生成速度,首先需要将相位因子查找表存入设备存储区域,即 GPU 的内存区域,由于 GPU 内存有限,对于传统查表法如此庞大的数据很难实现。本文对传统查表法优化,将相位因子拆成相乘的两部分,从而存储量可降为 $(M+N)\times K$,为后续利用并行计算做准备。

式(1)中

$$r_n=\sqrt{(x_n-x)^2+(y_n-y)^2+z_n^2} \tag{3}$$

式中:(x_n,y_n,z_n) 为空间物点的坐标;(x,y) 为全息面上的像素点。对式(3)进行菲涅尔近似

$$r_n=z+\frac{(x_n-x)^2+(y_n-y)^2}{2z} \tag{4}$$

则式(1)可表示为

$$u(x,y)=\sum_{n=0}^{N}a_n e^{j\varphi_{r_n}}H(\Delta_x,z)V(\Delta_y,z) \tag{5}$$

式中:

$$\Delta_x=x_n-x,\quad \Delta_y=y_n-y\quad; \tag{6}$$

$$H(\Delta_x,z)=\exp[jk(z+\frac{\Delta_x^2}{2z})]\quad; \tag{7}$$

$$V(\Delta_y,z)=\exp(jk\frac{\Delta_y^2}{2z})。 \tag{8}$$

从而,查找表中只需存储 $H(\Delta_x,z)$ 和 $V(\Delta_y,z)$ 信息即可,通过两者的乘积即可获取光波传播过程中产生的相位。改进后极大缩小了查找表的大小。

2.3 并行计算加速全息图的生成

Cuda 是 Nvidia 公司推出的通用并行计算架构,应用该架构可以利用 GPU 来并行地处理复杂的计算问题[10]。通常 GPU 有比 CPU 更多的处理单元,尽管 GPU 单独的处理单元处理能力不及 CPU,但利用 GPU 多处理单元的并行处理能力,能够快速处理复杂计算问题。利用 GPU 并行加速计算,关键在于并行块与线程的合理分配,使 GPU 的处理单元得到最大限度的利用[11]。

本文在设计 Cuda 核函数时，使二维线程块数为 $m \times n$（全息图的分辨率），每个块中的线程数为 M（M 为 2 的整数次幂，即 $M = 2^i$ 这么设计是为了方便后续的归约求和），每个块代表全息图的一个像素点，每个线程代表一个空间物点，并在线程内计算空间物点在该像素点处的复振幅，当空间物点大于线程数 M 时，使用循环叠加的方法进行解决，从而能够并行地计算空间各物点发出的光波传播到全息图上各点处的复振幅。线程块与块内线程的分配如图 2 所示。

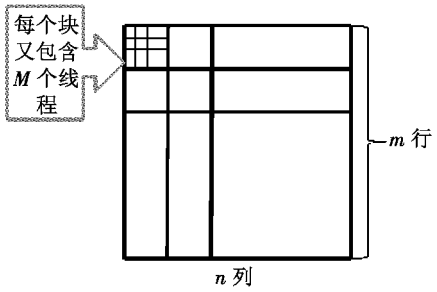


图 2　块及线程分配

Fig. 2　Distribution of blocks and threads

并行计算出空间各物点光波传播到全息图上各像素点处的复振幅后，需要将同一像素点处的所有物光波叠加，即将各个块内所有线程运算出的结果分别进行叠加求和。由于这一步的运算都是在块内进行，而 GPU 的共享内存在同一个块内进行数据操作具有较高的运算效率，因此为加快运算速度，引入共享内存来存储线程运算结果。在每个块内开辟 M 个共享内存空间 Cach[M]，用以存储 M 个线程的计算结果，最后再将共享内存的结果加起来。将共享内存中 M 个数据加起来的过程可以在一个线程上迭代完成，计算时间与数组的长度成正比，但为了充分利用并行计算的优势，在这里采用归约求和的方法，每个线程将 Cach[ ] 中的两个值加起来，假设 Cach[ ]数组的长度为 l，那么这一步就会有 $l/2$ 个线程并行地进行计算，比单线程迭代计算效率要高得多，并行相加后将结果存回 Cach[ ]，完成这个步骤后所得结果数据量就是原来的一半，在下一步中再对这一半数据执行同样的操作，执行 $\log 2(M)$ 个步骤后，就能得到 Cach[ ]中所有的数据总和，这也是设计块内线程数时强调要将每个块内的线程数设计为 2 的整数次幂的原因。以长度为 4 的数组为例，如图 3 所示，第一步 Cach[0]与 Cach[2]相加结果存入 Cach[0]，Cach[1]与 Cach[3]相加结果存入 Cach[0]，这两个运算是并行执行的，第二步 Cach[0]与 Cach[1]相加，存入 Cach[0]，即为最终结果。数组长度比较大的情况下，该方法效果明显。

至此，已经计算出整个空间场景发出的光波在全息图各像素点处的复振幅分布，并存储于各个块的共享内存 Cach[0]中，只需提取其相位信息，用以调控纯相位空间光调制器即可进行全息场景显示。

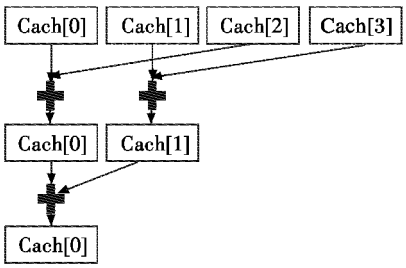


图 3　归约求和

Fig. 3　Reduction summation

3　实验验证

3.1　立体图像全息图及再现仿真

如图 4a 所示，x-y 平面为全息图所在的平面，空间中物体为距全息面的距离分别为 131 mm、151 mm、172 mm 和 192 mm 的 4 张图像。取空间物点的采样间隔与全息图采样间隔相等，都为 20 μm，每个面图像采样为大小 256 × 256 的矩阵。采用改进查表法，通过上述 Cuda 并行加速计算，得到的全息图如图 4b 所示，采用数值计算的方法对得到的全息图进行模拟衍射再现，重建原场景的图像，并在 131 mm、151 mm、172 mm 和 192 mm 距离处获取再现像，如图 5 中的 a ~ d 所示。实验表明，并行计算用于全息图的计算，能够正确生成全息图，并重建出原始空间场景。

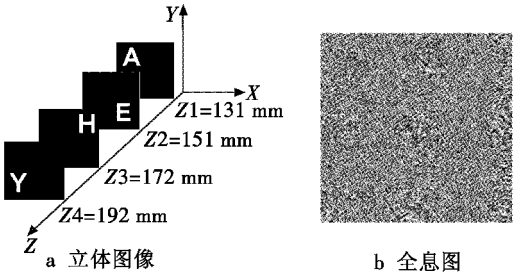


图 4　立体图像及其全息图

Fig. 4　The image of 3-D entity and the hologram

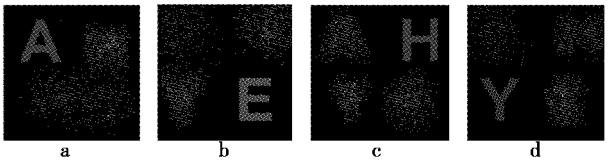


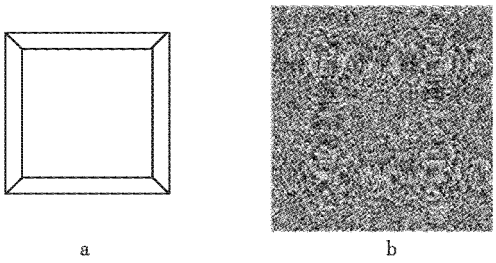
图 5　全息图在不同深度上的再现像

Fig. 5　The reconstructed images at different distance

3.2　交互式立体全息图

使用 OpenGL 绘制一个立方体，如图 6a 所示，读取空间物点信息，规整到标准的空间网格中，使用并行计算得出全息图，如图 6b 所示；通过交互操作改变立方体在场景中的位置，如图 7a 所示，重新读取物点信息，

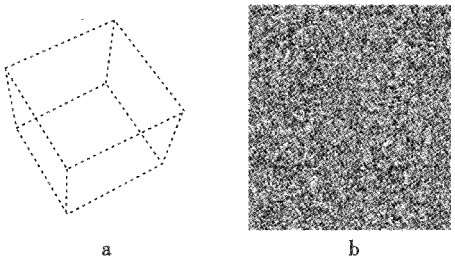
并计算全息图，如图 7b 所示。实验表明，将 OpenGL 用于物点离散采样，并进行交互操作，能够即时生成正确的全息图。



a b

图 6 正方体及其全息图

Fig. 6 Cube and its hologram



a b

图 7 改变位置后的场景及全息图

Fig. 7 Updated scene and its hologram

3.3 生成速度

并行计算的引入大大加快了全息图的生成速度，在此以基于 CPU 的全息计算与基于 GPU 并行计算的全息计算进行比较，结果如表 1 所示。硬件平台：处理器为 Intel Core2Duo E5300，内存 2 GB，英伟达 GT240 显卡。软件平台：Win7 操作系统，Vs2010 编译环境，Cuda 4.0 通用开发包。

表 1 运算时间测试表

Table 1 Table of computation time

空间物点数	基于 Cuda 并行计算/ms	基于 CPU/ms
100	282	4370
300	445	13053
1200	1089	41169
3000	2480	141059

4 结束语

计算机制全息图计算量大，计算时间长，一直以来都是困扰计算机制全息图发展的一大难题。本文在查表法的基础上，对查表法进行改进，将 Cuda 并行计算技术用于全息图的生成，能够正确生成所需的全息图，大大加快了全息图的生成速度，同时将 OpenGL 用于空间物点的离散采样与交互控制环节，为交互式计算机制全息图提供一定的参考作用。

参 考 文 献

[1] 虞祖良，金国藩. 计算机制全息图[M]. 北京：清华大学出版社，1984.

[2] 许富洋，李勇，金洪震，等. 三维场景相息图的白光再现研究[J]. 光子学报，2010，39(2)：271-274.

[3] HACKER M, STOBRAWA G, FEURER T. Iterative Fourier transform algorithm for phase-only pulse shaping[J]. Optics Express, 2001, 9(4)：191-199.

[4] RIPOLL O, KETTUNEN V, HERZIG H. Review of iterative Fourier-transform algorithms for beam shaping applications[J]. Optical Engineering, 2004, 43(11)：2549-2556.

[5] YANG H J, CHO J S, WON Y H. Rduction of reconstruction errors in Kinoform CGHs by modified simulated annealing algorithm[J]. Opt. Soc. 2009, 13(1)：92-97.

[6] 金洪震，李勇，王辉. 实现相息图快速计算的一种方法[J]. 应用激光，2002，22(1)：13-14.

[7] 张晓洁，刘旭，陈晓西. 利用菲涅尔波带法计算三维全息[J]. 光电工程，2004，31(12)：58-60.

[8] ITO T, SHIMOBABA T. 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[J]. Optics Express, 2004, 12(9)：1788-1793.

[9] 林培秋，王辉，庞辉. 基于液晶空间光调制器的相息图扫描三维成像[J]. 光电工程，2010，37(3)：138-143.

[10] 孙迎红，童元满，王志英. RSA 算法的 CUDA 高效实现技术[J]. 计算机工程与应用，2011，47(2)：84-98.

[11] 仇德元. GPGPU 编程技术：从 GLSL、CUDA 到 OpenCL[M]. 北京：机械工业出版社，2011.