第 35 卷第 10 期
2023 年 10 月

强 激 光 与 粒 子 束
HIGH POWER LASER AND PARTICLE BEAMS

Vol. 35, No. 10
Oct., 2023

·强激光物理与技术·

一种激光束照射复杂目标图像实时生成新方法*

张玉双[1]， 王 锐[2]， 苏 华[1]， 张飞舟[1]， 谢晓钢[1]

（1. 北京应用物理与计算数学研究所，北京 100094； 2. 北京神秘谷科技有限公司，北京 100086）

摘 要： 针对激光雷达探测中复杂目标姿态获取困难、目标与光斑真实耦合情况难获取等问题，提出了一种基于GPU编程的激光束照射复杂目标图像实时生成新方法。基于现代图形硬件的GPU编程技术和帧缓存对象特性，该算法以每个平面光源矩阵为观察者，在光源空间坐标系中渲染当前场景并将渲染结果记录到内存纹理中，然后在世界坐标中将光源观察到的结果复原和映射到模型上，可以实现实时映射渲染。采用深度缓存原理、纹理映射原理和OSG文件读写插件，即可正确获取模型每个三角面片顶点上的光源辐照度、顶点位置及面片法线等信息。经测试，该算法普适性强、能够读取多种格式三维文件，适应于均匀或非均匀面光源，对系统图形硬件的要求很低，能够满足两个面光源准实时性计算需求，可以准实时得到模型被照射面片所属部件、被照射三角面片顶点、法线信息以及三角面片顶点接收到的辐照强度值。该算法可为激光照明、识别探测等提供参考和依据。

关 键 词： 帧缓存对象；深度缓存；面光源；复杂目标；实时映射计算

中图分类号： TP391.9 **文献标志码：** A **doi**: 10.11884/HPLPB202335.230063

A new approach for real-time imaging from laser beam to complex targets

Zhang Yushuang[1]， Wang Rui[2]， Su Hua[1]， Zhang Feizhou[1]， Xie Xiaogang[1]

（1. *Institute of Applied Physics and Computational Mathematics*, *Beijing* 100094, *China*;
2. *Beijing Mana VR Co., Ltd* , *Beijing* 100094, *China*）

Abstract: Target detection by lidar is challenging due to the difficulty in obtaining the complex attitude of targets and capturing the real coincidence between target and facula. To address this problem, in this paper, a real-time mapping method of laser beam to complex targets based on GPU programming is proposed. By taking advantages of modern graphics hardware with respect to GPU programming technology and frame buffer object merit, the proposed approach takes each surface light source matrix as the observer, renders the current scene in the light source spatial coordinate system, and records the rendering results into the memory texture. To realize real-time mapping and rendering, the results observed by the light source in the world coordinates are restored and mapped to the model. Based on deep cache principle of Zbuffer and texture mapping principle, the model information (e.g., light source irradiance, vertex position and patch normal on the vertex of each triangular patch) can be correctly obtained with virtue OSG file reading-writing plug-in. Extensive experiments demonstrate the strong universality of the proposed algorithm. It is powerful in reading three-dimension files of various formats and is suitable for uniform or non-uniform surface light sources. It meets the quasi real-time computational requirements of two surface light sources with low requirements on system graphics hardware. Various model information could be acquired in quasi real-time, e.g., the components of the illuminated surface piece, the vertices of the illuminated triangular surface, the normal information and the irradiation intensity received by the vertex of the triangular patch. The algorithm is novel in providing reference and basis for laser illumination, recognition and detection.

Key words: frame buffer object, deep cache, surface light source, complex target, real-time mapping

* **收稿日期:**2023-03-28； **修订日期:**2023-07-18
基金项目:国家自然科学基金项目（52103073）
联系方式:张玉双，moshi2@126.com。
通信作者:王 锐，wangray84@foxmail.com。

激光成像中的目标散射特性是人们非常感兴趣的问题。在目标识别与反识别、隐身与反隐身、跟踪与制导中，目标散射特性研究是一个非常重要的研究方向。目前常用的光学散射截面计算方法分为理论计算和数值法：理论计算仅限于简单几何体散射特性计算；数值法多将三维模型拆分为小面元，计算面元散射特性后，合成目标整体散射截面，又称为面元网格法[1]。基于面元网格法，徐灿等[2]和许兴星等[3]通过 OpenGL 技术实现面元遮挡判断，假定光源为均匀光源（太阳光和地面辐射光），采用双向反射分布函数（BRDF）模型[4]，在普通计算机上实时获取空间目标光学横截面积，该方法只能读取 3ds 模型文件，无法读取纹理信息，且无法精确获取每个三角面片上辐照强度信息。韩意等[5]采用 BRDF 模型，假定光源为均匀光源，分别设置部件的散射特性，计算空间目标光学横截面积，该方法同样无法精确获取每个三角面片上辐照强度信息。针对任意构型目标，岳玉芳等[6]采用均匀光照明仿真图像进行光学散射截面数值计算，该方法无法模拟非均匀光束照明仿真图像，无法获取三角面片上辐照强度信息。激光束与复杂目标耦合计算，可以用无限细分的平行光线阵模拟面光源，采用光线追迹算法[7-8]，针对每一根光线与三维模型三角面片进行求交计算，从而获取三角面片上辐照强度值，然而，该类方法计算量大，并且需要平行光线阵细分度与模型建模细分度相匹配，否则会出现漏面现象。针对上述问题，本文提出了一种基于 GPU 编程的激光束照射复杂目标图像实时生成新方法，该算法借鉴了计算机图形学中纹理映射思想实现，将二维光斑作为纹理图像映射到三维目标外形上，实现了光束照射目标的实时渲染效果，纹理图像计算在 GPU 端实现。在 CPU 端，通过遍历三维目标顶点数据，查找对应纹理值，获取顶点接收到的辐照强度值。

1 计算方法

1.1 算法流程图

如图 1 所示，本文方法流程如下：构建场景，创建从相机摄像机渲染场景时所使用的着色器；创建光源矩阵对象，在创建光源矩阵对象时，同时创建辐照度数据数组；创建纹理对象，该纹理对象接收着色器渲染的结果并传递给辐照度结果数据数组；创建 RTT 相机[9]，将光源平面视角的场景渲染到辐照度计算结果数据数组中，调用光照空间片元着色器；光照空间渲染，在着色器中计算屏幕坐标对应的辐照度值；将光照空间渲染结果保存到纹理中；在模型空间中完成遮挡裁剪，完成模型带辐射度值的渲染显示；最后完成辐照强度值输出。

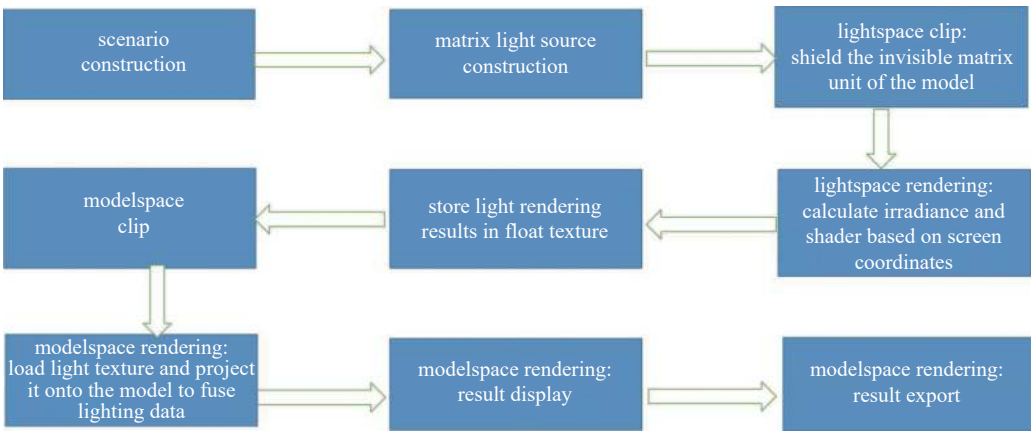

Fig. 1 Flow chart of proposed algorithm

图 1 算法流程图

为渲染场景，需要首先构建模型空间顶点着色器和片元着色器，以及光照空间顶点着色器和片元着色器[10]。光照空间片元着色器完成了在光照空间下，提前屏蔽看不到的模型面片。模型空间中顶点着色器，完成光照数据的恢复和坐标变换计算。模型空间中片元着色器完成屏幕坐标对应的辐照度计算。

1.2 光束源数据结构的建立

当激光束接近目标包围球范围时，此后的光束可以假定为平行光，可以用面光源来模拟。用有一定指向的矩形来描述面光源，二维数组描述光源二维分布。构建光照矩阵的数据结构见表 1。

整个矩形在空间的位置、旋转姿态以及尺寸大小可以通过 lightPos、lightDir、lightUp 和 lightSize 来进行调节。光源二维分布为 256×256 维的 float 类型数组（数组维数可以根据实际需求进行修改），每个数字代表该矩阵所在位置的光辐射强度。

为了满足设置多个光源数据的需求，通过 lightID 来区分各个光源的属性值。

表 1 光源类输入输出约定

Table 1 Description of input and output about light source class

No.	name	description	data type
1	lightPos	light position	input/vec3
2	lightDir	light direction	input/vec3
3	lightUp	light waggling direction	Input/vec3
4	lightSize	light size	input/vec2
5	lightDataW	width of light data	Input/int
6	lightDataH	height of light data	input/int
7	lightID	light ID	input/int
8	lightMaxAttenuation	the farthest distance the light source can reach	input/double
9	lightData	light data	input/W*H
10	lightSpaceImage	rendering results in the light source spatial coordinate system	output/numerical matrix

1.3 平行面光源照射模拟

如果将光源平面视为实时图形场景中的一个摄像机对象，那么它相当于一个正交投影类型的摄像机，如图 2 所示，该正交投影体可以根据表 1 中的光源 lightSize 参数和 lightMaxAttenuation 参数来构建，即

$$\boldsymbol{M}_{\text{proj}}=\begin{bmatrix}\dfrac{2}{r-l} & 0 & 0 & -\dfrac{r+l}{r-l}\\ 0 & \dfrac{2}{t-b} & 0 & -\dfrac{t+b}{t-b}\\ 0 & 0 & -\dfrac{2}{f-n} & -\dfrac{f+n}{f-n}\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{1}$$

式中：l 为 −lightSize.x*0.5；b 为 −lightSize.y*0.5；r 为 lightSize.x*0.5；t 为 lightSize.y*0.5；n 设置为 0；而 f 设置为 lightMaxAttenuation。

光源摄像机的观察矩阵设置为

$$\boldsymbol{M}_{\text{view}}=\begin{bmatrix}r_x & r_y & r_z & p_x\\ u_x & u_y & u_z & p_y\\ d_x & d_y & d_z & p_z\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{2}$$

式中：u 为上表中的 lightUp；d 为 lightDir；p 为 lightPos；r 是 d 和 u 叉积后的结果。

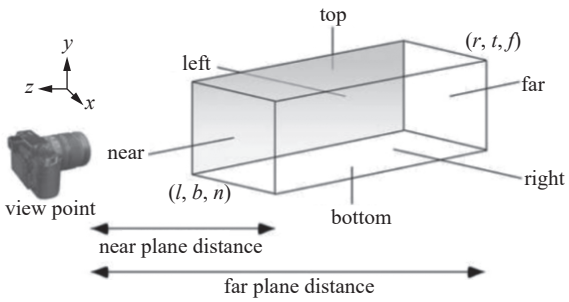

Fig. 2 Illustration of orthographic projection

图 2 正交投影示意图

图 3 中左下角小图显示了此时该摄像机的渲染效果，该图记录了以光源摄像机为主视角时渲染得到的场景图像。如果将该图像直接映射到世界坐标系的主相机画面中，则如大图所示，其记录了从光源平面发射的平行射线投射到场景模型之后，距离光源平面最近的交集点的信息。

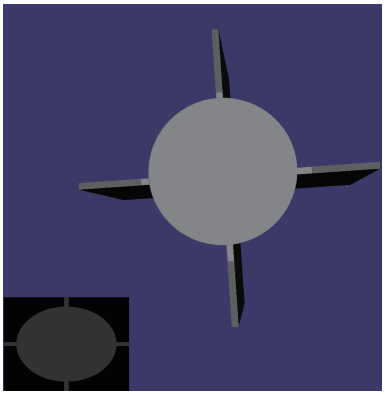

Fig. 3 Simulation of light irradiation on parallel planes

图 3 平行面光源照射模拟

1.4 渲染光照数据到纹理

我们通过帧缓存对象（FBO）的方式，将光源摄像机渲染场景的结果保存到内存中的一幅图像中，该图像的分辨率设置为 1024×1024，该值越大表示算法精度越高，但是也需要考虑图形硬件的实际承载能力。

为了提高计算精度，将 FBO 图像的像素格式设置为 GL_FLOAT，纹理的内部存储格式设置为 GL_RGBA32F_ARB，即采用 32 位浮点数来保存结果数据。

使用 OpenGL 着色语言完成像素值获取（光照空间片元

着色器)。核心代码如下:

```
vec2 lightUV=vec2(gl_FragCoord.x/lightDataSize.x,gl_FragCoord.y/lightDataSize.y);
vec4 lightColor=texture(lightDataTexture, lightUV);
gl_FragColor=vec4(lightColor.xyz,gl_FragCoord.z/gl_FragCoord.w);
```

其中: lightUV 表示光源平面上的某个射线起点位置; 而 lightDataTexture 来自于存储了光源子块数据强度的 lightData 参数; lightColor 记录的为当前射线所对应的光照强度值; 而 gl_FragCoord.z 表示的是光源空间的数据深度值。根据 Zbuffer 缓存原理, 该值为射线与模型上最近交集点的距离值。将 lightColor 与该距离值共同保存 lightSpaceImage 中。

1.5 光照数据的恢复和变换

从光源空间渲染得到的有效图元数据, 实际上就是模型上可以接受到光照的图元。将 lightSpaceImage 中的光照数据恢复到世界坐标系中, 并映射到三维模型表面, 即可得到模型表面三角面片上辐照强度结果。

假设已知场景局部坐标系下三维模型上的一点 P, 在光源空间坐标系下 P 点的映射点 P'可以通过下面的公式求得

$$\boldsymbol{P}' = \boldsymbol{M}_{\text{lightP}}\boldsymbol{M}_{\text{lightV}}\boldsymbol{M}_{\text{world}}\boldsymbol{P} \tag{3}$$

式中: $\boldsymbol{M}_{\text{world}}$ 表示将 P 点变换到世界坐标系的变换矩阵; $\boldsymbol{M}_{\text{lightV}}$ 表示光源虚拟相机的观察矩阵; $\boldsymbol{M}_{\text{lightP}}$ 表示光源虚拟相机的投影矩阵; 结果值 P'的 (x,y) 值对应于 lightSpaceImage 图像中的归一化坐标点, 取值范围为 [−1,1], 如果 P'_{x}或者 P'_{y}的值小于−1 或者大于 1, 说明三维模型上的当前点不能映射到光源平面上, 可以直接丢弃。将 $(P'_{\text{x}},P'_{\text{y}})$ 重新归一化到 [0,1] 区间, 然后作为纹理坐标去采样 lightSpaceImage 纹理中的纹素值, 得到的结果即为光源平面上该点处射线与模型上最近交集点的距离值和辐照强度。

利用纹理坐标查找对应纹理值, 实际上利用了 OpenGL 纹理插值算法, 从而避免了光线追迹算法[7-8] 中模型建模细分度与光线线阵细分度不匹配导致的漏面问题(如球体模型, 球体两端三角面片非常细, 采用光线追迹求交, 几乎无法得到正确的计算结果)。

1.6 遮挡裁剪

对于模型上一点 P, 它在光源平面上的投影点只有一个(不考虑透射情况), 即 P', 但是, 模型上可能有不止一个点的投影点计算结果为 P'。图 4 所示为某个复杂 3D 模型的剖面, 它的表面上至少存在 4 个点可以投影到光源平面上的 P'上。然而, 显然只有 P_0 会真正接收到辐照, 其他点都被遮挡。对于屏幕上的每个像素点, 深度缓存会记录场景中物体与视点在这个像素上的距离信息。利用深度缓存原理, 可在着色语言中判断和处理这一情形。

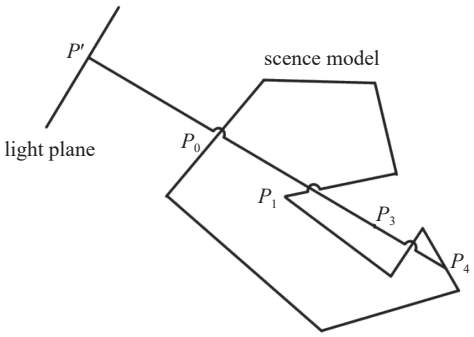

Fig. 4 Illustration of multipoint projection

图 4 多点投影示意图

之前的公式所求取的 P'点, 其 (x, y) 值对应了光源视角渲染结果图像的纹理坐标值, 而它的 z 值则表示局部坐标中的 P 点与光源平面的垂直距离。利用这一特性, 我们可以将 P'_z与 lightSpaceImage 中存储的距离值进行比较: 如果两者非常接近, 说明当前模型上的 P 点正是光源摄像机视角中渲染的同一点, 也就是应当接收到辐照的模型表面点; 如果 P'_z显著大于 lightSpaceImage 所记录的距离值, 那么这一点必然被其他的三角面所遮挡, 可以忽略不计。

使用 OpenGL 着色语言实现上述过程的代码片段如下(模型空间片元着色器):

```
//将当前点坐标归一化到 [−1,1] 区间
vec3 lightProjCoord = lightSpacePos.xyz/lightSpacePos.w;
//将当前点坐标归一化到 [0,1] 区间
lightProjCoord = lightProjCoord * 0.5 + 0.5;
float x = 0.0, y = 0.0;
vec4 lightColor = vec4(0.0);
for (y = −1.5; y < 2.0; y += 1.0)
```

```
for (x = −1.5; x ＜ 2.0; x += 1.0)
lightColor += texture(lightTexture, lightProjCoord.xy + vec2(x, y) * 0.000 5);
lightColor = lightColor * 0.062 5;
//判断是否照射，如被照射为 1，如未被照射为 0
float shadowing = (lightColor.w ＜ (lightProjCoord.z − 0.000 1)) ? 0.0 : 1.0;
//其中 intensity 为光照射方向矢量与面片法线点乘值，考虑光束投影效果。
lightColor.xyz = lightColor.xyz * (intensity * shadowing);
return lightColor;
//intensity 计算求解代码如下（模型空间片元着色器）
vec4 n = normalize(osg_ViewMatrixInverse * vec4(eyeNormal.xyz, 0.0f));
float intensity = min(-dot(n.xyz, normalize(globalLightDir)), 1.0);
//eyeNormal 计算代码如下（模型空间顶点着色器）
eyeNormal = normalize(gl_NormalMatrix * gl_Normal);
```

其中：lightSpacePos 为我们通过矩阵级联运算得到的 P'点，lightTexture 为之前 FBO 渲染得到的 lightSpaceImage 对应的纹理。代码片段最终返回的结果即为当前片元接收辐照度的结果值，它可能是 lightSpaceImage 中记录的数值，也可能直接是 0（表示没有收到辐照）。

1.7 辐射照度值输出

来自 lightSpaceImage 的图像数据中记录了模型被辐照时表面的辐照度值。但是，通过 GLSL 着色语言的形式，只能渲染得到当前视角下的模型受辐照形态，即只能得到渲染效果图，无法真正输出最终模型每个三角面片上的辐照度值。因此，本文选择在 CPU 端完成辐照数值的输出。

为此，需要首先将 lightSpaceImage 的数据通过 ReadPixels 方法从 GPU 显存读取到系统内存中，并收集模型的每一个三角面和三角面上的每个顶点数据，进行遍历：

（1）将顶点 P 投影到光源坐标系中，得到一个 2D 坐标 P'以及一个深度值（距离值）d。其中，P'的取值范围不应该超过 lightSpaceImage 的分辨率，即 [0,1 024] 的区间范围，如果超过该范围，说明该点 P 在光照范围之外，可以直接抛弃（设置其光照结果为默认值 0）。

（2）可以用 P'作为指针，从 lightSpaceImage 的内存数据中获取对应位置的浮点数值，也就是该位置光源射线与模型上最近点的距离值 c。

（3）如果当前点所在三角面的面法线和光源方向是同侧的（法线和光源方向矢量的点积结果大于 0），则该点应该被提前抛弃（它是背对光源的，设置其光照结果为默认值 0）。

（4）如果这个距离值 c 接近于 d，则认为当前顶点就是接收光源辐照的最近点，纹理对应的当前的亮度值/颜色值就是模型上这一点的颜色值，即激光辐照度值。

本文采用 OSG 的 FaceData 类遍历模型所有面片，将超出光源范围、被其他面片遮挡的顶点对应的激光辐照度值设为零，否则，设置为该点纹理对应的值，并将三角面片对应的顶点信息、法线信息和辐照度值保存到文件中。

2 应用实例

2.1 实验环境搭建

本文使用 OpenSceneGraph 3.6.5 版本来搭建实验环境，配置一般商用 PC 和显卡。

2.2 测试结果

本文用一个圆柱体加两个球体和两个棱锥体作为测试模型，该模型总三角面数约为 5 000 面，其建模截图如图 5 所示，从截图中可以看到球体两端处和锥体顶点处三角面片剖分非常细。

如果基于传统的射线与模型求交的方式采用单束光输入，大约需要花费 100 s 的时间；采用三束光输入，大约需要花费 350 s 的时间，运算过程中系统无法同时进行其他工作，无法满足实时计算的要求；如果采用本文所述算法，则单束光的计算时间约为 35 ms，系统实时运行的帧速率为 30 帧·s^{-1}，三束光同时叠加计算的时间约为 66 ms，对应的系统实时运行帧速率为 15 帧·s^{-1}。

本文所述算法的实时运行界面如图 6 所示（实时渲染效果为太阳光、环境光和激光三种光源共同作用下的结

果。本算法不考虑大气湍流对目标成像过程的影响，但激光传输到目标处二维光斑分布考虑了大气湍流影响[11])。

实时计算得到的光束叠加结果可以通过“渲染到纹理”的方式，从 GPU 端传送到 CPU 端，并保存到文件中。考虑到 GPU 端计算得到的辐照强度值为 32 位的浮点数值，且辐照范围超过 [0, 1] 的数值区间，不宜采用一般的 OpenGL 纹理进行存储。本文使用了浮点纹理格式来进行相关数据的保存，即：

```
tex->setDataType(GL_FLOAT);
tex->setPixelFormat(GL_RGBA);
tex->setInternalFormat(GL_RGBA32F);
```

这里的 tex 即是 OSG 引擎代码中用于存储纹理的对象。具体存储的部分三角面片顶点对应辐照度数据见表 2。

图 7 为被辐照度三角面顶点坐标三维点云图，可以看到其分布区域和变化趋势，并且与传统算法计算得到的结果相符。

测试结果表明：光束与模型映射关系和计算结果正确；支持多束光、非均匀光源情况；针对球体顶点处网格细分过细情况，算法计算正确；当模型进一步细分时，基本不影响计算效率。算法对硬件要求不高，适合一般 PC 机；算法采用 OSG 引擎进行开发，具有跨平台性。

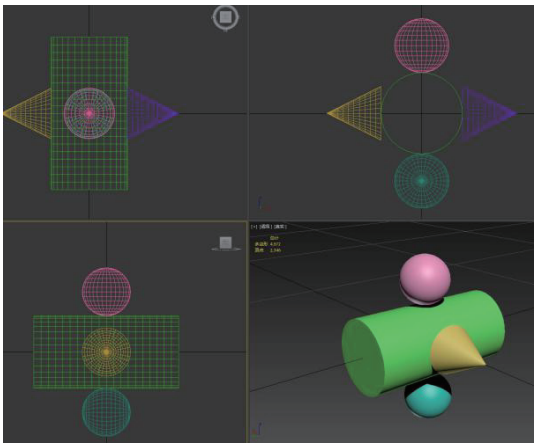

Fig. 5　Screenshot of test model

图 5　测试模型截图

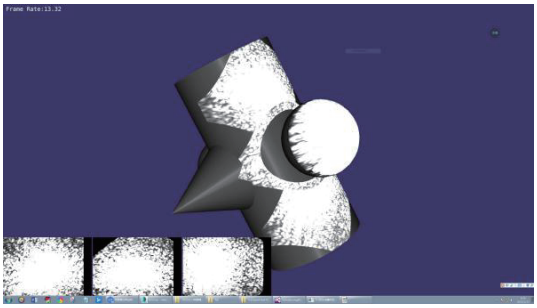

Fig. 6　Screenshot of three-beam mapping display

图 6　三束光映射显示截图

表 2　三角面片上辐照度信息

Table 2　Irradiance information on triangular patches

part name	x/m	y/m	z/m	x/m	y/m	z/m	x/m	y/m	z/m	x/m	y/m	z/m	energy/ ($\mu W\cdot m^{-2}$)
	first vertex			second vertex			third vertex			identity normal of a triangle			
cylinder1	0.1736	0.12	0.9848	0.0872	0.12	0.9962	0.0872	0.00	0.9962	0.1305	0.00	0.9914	132867.00
cylinder1	0.0872	0.12	0.9962	0.0000	0.00	1.0000	0.0872	0.00	0.9962	0.0436	0.00	0.9990	132867.00
box4	0.0500	0.20	2.0800	0.0500	0.20	2.2000	0.0500	0.00	2.2000	1.0000	0.00	0.0000	111684.00
box4	0.0000	0.20	2.2000	0.0000	0.00	2.2000	0.0500	0.00	2.2000	0.0000	0.00	1.0000	111684.00
box4	−0.0500	0.20	2.2000	−0.0500	0.00	2.2000	0.0000	0.00	2.2000	0.0000	0.00	1.0000	16894.30
cylinder1	0.0872	0.48	0.9962	0.0000	0.48	1.0000	0.0000	0.36	1.0000	0.0436	0.00	0.9990	14565.00
cylinder1	0.0000	0.48	1.0000	−0.0872	0.36	0.9962	0.0000	0.36	1.0000	−0.0436	0.00	0.9990	14565.00
box4	−0.0500	0.20	2.0800	−0.0500	0.00	2.0800	−0.0500	0.00	2.2000	−1.0000	0.00	0.0000	14203.50
box4	0.0000	−0.20	2.2000	0.0000	0.00	2.2000	−0.0500	0.00	2.2000	0.0000	0.00	1.0000	14203.50
box4	−0.0500	0.40	2.0800	−0.0500	0.20	2.0800	−0.0500	0.20	2.2000	−1.0000	0.00	0.0000	14148.40

3　小　结

本文提出了一种基于 GPU 编程的激光束照射复杂目标图像实时生成新方法。测试结果证明：该方法可以读取多种格式的三维文件；适应于均匀或非均匀面光源及多束光情况；对系统图形硬件要求低；能够满足多个面光源准实时性计算需求；能够获取模型被照射面片所属部件、被照射三角面片顶点、法线信息以及三角面片顶点接收到的辐照强度值。

后续可以开展的研究工作包括：(1)目前光照模型采用 phong 模型，漫反射率和镜面发射率取相同值，无法描述材质的真实效果，下一步工作拟采用五参数光照模型[12-13]，将模型部件赋予不同的材质，提高光散射截面计算精度；(2)目前每个光束按照 OSG 文件输出格式独立生成一个文件，被照射的点有辐照度值，未被照射的点该值为 0，下一步工作将多个光束累积生成一个输出文件，并只输出光束照射到的三角面片信息。

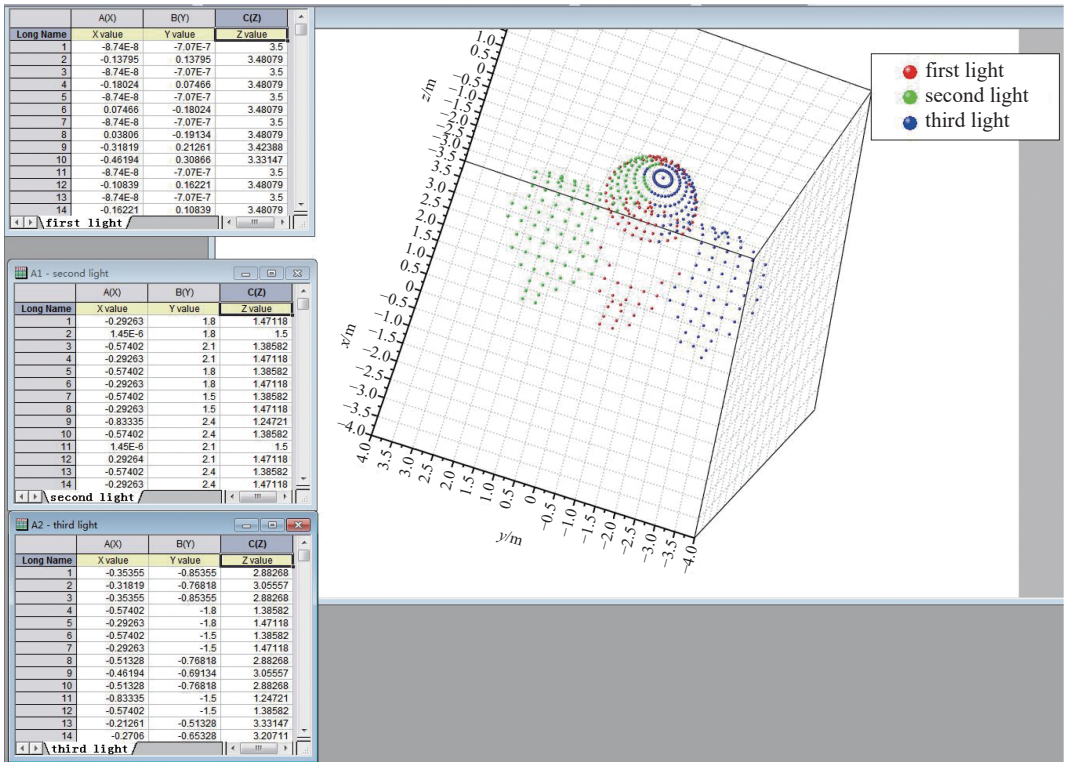

Fig. 7 Coordinate point cloud of illuminated triangular patch

图 7 被照射的三角面片坐标点云

参考文献：

[1] 鲍文卓, 丛明煜, 张伟, 等. 基于面元网格化的空间目标光学特性计算方法[J]. 哈尔滨工业大学学报, 2010, 42(5): 710-715. (Bao Wenzhuo, Cong Mingyu, Zhang Wei, et al. An optical characteristics calculating method based on surface mesh-creation for space targets[J]. Journal of Harbin Institute of Technology, 2010, 42(5): 710-715)

[2] 徐灿, 张雅声, 李鹏, 等. 基于OpenGL拾取技术的空间目标光学横截面积计算[J]. 光学学报, 2017, 37: 0720001. (Xu Can, Zhang Yasheng, Li Peng, et al. Calculation of optical cross section areas of spatial objects based on OpenGL picking technique[J]. Acta Optica Sinica, 2017, 37: 0720001)

[3] 许兴星, 丁雷. 基于OpenGL的星载可见光相机成像仿真系统[J]. 红外, 2017, 38(7): 15-21. (Xu Xingxing, Ding Lei. Spaceborne visible light camera imaging simulation system based on OpenGL[J]. Infrared, 2017, 38(7): 15-21)

[4] 孙华燕, 郭惠超, 范有臣, 等. 激光主动成像中的BRDF特性[J]. 红外与激光工程, 2017, 46: S106004. (Sun Huayan, Guo Huichao, Fan Youchen, et al. Analysis of BRDF character in active laser imaging[J]. Infrared and Laser Engineering, 2017, 46: S106004)

[5] 韩意, 陈明, 谢剑锋, 等. 地基光学望远镜对目标飞行器成像的仿真与验证[J]. 红外与激光工程, 2019, 48: 1214002. (Han Yi, Chen Ming, Xie Jianfeng, et al. Simulation & validation of ground-based optical-telescope imaging on target craft[J]. Infrared and Laser Engineering, 2019, 48: 1214002)

[6] Yue Yufang, Zang Feizhou, Zou Kai, et al. Numerical simulation on optical cross section of complex targets[J]. Chinese Journal of Computational Physics, 2017, 34(1): 109-118.

[7] 张发强, 张维光, 万文博. 基于光线追迹的红外探测光学系统杂散辐射研究[J]. 红外与激光工程, 2019, 48: 090406. (Zhang Faqiang, Zhang Weiguang, Wan Wenbo. Research on stray radiation of infrared detection optical system based on ray-tracing[J]. Infrared and Laser Engineering, 2019, 48: 090406)

[8] 汪喆. 基于光线跟踪算法的空间目标光散射特性研究[D]. 西安: 西安电子科技大学, 2018. (Wang Zhe. Study of the light scattering characteristics from the spatial target based on ray tracing[D]. Xi'an: Xidian University, 2018)

[9] 王锐, 钱学雷. OpenSceneGraph三维渲染引擎设计与实践[M]. 北京: 清华大学出版社, 2009. (Wang Rui, Qian Xuelei, OpenSceneGraph 3D rendering engine design and practice [M]. Beijing: Tsinghua University Press, 2009)

[10] 陈继选, 王毅刚. 基于OSG的GLSL着色器编辑环境[J]. 计算机系统应用, 2011, 20(3): 153-156. (Chen Jixuan, Wang Yigang. GLSL shader editing environment based on OSG[J]. Computer Systems & Applications, 2011, 20(3): 153-156)

[11] 张建柱, 李有宽. 大气湍流对部分相干平顶高斯光束的影响[J]. 强激光与粒子束, 2005, 17(2): 197-202. (Zhang Jianzhu, Li Youkuan. Atmospheric turbulence effects on partially coherent flat-topped Gaussian beam[J]. High Power Laser and Particle Beams, 2005, 17(2): 197-202)

[12] 刘程浩, 李智, 徐灿, 等. 针对空间目标常用材质菲涅耳反射现象的改进Phong模型[J]. 激光与光电子学进展, 2017, 54: 102901. (Liu Chenghao, Li Zhi, Xu Can, et al. A modified Phong model for Fresnel reflection phenomenon of commonly used materials for space targets[J]. Laser & Optoelectronics Progress, 2017, 54: 102901)

[13] 田琪琛, 李智, 徐灿, 等. 基于实验测量与OCS仿真计算的卫星光学散射特性对比验证[J]. 光子学报, 2018, 47: 0129004. (Tian Qichen, Li Zhi, Xu Can, et al. Comparison and verification of satellite optical scattering characteristics based on experimental measurements and OCS simulation[J]. Acta Photonica Sinica, 2018, 47: 0129004)