深度剥离与 GPU 结合的近似软影算法

孙明彦[1),2)]，吕伟伟[1),2)]，刘学慧[1)]，吴恩华[1),3)]
[1)]（中国科学院软件研究所计算机科学国家重点实验室，北京 100190）
[2)]（中国科学院研究生院，北京 100049）
[3)]（澳门大学科学技术学院电脑与资讯科学系，澳门）

摘 要：针对基于阴影图算法扩展的一些近似软影算法中存在的只考虑外半影区而导致的本影区过多估计的问题，提出了一种深度剥离与 GPU 结合的近似软影实时绘制算法。算法利用 GPU 的几何着色器来提取场景物体的轮廓边并生成内半影和外半影图元，进而得到整个内外半影颜色图和深度图，最终阴影绘制的时候通过参考阴影图和内外半影图来确定每个可见像素的明暗值，从而得到比以往算法较真实的绘制效果，算法完全在 GPU 中实现。实验结果表明，对相对不复杂的场景，该算法可以生成较真实的软影效果，且绘制帧率完全达到实时。

关键词：GPU；软影；深度剥离；几何着色器
中图法分类号：TP391.41 **文献标志码**：A **文章编号**：1006-8961(2010)09-1391-07

Approximate soft shadow algorithm based on depth peeling and GPU

SUN Mingyan[1),2)], LÜ Weiwei[1),2)], LIU Xuehui[1)], WU Enhua[1),3)]
[1)]（*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190）
[2)]（*Graduate University of Chinese Academy of Sciences, Beijing* 100049）
[3)]（*Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macao*）

Abstract: To deal of umbra overestimation in some current soft shadow algorithms considering only the outer penumbra effects extended from traditional shadow mapping, a real-time approximating algorithm for plausible soft shadows based on depth peeling is proposed. The algorithm extracts the scene silhouette edges in geometry shader and generates the inner and outer penumbra primitives respectively through the silhouette edges. In the final rendering pass, the shading of each pixel is decided by referring to the shadow map and penumbra map, and plausible soft shadow effect which is more realistic than the results of previous methods is finally simulated. This algorithm runs entirely on GPU. Several experimental results show that, this algorithm generates more plausible soft shadows for relatively uncomplicated scenes than other methods and has real-time rendering speed.

Keywords: GPU; soft shadow; depth peeling; geometry shader

0 引 言

阴影是非常重要的场景元素，它可以很好地提供场景中物体之间的空间关系，对增加场景的真实感有着非常重要的意义[1]。经典的阴影绘制算法主要包括阴影图[2]和阴影体算法[3]两大类，它们是用来模拟点光源的硬阴影效果，光源投射场景所生成的阴影区和非阴影区的过渡非常明显，而真实世界中是不存在绝对的点光源的，所以如何实时绘制

基金项目：国家自然科学基金项目(60573155)。
收稿日期：2009-04-07；**改回日期**：2009-05-19
第一作者简介：孙明彦(1983—)，男。2009 年于中国科学院软件研究所获计算机应用技术硕士学位。主要研究方向为计算机图形学。E-mail: sunmy@ios.ac.cn。

面光源所产生的软影效果已成为计算机图形学研究的一个重要课题。

由于阴影图算法有着实现简单以及与场景几何复杂度无关等优点，所以非常适合 GPU 硬件实现[4]。而绘制软影最终需要解决的问题是位于从光源视点所看到的轮廓边对应部分附近区域的亮度过渡问题，所以许多学者通过对阴影图算法进行扩展来绘制软影。

基于阴影图算法扩展的物理正确软影算法[5-9]一般需要考虑面光源的具体形状大小，然后通过阴影图提供的信息进行反投影求得与光源的交集部分占光源面积的比例。而基于阴影图的近似软影算法大都通过考虑轮廓边或给轮廓边增加一些信息来近似估计半影区的亮度，并不考虑面光源的具体形状大小。本文提出的也是一种近似软影算法。

Wyman 等人首先利用 CPU 提取从光源视点所能看到的场景物体的轮廓边，然后对每个轮廓边顶点生成一个锥体，通过将相邻的锥体连接形成新的面元来构建半影区图元，并绘制得到半影图，最终通过深度图和半影图结合来确定每个可见像素的明暗值[10]。

Chan 等人提出的算法和文献[10]类似，不同的是在确定一条轮廓边后向外生成一个矩形图元，将这些新生成的图元绘制得到半影图[11]。这两种算法都是仅仅考虑外半影效果，当面光源的大小发生变化时，由于是根据场景轮廓确定出本影区，所以本影区的大小并不随着变化。

Cai 等人针对半影区估计不足的问题，利用 CPU 提取物体轮廓边，并在轮廓边上同时向内向外构建半影图元，然后通过分成多层的方式，将内部和外部半影图元分别投影到各层得到内部和外部两个多层半影图，再通过这些多层图计算出屏幕空间的亮度缓存信息，最终在绘制的时候通过结合亮度缓存信息来提供更精确的软影效果[12]。

随着 GPU 的不断发展，其强大的并行加速计算能力，特别是几何着色器的出现，使得基于阴影图的软影算法中可以加入一些通过 GPU 实现的和场景几何信息相关的计算。

Lü 等人利用几何着色器，实时地提取轮廓边和生成半影图，提出一种针对文献[11]所提算法的完全基于 GPU 的实现思路[13]，但该算法也没有考虑内半影的效果。

针对文献[10-11]算法存在的本影区过估计的问题，本文算法在生成外半影图元的同时生成内半影图元，通过对内半影和外半影进行估计来生成较真实的软影效果。

算法的出发点和文献[12]类似，但和文献[12]不同的是本文利用 GPU 的几何着色器来提取轮廓边并生成内外半影图元，将不可见轮廓边生成的内外半影图元剔除掉，然后将余下的半影图元所代表的内外半影过渡信息同时绘制到一张半影颜色图中，最终绘制的时候通过各种比较得到可见像素的明暗值，从而得到相对以往算法较真实的软影效果。算法对一些关键问题给出了解决方案，并且完全在 GPU 中实现，可以达到实时。

1 算法思想

本文算法也是通过光源视点所能看到的轮廓边来生成近似的半影图元，文中内、外半影图元分别指根据轮廓边向内和向外生成的图元，内、外半影区分别指由光源向轮廓边投射光线把接收面半影区分成的两部分。

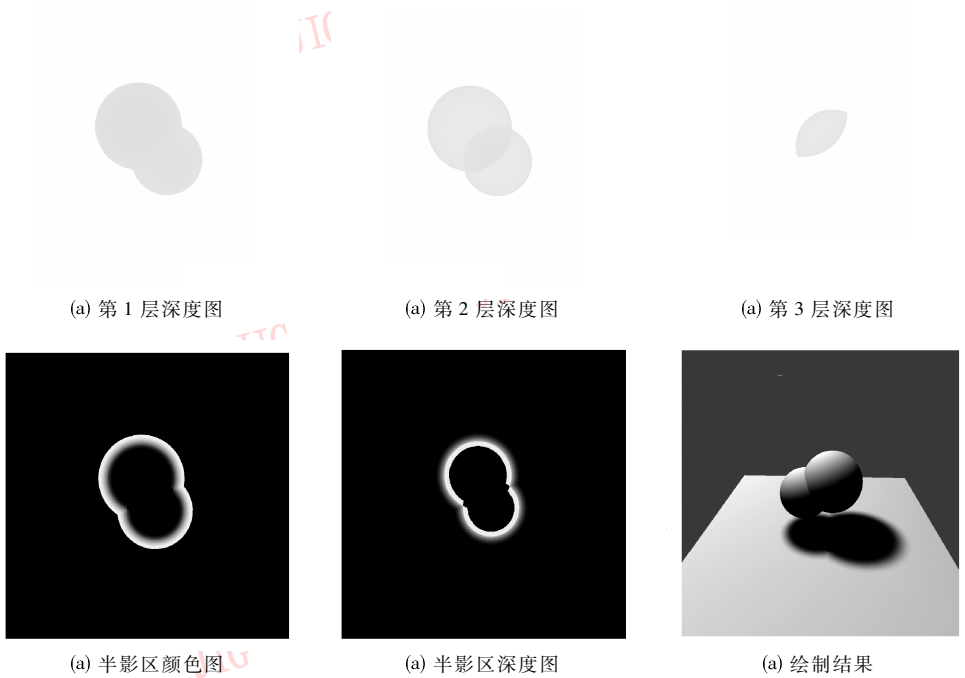
本文算法主要有 3 步，分别如下：

1）利用深度剥离的方法获得场景相对光源视点的最近 3 层深度值信息，将它们保存在 3 张深度纹理中。如图 1(a)—(c)所示，为了清晰起见，第 1 步和第 2 步都没有考虑接收平面。

深度剥离[14]是一种利用多遍绘制来对场景中的片元排序的方法。第 1 遍绘制得到距离视点最近的一层片元的深度，也就是常说的阴影图或第 1 层深度图；第 2 遍绘制的时候通过和第 1 遍绘制得到的最近片元深度进行比较从而得到距离视点次近的片元深度，称为第 2 层深度图；同理可以得到第 3 层乃至第 n 层深度图。文献[15]给出了该算法的 GPU 实现。

2）利用几何着色器提取相对光源视点的物体轮廓边，然后同时生成内外半影图元，并剔除不可见轮廓边所生成的半影图元，最终通过绘制得到半影图元部分的颜色图和深度图，分别如图 1(d)和图 1(e)所示。其中颜色图对应绘制后颜色缓存中的值，深度图对应深度缓存中的值。

3）通过以上得到的各种信息来最终绘制软影效果。最终结果如图 1(f)所示。



(a) 第 1 层深度图　(a) 第 2 层深度图　(a) 第 3 层深度图

(a) 半影区颜色图　(a) 半影区深度图　(a) 绘制结果

图 1　算法过程

Fig. 1　Algorithm procedure

2　算法的实现

2.1　半影颜色图和深度图的生成

这一步利用了 GPU 的几何着色器和片元着色器,其基本过程如图 2 所示。

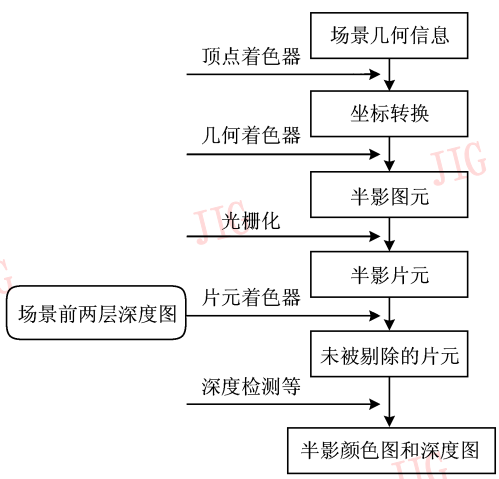


图 2　半影颜色图和深度图生成过程

Fig. 2　Flow of generating penumbra color and depth maps

2.1.1　半影图元的生成

提取轮廓边的方法和文献[13]一样,如果模型三角形网格上两个相邻的三角形相对光源视点朝向相反,则它们的公共边为轮廓边。假定找到了一条轮廓边 BC,如图 3 所示,$\triangle ABC$ 和 $\triangle BCD$ 为其所邻接的两个三角形。沿着该轮廓边顶点 B 和 C 的法线正方向和反方向创建 4 个新的顶点 B',C',B''和 C'',坐标分别为 $B' = B + r \times N_B$,$C' = C + r \times N_C$,$B'' = B - t \times N_B$,$C'' = C - t \times N_C$,其中参数 r 和 t 用来控制外半影图元和内半影图元的大小,近似模拟面光源的大小变化。在几何着色器中将输出 4 个三角形图元 $\triangle BCC'$,$\triangle C'B'B$、$\triangle CBB''$和 $\triangle B''C''C$,所有新生成的三角形图元就构成了半影图元集。

软影区一般是从暗到明的过渡过程,而希望半影颜色图会记录这种过渡信息,故指定 B''和 C''的颜

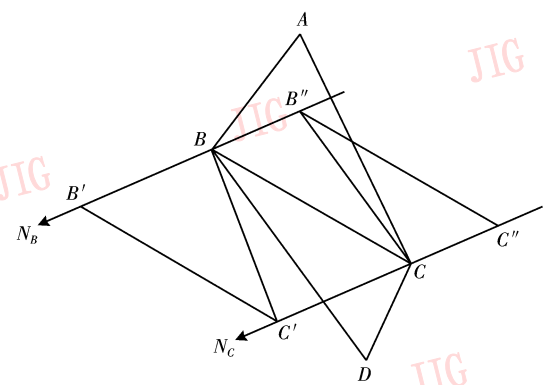


图 3　生成半影图元

Fig. 3　Generating penumbra primitives

色为(0,0,0),B 和 C 的颜色为(0.5,0,5,0,5),B' 和 C'的颜色为(1,1,1),然后利用硬件自动插值得到线性的过渡值,颜色缓存的 R 通道在本文算法一直保存过渡信息的值。

2.1.2 剔除多余的半影图元

当半影区图元被光栅化之后,需要对不可见轮廓边所生成的半影片元进行剔除。所谓片元,是指在 GPU 中,三角面片经过光栅化后处理的基本单位。设定当前半影片元在光源空间的深度为 *fragDepth*,颜色值为 *fragColor*,其对应的前两层深度图的值分别为 Z_1 和 Z_2,判断的基本思路如下:

1) 根据 *fragColor* 的 R 通道的值是位于区间 0 到 0.5 还是 0.5 到 1 来区分该片元是内半影片元还是外半影片元。

2) 对外半影片元而言,如果 $fragDepth < Z_1$,则说明该片元未被遮挡,需要保留并将 *fragColor* 的 B 通道的值赋为 0.5,否则被剔除掉。

3) 如图 4(a)所示,对一般场景而言,不可见轮廓边所生成的内半影图元的 *fragDepth* 应该大于 Z_2,如果 $fragDepth > Z_2$ 的话,则被剔除掉;否则保留,并将 *fragColor* 的 B 通道的值赋为 0。

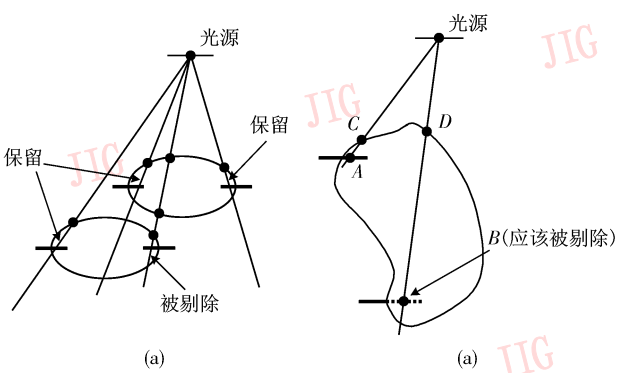


图 4 内半影图元的不同位置

Fig. 4 Different conditions of penumbra primitives

上面判断步骤对大多数片元都可以正确地进行剔除工作,但对一些相对复杂的模型会存在如图 4(b)的情况。假设 A 和 B 分别为图中所示位置轮廓边生成的两个内半影片元,其中 A 是由可见轮廓边所生成的,需要被保留,而 B 是由不可见轮廓边生成的,需要被剔除掉,但根据上面的基本判断,A 和 B 都没有被剔除掉。图中点 C 和 D 分别是片元 A 和 B 所对应的离光源视点最近的点,通过观察可以看出,大多这种情况的场景,B 和 D 之间的距离远大于 A 和 C 之间的距离,所以通过比较 *fragDepth* 和 $Z_1 + \beta$ 的大小来将 B 剔除掉,其中 β 是一个自己设定的值。

2.1.3 颜色和深度修正

1) 真实半影明暗度的过渡一般不是一个线性的变化过程,希望过渡值的变化如图 5 曲线所示。

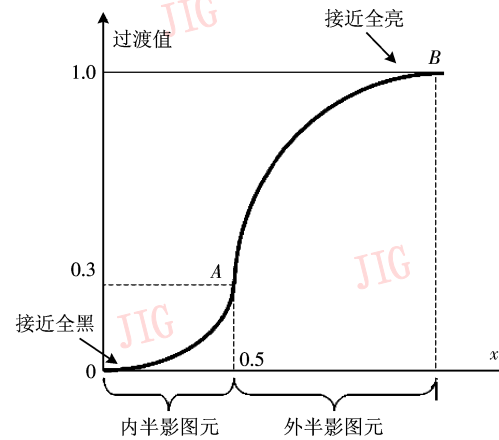
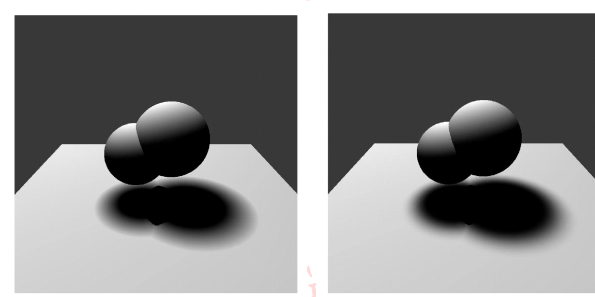


图 5 半影区过渡值的变化曲线

Fig. 5 Transition curve for penumbra color

使用正弦和余弦函数来模拟这种变化,设初始线性过渡值为 θ,修正后的过渡值为 ω,具体变换公式为:如果为内半影图元,则 $\omega = (1 - \cos(\theta \cdot \pi)) \times 0.3$;如果外半影图元,则 $\omega = \sin((\theta - 0.5) \cdot \pi) \times 0.7 + 0.3$。使用修正后的过渡值所生成的结果如图 6(b)所示,图 6(a)使用的是线性变化的过渡值,可以看出修正后的结果更加接近真实,在半影区边界过渡的地方都非常柔和。



(a) 过渡值线性变化 (b) 修正后

图 6 过渡值修正前后的结果比较

Fig. 6 Comparison before and after modification to transition value

2) 当场景中物体的深度层次较复杂的时候,会有多条轮廓边生成的且都被正确保留下来的半影片元需要同时绘制到半影图相同位置的情况发生,复杂的情况有以下两种可能,即半影图的某一位置可

能对应:(1) 一个内半影片元和多个外半影片元,此时内半影片元必然在相对光源视点最远处;(2) 多个外半影片元。

首先对于(2),这种情况本文和以前算法处理方式一样,希望选用相对光源视点最远的外半影片元作为最终的值。而对于(1),存在一点问题,因为无论选用什么方式作为半影图的值,在最后阴影绘制阶段,都会有一些像素对应不到正确的半影过渡信息。本文算法选用相对光源视点最远的外半影片元,不考虑最下方的内半影片元,考虑的依据是外半影较亮,内半影较暗,将极小部分内半影区误判为本影区对视觉造成的影响小于将外半影区域误判为光亮区的影响。

鉴于上述分析,按以下方法处理保留的半影片元的深度:如果是内半影片元,将其原始深度除以一个定值,使之变小,从而可以处理(1)这种情况;如果为外半影片元,采用文献[13]的半影区融合处理方法,使用伪深度值,可以很好的处理(2)这种情况。这种深度值的设定方法对其他情况没有任何影响,在最终阴影绘制阶段,并不需要深度图外半影部分的深度,而当需要内半影部分深度信息的时候,将深度图保存的值乘以上面那个定值即可还原真实内半影片元的深度值信息。

2.2　软影的生成

在最后的绘制阶段,将可见片元的坐标转换到光源坐标系得到该片元在光源空间的深度后,就可以根据已知的各种深度图信息来确定每个可见像素的亮度值。如图 7 所示,粗线表示的是物体轮廓边生成的半影图元区,数字标号表示该标号正下方那小块区域的阴影情况,在这里不考虑背向光源部分。

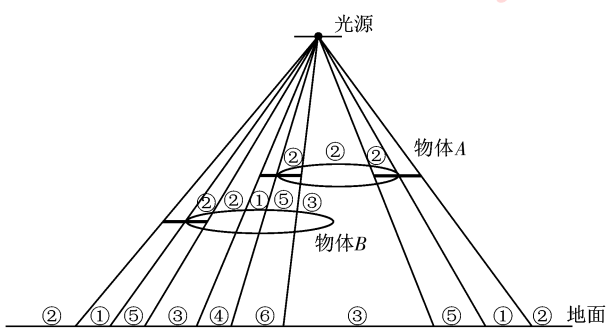


图 7　各种情况分析

Fig. 7　Analysis of complex condition

假定可见片元在光源空间的深度为 *fragDepth*,其对应的前三层深度图纹理的值分别为 Z_1,Z_2 和 Z_3,对应的半影颜色图和深度图的值为 *pColor* 和 *pDepth*,设 ε 为一个很小的浮点数。

1) 如果 $fragDepth < Z_1 + \varepsilon$ 的话,则该片元对于光源视点可见,要么受外半影图元影响(如①),要么完全被照亮(如②)。我们根据 *pColor*. *b* 的值来判断,等于 0.5 的话,则属于①;否则属于②。

2) 如果不满足 1)的情况,则说明该片元对于光源视点不可见。在这种情况下,要么该片元对应内半影图元且受其影响(如⑤),要么处于完全阴影中(如③④⑥)。首先根据 *pDepth* 的值判断出③这种情况,因为没有半影图元对应的半影深度图所保存的深度是初始深度 0;然后根据 *pColor*. *b* = 0.5 来判断出④这种情况,因为相对光源不可见的片元不可能受到外半影图元的影响。对于⑤和⑥两种情况,根据 *fragDepth* 和 Z_3 的关系来判断,如果 $fragDepth < Z_3 + \varepsilon$,则属于情况⑤;否则属于情况⑥。

经过上面的判断后,对于情况②,直接使用片元原来的颜色,对于情况③,④和⑥,将其颜色赋为(0,0,0),对于情况①和⑤,将片元颜色和半影颜色图对应的过渡值进行混合。

这里还存在一个问题,经过上面的步骤后会有一些像素没有被判断出,主要因为半影颜色图采用的是线性插值,在本影区、内半影区、外半影区和完全照亮区的分界处会将 *pColor* 的 *B* 通道的值插值得到 0 到 0.5 区间的一个值。这对上述 1)的结果没有任何影响,最终影响到的是正确的内外半影分界处和应该处于完全阴影的区域,可以通过对最后余下的未判断的片元进行查找,查找出内外半影分界处的那些片元,将其颜色过渡值设为(0.3,0.3,0.3),其余部分片元颜色赋为(0,0,0)。由于这些片元在整个场景占很小一部分,查找的工作不会对最后帧率产生较大影响。

3　实验结果及分析

在一台 CPU 为 2.4 GHz,显卡为 NVIDIA GeForce GTX 280 的计算机上,使用 OpenGL 和 cg 实现了本文的算法。所有结果的深度图和颜色图的分辨率都是 800×800,最终绘制结果图像的分辨率也是 800×800。

本文算法在第 1 步使用了深度剥离的方法,由于深度剥离是一种多遍的算法,会对绘制帧率有一定的影响,表 1 给出了本文 3 遍深度剥离这一步和

表 1 深度剥离环节与常规阴影图生成的时间比较

Tab. 1 Comparison between depth peeling part and general shadow map

测试三角形个数	深度剥离/ms	阴影图生成/ms
2 000	0.781	0.407
7 700	0.974	0.46
30 000	1.88	0.86
69 000	5.882	2.058
97 000	9.26	3.40

文献[13]算法第 1 步的阴影图生成所需时间的比较,可以看出 3 遍深度剥离对三角形个数在 50 000 以下的模型而言,耗费的时间大概是常规阴影图生成时间的 2 倍左右。

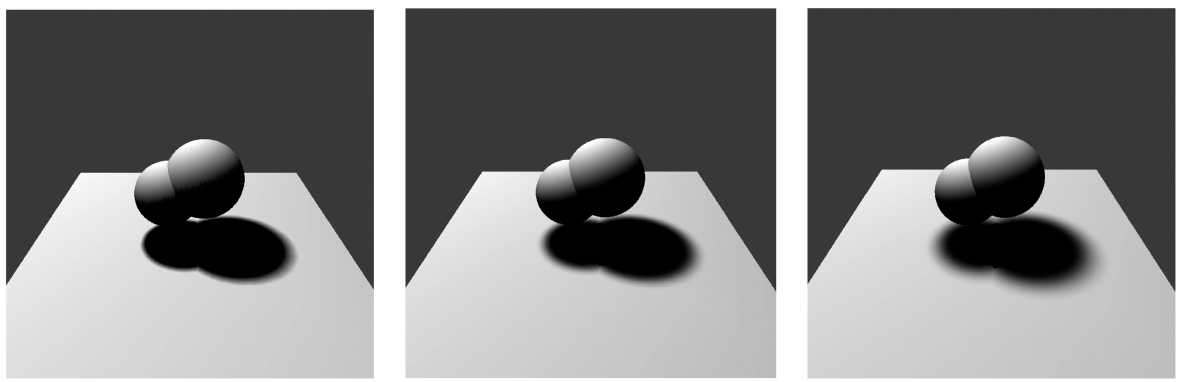
表 2 是图 8 和图 9 场景的绘制帧率统计数据和文献[13]方法得到的数据的比较,本文算法受第 1 步深度剥离的影响,绘制帧率比文献[13]算法低,但对一般场景而言仍然是实时的。

表 2 测试结果统计比较

Tab. 2 Statistic results

场景(三角形个数)	帧率/(帧/s)	
	文献[13]	本文算法
图 8(30 000)	460	318
图 9(a)(7 700)	980	712
图 9(b)(46 000)	338	231

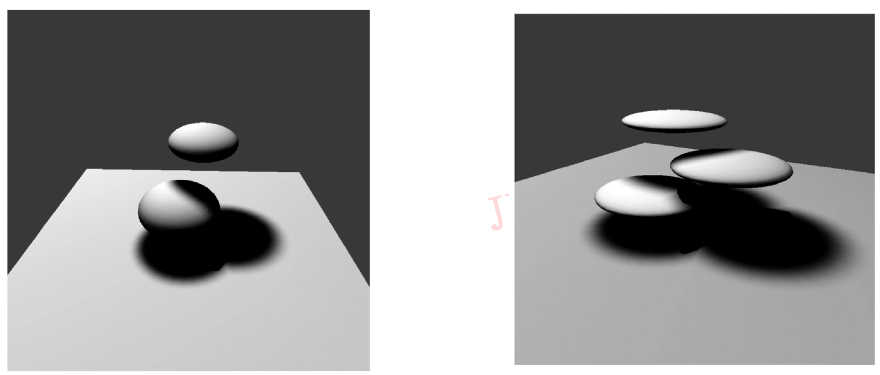
图 8 是本文算法和文献[13]算法的结果比较,图 8(a)是文献[13]的结果,图 8(b)和图 8(c)是本文的结果,其中图 8(b)是和图 8(a)在相同情况下的结果,图 8(c)是调节半影图元大小后的结果,可以看出本文结果的半影区边界过渡更加柔和一些,在场景不复杂的情况下可以很好地模拟相对较大面光源的内半影效果。



(a) 文献[13]的结果 (b) 本文结果 (c) 相对较大面光源

图 8 和文献[13]的结果比较

Fig. 8 Result comparison between paper[13] and ours



(a) (b)

图 9 其他场景

Fig. 9 Other results

图 9 是对多重遮挡场景的绘制结果,由于本文算法仅仅使用一张半影颜色图来进行半影区明暗过渡的估计,如果内外半影图元长度过大,会导致深度层次变化过多的区域产生不正确的结果,如图中在相对光源视点半影图元交叉的部分存在一小块全黑的错误结果。所以,对于相对较复杂的场景,内外半影图元的大小参数 r 和 t 只能尽可能的小一些,无法模拟相对较大的面光源。

4　结　论

提出了一种基于 GPU 的近似软影实时绘制算法,在场景相对不复杂的情况下可以比较好地模拟近似软影效果。该算法利用 GPU 来生成内外半影图元,并给出了使阴影区边界柔和的处理方法。算法中颜色的不同通道所保存的不同信息,加上深度剥离得到的最近三层深度图,都为最终软影的生成提供了判断的依据。

对于较复杂场景,本文算法对深度层次变化过多的区域会产生一些错误的判断结果,如何更好地进行重叠半影图元的混合以及如何让半影颜色图中一个值对多个可见像素起作用是下一步研究的方向。

参考文献(References)

[1] Wanger L. The effect of shadow quality on the perception of spatial relationships in computer generated imagery [C]// Proceedings of the Symposium on Interactive 3D Graphics. Cambridge, Massachusetts USA: ACM Press, 1992: 39-42.

[2] Williams L. Casting curved shadows on curved surfaces [C]// Proceedings of the SIGGRAPH. Atlanta, Georgia, USA: ACM Press, 1978: 270-274.

[3] Crow F C. Shadow algorithms for computer graphics [C]// Proceedings of the SIGGRAPH. San Jose, California, USA: ACM Press, 1977: 242-248.

[4] Wu Enhua, Liu Youquan. General purpose computation on GPU [J]. Journal of Computer-Aided Design & Computer Graphics, 2004, 16(5): 601-612. [吴恩华, 柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报, 2004, 16(5): 601-612.]

[5] Guennebaud G, Barthe L, Paulin M. Real-time soft shadow mapping by backprojection[C]//Proceedings of the Eurographics Symposium on Rendering. Nicosia, Cyprus: Eurographics, 2006: 227-234.

[6] Atty L, Holzschuch N, Lapierre M, et al. Soft shadow maps: efficient sampling of light source visibility [J]. Computer Graphics Forum, 2006, 25(4): 725-741.

[7] Schwarz M, Stamminger M. Bitmask soft shadows[J]. Computer Graphics Forum, 2007, 26(3): 515-524.

[8] Guennebaud G, Barthe L, Paulin M. High-quality adaptive soft shadow mapping[J]. Computer Graphics Forum, 2007, 26(3): 525-534.

[9] Bavoil L, Callahan S P, Silva C T. Robust soft shadow mapping with backprojection and depth peeling[J]. Journal of Graphics Tools, 2008, 13(1): 19-29.

[10] Wyman C, Hansen C. Penumbra maps: approximate soft shadows in real-time [C]//Proceedings of the Eurographics Symposium on Rendering. Leuven, Belgium: Eurographics, 2003: 202-207.

[11] Chan E, Durand F. Rendering fake soft shadows with smoothies [C]//Proceedings of the Eurographics Symposium on Rendering. Leuven, Belgium: Eurographics, 2003: 208-218.

[12] Cai X H, Jia Y T, Wang X, et al. Rendering soft shadows using multi-layered shadow fins[J]. Computer Graphics Forum, 2006, 25(1): 15-28.

[13] Lü Weiwei, Meng Weiliang, Xue Gaichao, et al. Real-time approximate soft shadow rendering on GPU [J]. Journal of Computer-Aided Design & Computer Graphics, 2009, 21(3): 275-281. [吕伟伟, 孟维亮, 薛盖超, 等. 基于 GPU 的近似软影实时绘制[J]. 计算机辅助设计与图形学学报, 2009, 21(3): 275-281.]

[14] Mammen A. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique [J]. IEEE Computer Graphics and Applications, 1989, 9(4): 43-55.

[15] Everitt C. Interactive Order-Independent Transparency [EB/OL]. (2001-05-15)[2009-04-02]. http://developer.nvidia.com/object/Interactive_Order_Transparency.html.