基于 OptiX 光线跟踪引擎的光线跟踪全息图生成算法

孙敏远[1,2,3]，袁园[3]，毕勇[3*]，朱建英[2,3]，张硕[2,3]，张文平[3]

[1]中国科学院空天信息创新研究院光学工程研究部，北京 100094；

[2]中国科学院大学，北京 100049；

[3]中国科学院理化技术研究所应用激光研究中心，北京 100190

摘要 为了实现全息图的快速计算，提出了一种基于 OptiX 光线跟踪引擎和 NVIDIA 图形处理器(GPU)的光线跟踪全息图生成算法。该算法充分利用了 GPU 中的硬件光线跟踪核心，可有效提高全息图的计算速度。当组成三维模型的多边形数量为 1.6 万个，物点数量为 4 万个时，该光线跟踪全息图生成算法的计算速度约为基于 GPU 的点源全息图生成算法的 11.5 倍。

关键词 全息；计算全息；光线跟踪；图形处理器；光线跟踪引擎

中图分类号 O438.1 **文献标志码** A **doi**：10.3788/LOP57.240901

Ray-Tracing Hologram Generation Algorithm Based on OptiX Ray-Tracing Engine

Sun Minyuan[1,2,3], Yuan Yuan[3], Bi Yong[3*], Zhu Jianying[2,3], Zhang Shuo[2,3], Zhang Wenping[3]

[1]*Optical Engineering Research Department, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing* 100094, *China*;

[2]*University of Chinese Academy of Sciences, Beijing* 100049, *China*;

[3]*Applied Laser Research Center, Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, Beijing* 100190, *China*

Abstract In order to realize the fast calculation of computer-generated holograms, a ray-tracing algorithm is proposed, which is based on the OptiX ray-tracing engine and the NVIDIA graphic processing unit (GPU). This algorithm makes full use of the ray-tracing hardware core in GPU and thus effectively increase the calculation speed of holograms. When the three-dimension model is consisted of 1.6×10^4 polygons and 4×10^4 points, the calculation speed of the proposed algorithm is 11.5 times that of the traditional point source hologram generation algorithm based on GPU.

Key words holography; computer-generated holography; ray-tracing; graphic processing unit; ray-tracing engine

OCIS codes 090.1760; 090.1795; 090.2870

1 引 言

全息三维显示技术可再现三维场景的全部信息，是未来显示技术的重要发展方向。目前基于空间光调制器(SLM)的计算全息(CGH)是研究的主流，但计算全息存在计算量大和耗时长的问题，这不利于全息三维显示技术的应用。

计算全息广泛使用点源全息图生成算法[1]。在该算法中，三维模型被离散为多个发光的物点，分别计算该物点到达全息平面各个采样点处的光线复振幅，并将其叠加以获得全息图。为了提高点源全息图生成算法的计算速度，研究人员一方面提出如查找表法(LUT)[2-3]和波前平面记录(WRP)法[4]等快速算法，另一方面使用现场可编程逻辑门阵列

收稿日期：2020-04-14；**修回日期**：2020-05-15；**录用日期**：2020-06-01

基金项目：国家重点研发计划(2016YFB0401902)

*E-mail：biyong@mail.ipc.ac.cn

(FPGA)[5]、协处理器(Xeon Phi)[6]和图形处理器(GPU)[7]等高性能硬件并通过并行计算的方式来提高全息图的计算速度。

光线跟踪全息图生成算法是点源全息图生成算法的一种改进算法。光线跟踪(Ray-Tracing)最早由 Whitted 在 1979 年提出[8]。2013 年,Ichikawa 等[9]将光线跟踪应用于计算全息中,生成的全息图去除了隐藏表面,渲染效果较为逼真。2018 年,Wang 等[10]将 WRP 方法应用于光线跟踪全息图生成算法中,极大提高了全息图的计算速度。光线跟踪全息图生成算法相比于传统点源全息图生成算法,主要有两点不同。一是计算过程中考虑了光线传播过程中的折射和反射现象,能够还原场景中三维模型的遮挡关系与光影信息,生成的图形更加逼真。二是点源全息图生成算法的计算复杂度一般与物体离散点数以及全息面分辨率相关,而光线跟踪全息算法的计算复杂度与全息平面分辨单元的数量以及每个分辨单元所发射的虚拟光线数量有关。当组成三维模型的离散点数量较少时,点源全息图生成算法的计算速度较快;当组成三维模型的点数量增加时,光线跟踪算法的计算量变化较小,因而在复杂场景下其计算速度有一定的优势。

在光线跟踪全息图生成算法中,需要对全息图中虚拟光线与组成三维模型的多边形进行求交计算且后续还需要进行光场复振幅计算,计算量十分巨大。为了提高全息图的计算速度,本文提出了一种基于 OptiX 光线跟踪引擎的光线跟踪全息图生成算法,并在英伟达(NVIDIA)含有硬件光线跟踪计算单元(RT Core)的 RTX 系列 GPU 上进行了全息图计算与重建实验,比较了点源全息图生成算法在中央处理器(CPU)和图形处理器(GPU)以及光线跟踪全息图生成算法在 RTX/GTX 系列 GPU 上的计算速度。研究结果表明,光线跟踪全息图生成算法的计算速度较快。

2 基本原理

2.1 光线跟踪全息图生成算法

点源全息图生成算法的计算过程如图 1(a)所示。在点源全息图生成算法中,三维模型被离散为若干独立的发光物点,分别计算每个发光物点发射的光线到达全息面上各个采样点处的复振幅并将其进行叠加以获得物体的全息图。全息平面位于 (x_H, y_H, z_H) 采样点处的光场复振幅为

$$U(x_H, y_H, z_H) = \sum_{i=1}^{N} \frac{A_i}{r_i} \exp\left[\mathrm{j}(kr_i + \varphi_i)\right], \tag{1}$$

式中:N 为组成三维模型的物点的数量;A_i 为第 i 个物点发射光波的振幅,r_i 为该物点到全息平面采样点的距离且 $r_i = \sqrt{(x_H - x_i)^2 + (y_H - y_i)^2 + (z_H - z_i)^2}$,其中 (x_i, y_i, z_i) 为第 i 个物点的坐标;k 为波数,φ_i 为该物点发射光的初始相位,对于漫反射体,一般认为该初始相位在 $(0, 2\pi)$ 范围内随机分布。

光线跟踪全息图生成算法是点源全息图生成算法的反向运算,其计算过程如图 1(b)所示。从全息面的每个采样点处向三维模型发射若干条虚拟光线,计算虚拟光线与三维模型的交点并根据模型表面特征基于漫反射、镜面反射或者透射生成新的虚拟光线,计算生成的虚拟光线与三维模型的交点,如此反复递归,直到虚拟光线到达光源处或者超出三维模型所在的空间范围。全息面采样点 (x_H, y_H, z_H) 处的光场复振幅 $U(x_H, y_H, z_H)$ 仍根据(1)式进行计算,其中交点处的振幅 A_i 由光源的色彩、亮度以及虚拟光线与模型交点处的反射率和颜色给出,距离 r_i 为与三维模型相交的虚拟光线的传播距离。

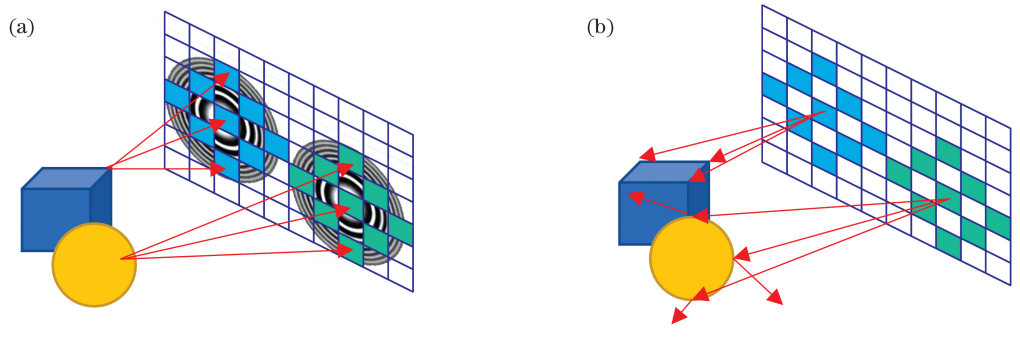

图 1 全息图生成算法示意图。(a)点源法;(b)光线跟踪法

Fig. 1 Schematic of hologram generation algorithm. (a) Point source algorithm; (b) ray-tracing algorithm

在光线跟踪全息图生成算法中，全息面上每个采样点发射的光线数量 N 取决于场景中模型的分辨角度和空间光调制器(SLM)的衍射角度，$N=\arcsin(\lambda/d)/\Delta\theta$，其中 d 为全息面采样点的间距，λ 为波长，$\Delta\theta$ 为所选取的分辨角度。

2.2 基于 OptiX 光线跟踪引擎的全息图生成算法

在光线跟踪全息图生成算法中，每一条虚拟光线都需要与组成三维模型的多边形进行求交计算和振幅计算，运算量十分巨大，这限制了光线跟踪全息算法的应用。本文使用基于 OptiX 光线跟踪引擎的全息图生成算法，采用硬件并行计算的方式来提高全息图的计算速度。

OptiX 是 NVIDIA 公司推出的光线跟踪计算引擎，应用在 NVIDIA 公司的 GPU 硬件上。OptiX 光线跟踪程序的结构如图 2 所示[11]。每一个 OptiX 光线跟踪程序都包含有一个或若干场景(Context)，场景由描述光线发射的 Ray Generation 程序、描述光线未与三维模型相交的 Miss 程序、处理异常情况的 Exception 程序和描述场景中三维模型特征的几何体组(Geometry Group)构成，其中几何体组通常包括若干几何体实例(Geometry Instance)。几何体实例中的几何体(Geometry)部分是由 Intersection 程序进行虚拟光线与三维模型的相交计算，由 Bounding Box 程序确定碰撞盒并加速相交计算过程；材质(Material)部分由 Closet Hit 程序计算虚拟光线的强度、颜色信息并生成反射、折射光线，由 Any Hit 程序处理三维模型在光源照射下的阴影。

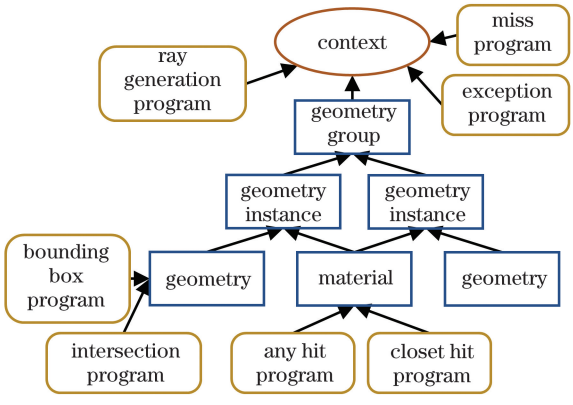

图 2 OptiX 光线跟踪程序的结构

Fig. 2 Structural diagram of OptiX ray-tracing program

基于 OptiX 光线跟踪引擎的全息图生成算法原理如图 3 所示。对组成三维模型的多边形进行计算并生成几何体，通过加入由 Closet Hit 和 Any Hit 程序所描述的材质信息，生成一个几何体实例。在光线跟踪过程中，由光线发射程序根据全息图平面的位置、像素尺寸、光线衍射角度以及虚拟光线间隔角度向三维模型发射若干虚拟光线。每一条虚拟光线 $\boldsymbol{r}_{\mathrm{ray}}$ 可表示为

$$\boldsymbol{r}_{\mathrm{ray}}=\boldsymbol{o}_{\mathrm{origin}}+t\times\boldsymbol{d}_{\mathrm{direction}},\tag{2}$$

式中：$\boldsymbol{o}_{\mathrm{origin}}$ 为该虚拟光线的原点；$\boldsymbol{d}_{\mathrm{direction}}$ 为虚拟光线的方向向量；t 为该虚拟光线的长度。

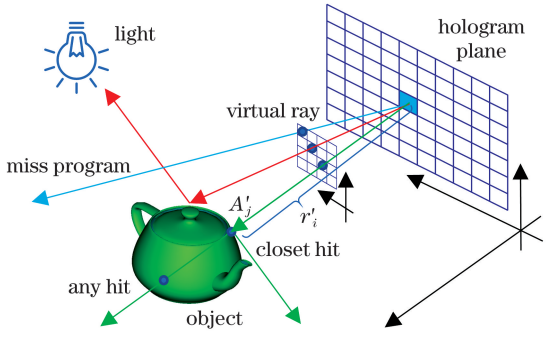

图 3 光线跟踪全息生成算法的原理

Fig. 3 Principle of ray-tracing hologram generation algorithm

虚拟光线生成后，OptiX 光线跟踪引擎自动对每一条虚拟光线与组成几何体实例的多边形进行求交计算。如果虚拟光线与若干多边形相交，则可计算出每个虚拟光线/多边形交点的光线方程(2)式中的 t 值。其中最小的 t 值 $t_{\min}$ 对应于距离虚拟光线原点最近的交点位置。随后光线跟踪引擎调用 Closet Hit 程序根据 Phong 光照模型或其他光照模型计算当前交点处的光强。对于当前虚拟光线与几何体实例相交计算获得的其他 t 值，则调用 Any Hit 程序进行阴影处理；如果虚拟光线没有与几何体实例相交则调用 Miss 程序进行处理，利用该虚拟光线可获得场景中背景的强度与颜色信息。虚拟光线与场景中的几何体实例进行求交计算完成后，由 Closet Hit 程序计算虚拟光线发射点(全息平面分辨点)$(x'_{\mathrm{H}},y'_{\mathrm{H}})$处的光场复振幅 $U'(x'_{\mathrm{H}},y'_{\mathrm{H}})$：

$$U'(x'_{\mathrm{H}},y'_{\mathrm{H}})=\sum_{i'=1}^{N'}\frac{A'_{i'}}{r'_{i'}}\exp\left[\mathrm{j}(k'r'_{i'}+\varphi'_{i'})\right],\tag{3}$$

式中：N' 为全息平面 $(x'_{\mathrm{H}},y'_{\mathrm{H}})$ 位置处所发射虚拟光线的数量；i' 为虚拟光线的序号；k' 为波数；$A'_{i'}$、$\varphi'_{i'}$ 和 $r'_{i'}$ 分别为第 i' 条虚拟光线与几何体实例的一系列交点中距离虚拟光线原点最近的交点处的光场振幅、初始相位及第 i' 条虚拟光线的长度，且 $r'_{i'}$ 由光线跟踪引擎自动计算给出。由此遍历全息面上每个分辨单元，即可得到整幅全息图的复振幅信息。

光线跟踪全息图生成算法的流程如图 4 所示，

程序由运行于 CPU 的主机(Host)端 C 代码和运行于 GPU 的设备(Device)端 CUDA C 代码组成。程序运行开始后,首先在主机端完成场景的初始化,包括设置 Ray Generation、Miss 和 Exception 程序,设置对应于场景几何结构的 Intersection、Bounding Box 程序和对应于材质的 Closet Hit、Any Hit 程序。之后将场景载入设备端并利用二维的包含 $M \times N$ 个元素的计算网格开始光线跟踪计算。在设备端,OptiX 光线跟踪引擎并行完成虚拟光线生成、碰撞计算、光场复振幅计算以及复振幅的累加并将结果输出至位于显存中的缓冲区(Buffer)内。完成光线跟踪后,储存在缓冲区内的全息光场复振幅被传输至计算机,计算机根据光场复振幅生成相息图,并传输至 SLM 完成光学复现。

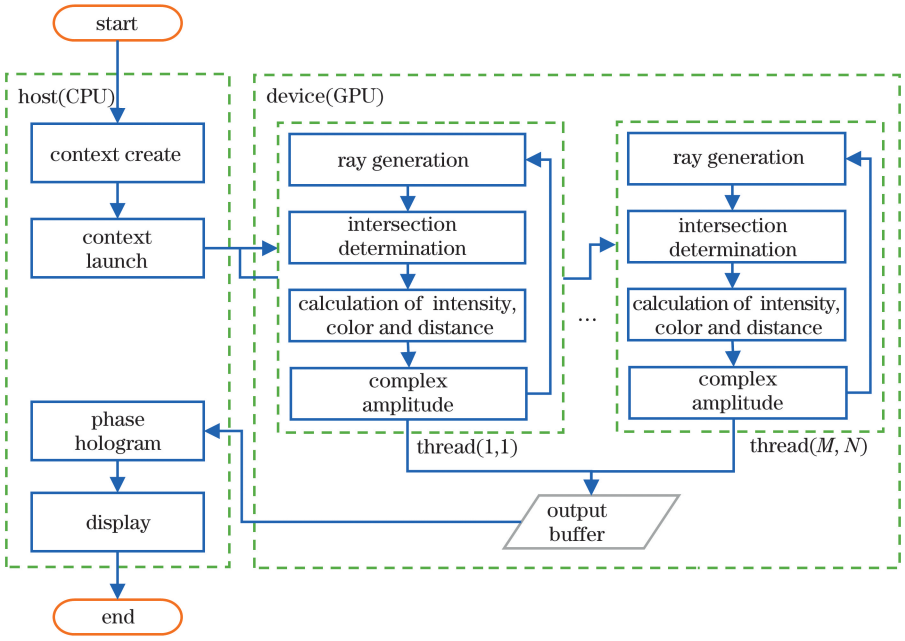

图 4 光线跟踪全息图生成算法的流程

Fig. 4 Flow chart of ray-tracing hologram generation algorithm

3 实验结果

3.1 算法实现

在实验平台分别测试点源全息图生成算法(CPU/GPU)以及光线跟踪全息图生成算法(GPU)的计算速度。实验采用的硬件为 CPU INTEL i7-4790K (4 cores, 8 threads, 4.5 GHz), GPU1 NVIDIA GTX 1060 (1280 CUDA cores, 1506～1708 MHz), GPU2 NVIDIA RTX 2060 (1920 CUDA cores, 30 RT cores, 1365～1680 MHz),计算能力对比如表 1 所示,其中 S 为每秒单精度浮点运算速度,R_1 为每秒进行光线跟踪的光线数量,R_2 为综合评价 GPU 运算能力的参数。点源法程序采用 MATLAB 2019b 编写,光线跟踪法程序使用 C 语言编写,编译环境为 CUDA 10.1, OptiX 6.5。

表 1 CPU 与 GPU 计算能力的对比

Table 1 Comparation of computing power between CPU and GPU

Computing power	CPU (i7-4790K)	GPU1 (GTX 1060)	GPU2 (RTX2060)
S /10^{12}	0.58	3.85	5.24
R_1/(10^9 s^{-1})			5
R_2/(10^{12} s^{-1})			37

全息图实验的参数如表 2 所示。实验一选取复杂程度逐步提高的薄四面体(含 2500 个点源和 12 个多边形)、牛(含 10084 个点源和 5804 个多边形)和茶壶(含 41472 个点源和 16188 个多边形)三维模型进行计算,在保持三维模型比例的情况下统一设置模型宽度为 10 mm。该实验用于比较点源全息

图生成算法与光线跟踪全息图生成算法在对不同复杂度三维模型进行全息图生成计算时的性能区别。实验二选取 GPU2 平台，固定三维模型形状以及虚拟光线的发射角度和数量，分别设置三维模型的宽度为 10 ，15 ，20 mm，如表 3 所示。该实验用于比较光线跟踪全息图生成算法对不同尺寸模型的计算速度。可见对于同一模型，随着模型尺寸的增加，全息图的计算时间增加。

表 2 全息图实验参数

Table 2 Experimental specifications of holograms

Parameter	Value
Hologram resolution /(pixel×pixel)	2048×2048
Hologram pixel pitch /μm	3.74
Hologram size /(mm×mm)	7.66×7.66
Light wavelength /nm	638
Diffraction distance /mm	600

表 3 不同三维模型宽度下光线跟踪全息图生成算法的运行时间

Table 3 Running time of ray-tracing algorithm for 3D models with different sizes unit: s

Model	Model width of 10 mm	Model width of 15 mm	Model width of 20 mm
Plane (10 mm×10 mm)	8.61	9.62	11.05
Cow (10 mm×6 mm)	8.08	8.44	8.91
Teapot (10 mm×4.9 mm)	8.06	8.38	8.81

点源全息图生成算法根据(1)式使用 MATLAB 分别在 CPU 与 GPU 平台上进行计算，其中 CPU 程序使用向量化运算代替 for 循环以提高 CPU 利用率，进而提高计算速度；GPU 程序使用 MATLAB 并行计算工具箱，以 gpuArray 方式调用 GPU 执行计算[12]。光线跟踪全息图生成算法使用 OptiX 光线跟踪引擎根据(2)～(4)式进行计算，设定全息平面上每个分辨单元向三维模型发射 100×100 条光线。

两种算法运行时间的对比如表 4 所示。点源全息图生成算法在 CPU 和 GPU 上的运行时间与组成三维模型的物点数成正比，并且基于 GPU 的点源法计算速度约为基于 CPU 的点源法的 55 倍。在光线跟踪全息图生成算法中，组成物体的多边形由 12 个增加至 16188 个，但程序的计算时间没有显著增加。对于最复杂的茶壶模型，基于 GPU 的光线跟踪全息图生成算法的计算速度约为基于 GPU 的点源全息图生成算法的 11.5 倍；光线跟踪全息图生成算法在调用硬件光线跟踪单元的情况下，RTX 系列 GPU 的计算速度平均约为无硬件光线跟踪单元的 GTX 系列 GPU 的 3.5 倍。

表 4 各算法运行时间

Table 4 Running time of each algorithm unit: s

Model	Point source algorithm		Ray-tracing algorithm	
	CPU	GPU2	GPU1	GPU2
Plane	313.07	5.61	29.54	8.61
Cow	1214.73	21.65	29.17	8.08
Teapot	4999.91	92.89	29.17	8.06

3.2 光学重建

光学重建实验光路如图 5 所示，以 638 nm 半导体激光器为光源，激光经过扩束后照射至 SLM，全息计算程序调用 SLM API 函数将计算后的全息图加载到 SLM 上。激光经 SLM 调制后被分光棱镜(BS)反射，利用一套 $4f$ 系统以及配套的空间滤波器过滤零频光，最后对入射至 CCD(Charge Coupled Device)的激光进行图像拍摄。基于多边形和点云茶壶模型，分别使用光线跟踪算法和点源算法进行光学重建实验，茶壶宽度设置为 12.5 mm，茶壶模型下方设置为边长 15 mm 为的平面模型并呈 30°倾斜布置，模型中心点距离全息平面 100 mm，以突出重建图像的景深效果。图 6(a)所示为多边形三维模型图像，图 6(b)、(c)所示为光线跟踪方法生成的全息图在距离全息平面 95 mm 处和 105 mm 处的重建图像，图 6(d)所示为点云三维模型图像，图 6(e)、(f)是点源法生成的全息图在距离全息平面 95 mm 处和 105 mm 处的重建图像。由图 6 对

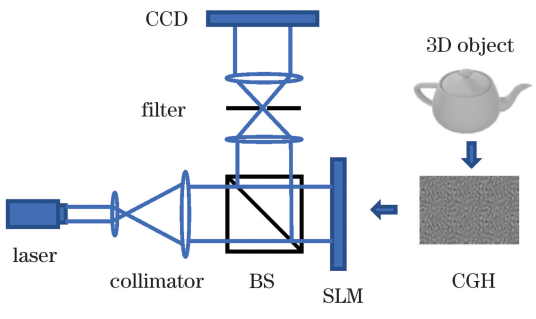

图 5 光学重建实验光路

Fig. 5 Optical path for reconstruction experiment

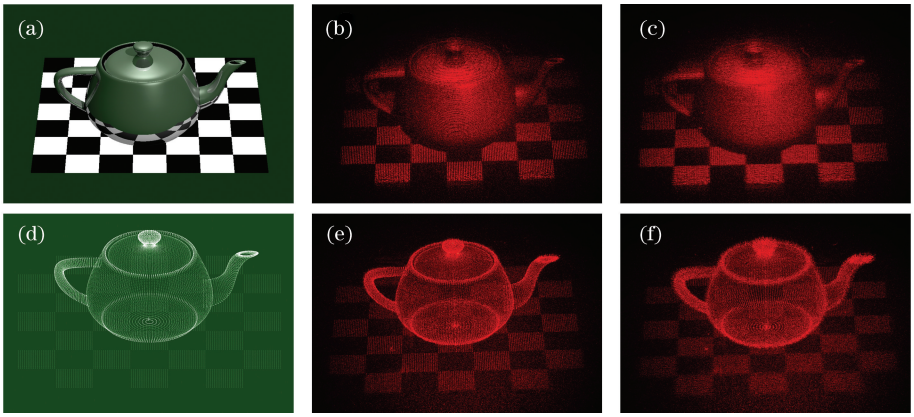

图 6 三维模型及不同距离处的光学重建结果。(a)多边形模型图像;光线跟踪算法生成的全息图在距离全息平面 (b) 95 mm 和 (c) 105 mm 处的重建图像;(d)点云模型图像;点源算法生成的全息图在距离全息平面 (e) 95 mm 和 (f) 105 mm 处的重建图像

Fig. 6 3D model and optical reconstructed images at different distances. (a) Polygon model image; reconstructed images by ray-tracing algorithm at (b) 95 mm and (c) 105 mm away from hologram plane; (d) point cloud model image; reconstructed images by point cloud algorithm at (e) 95 mm and (f) 105 mm away from hologram plane

比可见,光线跟踪算法再现了点源法无法实现的三维模型遮挡关系以及光照效果,该算法在保证较高计算速度的同时可实现逼真的图像重建效果。

4 结 论

为了提高全息图像的计算速度,提出了一种基于 OptiX 光线跟踪引擎的光线跟踪全息图生成算法。当场景复杂度增加时,基于 GPU 的光线跟踪全息图生成算法的计算速度比基于 GPU 的点源全息图生算法提高了 11.5 倍左右。当场景中的三维模型包含的多边形数量大且具有复杂的遮挡关系时,所提算法可实现逼真的渲染效果,具有很强的应用价值。后续将通过优化光线发射密度与方向,利用多 GPU 并行处理的方式从算法和硬件两方面进一步提升全息图像的计算速度。

参 考 文 献

[1] Jin X Y, Gui J B, Liu C, et al. Progress of fast generation algorithm of computer-generated hologram based on point source model [J]. Laser & Optoelectronics Progress, 2018, 55(10): 100005.
金晓宇, 桂进斌, 刘超, 等. 基于点源模型计算全息图快速生成算法的研究进展[J]. 激光与光电子学进展, 2018, 55(10): 100005.

[2] Pi D P, Liu J, Kang R D, et al. Reducing the memory usage of computer-generated hologram calculation using accurate high-compressed look-up-table method in color 3D holographic display [J]. Optics Express, 2019, 27(20): 28410-28422.

[3] Jiang X Y, Cong B, Pei C, et al. A new look-up table method of holographic algorithms based on compute unified device architecture parallel computing[J]. Acta Optica Sinica, 2015, 35(2): 0209001.
蒋晓瑜, 丛彬, 裴闯, 等. 一种基于新型查表方法的统一计算设备架构并行计算全息算法[J]. 光学学报, 2015, 35(2): 0209001.

[4] Arai D, Shimobaba T, Nishitsuji T, et al. An accelerated hologram calculation using the wavefront recording plane method and wavelet transform [J]. Optics Communications, 2017, 393: 107-112.

[5] Nishitsuji T, Yamamoto Y, Sugie T, et al. Special-purpose computer HORN-8 for phase-type electro-holography [J]. Optics Express, 2018, 26(20): 26722-26733.

[6] Murano K, Shimobaba T, Sugiyama A, et al. Fast computation of computer-generated hologram using Xeon Phi coprocessor [J]. Computer Physics Communications, 2014, 185(10): 2742-2757.

[7] Ikawa S, Takada N, Araki H, et al. Real-time color holographic video reconstruction using multiple-graphics processing unit cluster acceleration and three spatial light modulators[J]. Chinese Optics Letters, 2020, 18(1): 010901.

[8] Whitted T. An improved illumination model for shaded display [J]. Communications of the ACM, 1980, 23(6): 343-349.

[9] Ichikawa T, Yamaguchi K, Sakamoto Y. Realistic expression for full-parallax computer-generated

holograms with the ray-tracing method[J]. Applied Optics, 2013, 52(1): A201-A209.

[10] Wang Y, Sang X Z, Chen Z D, et al. Real-time photorealistic computer-generated holograms based on backward ray tracing and wavefront recording planes[J]. Optics Communications, 2018, 429: 12-17.

[11] Parker S G, Bigler J, Dietrich A, et al. OptiX: a general purpose ray tracing engine [J]. ACM Transactions on Graphics, 2010, 29(4): 66.

[12] Ning R, Li C G, Lou Y L, et al. Matlab fast algorithm of computer generated holograms based on multi-graphic processing unit [J]. Laser & Optoelectronics Progress, 2019, 56(5): 050901.
宁冉, 李重光, 楼宇丽, 等. 基于多图像处理单元的Matlab计算全息图快速算法[J]. 激光与光电子学进展, 2019, 56(5): 050901.