文章编号:1671 - 6833(2014)06 - 0064 - 05

传统软阴影算法在 GPGPU 构架上的优化技术研究

高明磊, 赵新灿, 詹 云

(郑州大学 信息工程学院,河南 郑州 450001)

摘 要: 针对百分比靠近软阴影 PCSS 算法存在阴影粉刺和偏离的问题,以及图形渲染实时性的要求,提出一种改进的软阴影算法. 该方法在 PCSS 算法的基础上,利用深度梯度消除阴影粉刺和偏离现象,使用方差阴影映射 VSM 算法进行过滤,利用 GPU 的并行计算能力加速生成区域求和表以实现对阴影图预先滤波. 实验结果表明,该方法能够得到更高的软阴影质量,并且可以获得很好的加速比,实时性明显提高.

关键词: GPGPU;百分比靠近软阴影;区域求和表

中图分类号: TP391 **文献标志码**: A doi:10.3969/j.issn.1671 - 6833.2014.06.016

0 引言

近年来,计算机图形学有着巨大的发展,随着应用需求的增加,真实感图形渲染也越来越受到关注,要生成视觉上十分精确的软阴影效果需要耗费很多的计算. 近些年,GPU 的运算性能快速增长,其浮点运算能力明显超越了 CPU,使得 GPU 擅于进行大规模密集型数据的并行计算[1-2]. GPU 强大的性能优势为高实时性的阴影研究提供了良好的解决思路,对设计虚拟场景中真实感强、实时性高的阴影,特别是对软阴影效果有着重要的研究意义.

阴影绘制算法主要分为阴影体算法[3]、阴影映射算法[4]和全局光照算法[5]. 在真实感方面,阴影体算法和全局光照算法是以牺牲实时性为代价,换取稍好的真实感;而阴影映射算法虽然存在走样等现象,通过过滤等方式依然能够获得很好的阴影效果[6]. 在实时性方面,阴影体算法和全局光照算法实时性较差;而阴影映射算法基于图像空间,算法复杂度与场景复杂度无关,且易于 GPU 加速,实时性较高. 阴影映射算法根据半影区情况分为两大类:半影区大小固定和半影区大小可变的阴影算法. 半影区大小固定的阴影算法主要有百分比接近滤波 PCF 算法、方差阴影映射 VSM 算法、卷积阴影映射 CSM 算法和指数阴影映射 ESM 算法;半影区大小可变的阴影算法主要有百分比靠近软阴影 PCSS(Percentage Closer Soft Shadow)算法[7],算法性能如表 1 所示.

1 基于 GPGPU 的改进软阴影映射算法

1.1 算法主要改进思想

由表 1 可知,PCSS 算法是相对更好的一种软阴影生成算法,软阴影效果更加符合阴影的物理特征. 在 PCSS 算法的基础上,提出一种新的软阴影生成算法,如图 1 所示.

首先,针对 PCSS 算法不能硬件预滤波的问

表 1 主要阴影算法特点对比

Tab. 1 Comparison of shadow algorithm

算法名称	算法所需存储量大小	硬件预滤波	半影区固定	算法主要不足之处
PCF	较小,存深度值	否	是	采样越多速率越低;自阴影现象
VSM	居中,存深度值和深度值平方	是	是	漏光现象
CSM	较大,存深度值正弦值和余弦值	是	是	存储量大;振铃现象
ESM	较小,存深度值的指数值	是	是	计算量大;稳定性较差
PCSS	较小,存深度值	否	否	采样越多速率越低;自阴影现象

收稿日期:2014 - 06 - 06;**修订日期**:2014 - 09 - 05

基金项目:国家自然科学基金委员会 - 中国民用航空总局联合研究基金项目(U1433106)

作者简介:高明磊(1963 -),女,山东诸城人,郑州大学实验师,研究方向为计算机技术,E-mail:iemlgao@zzu.edu.cn.

题,使用 VSM 算法代替 PCF 算法进行过滤,这样在阴影图中存储的是深度值和深度值的平方,对阴影图就可以使用各向异性过滤、区域求和表等滤波技术,加速滤波效率,提高阴影质量.

其次,由于阴影图滤波中每个像素点的滤波区域是动态计算的,生成区域求和表耗时多,针对此问题,笔者采用 DirectCompute 在 GPU 上加速生成区域求和表,提高滤波效率.

最后,使用泊松圆盘采样代替规则采样,无需过多的采样便能提高阴影质量,针对自阴影问题,采用深度梯度法得到可变的深度偏移值来处理.

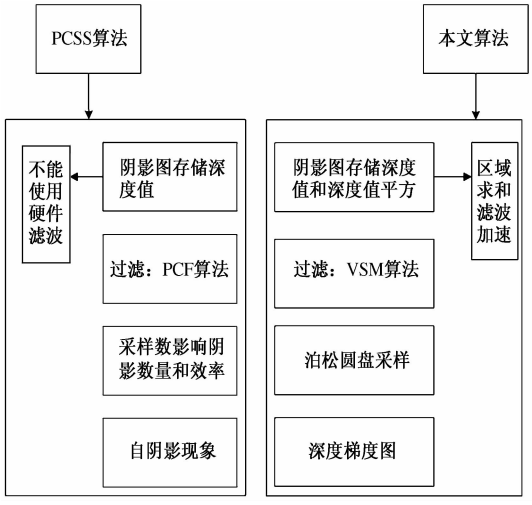


图 1　本文拟采用方法

Fig. 1　Research methods of this article

本文软阴影算法由 CPU 端和 GPU 端共同完成,其中,在 CPU 上主要实现场景初始化、信息更新等任务;算法的大部分工作由 GPU 实现,利用 DirectX 11 中可编程部分 vertex shader、pixel shader 和 compute shader,充分利用 GPU 的性能优势,加速和优化算法的实现,从而获得更高的质量和效率,任务划分如图 2 所示.

1.2　基于区域求和表的阴影图预滤波

利用区域求和表技术对阴影图动态滤波,得到深度值和深度值平方的期望,进而计算相应的均值和方差. 再根据切比雪夫不等式得到像素点的阴影权值. 实现时将接受体当前点深度值与区域求和表求得的滤波区域均值进行比较. 如果当前点深度值小于区域均值,则当前点不在阴影中,$P=1$;如果当前点深度值大于区域均值,则利用切比雪夫不等式求出当前点在阴影中的概率上限 $P_{\max}$,以此来渲染阴影效果.

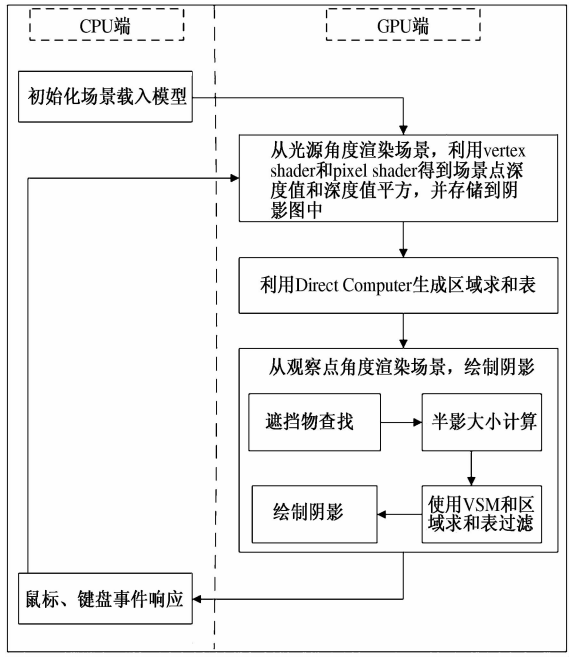


图 2　基于 GPGPU 的软阴影映射算法流程

Fig. 2　Soft shadow mapping algorithm based on GPGPU

对于具有元素 $a[i,j]$ 的纹理可按公式(1)建立一个区域求和表 $t[i,j]$:

$$t[i,j]=\sum_{x=0}^{i}\sum_{y=0}^{j}a[x,y]. \tag{1}$$

区域求和表中的每个元素为纹理中位于这个元素左侧及上侧所有元素之和,生成区域求和表后,就可以求得任意一个矩形滤波区域的和. 通过对滤波区域的 4 个角点进行采样求区域和值,再将区域和值除以区域面积,就可以得到滤波区域的均值. 如图 3 所示,左图灰色部分代表滤波区域,采样该区域的 4 个角点 A、B、C 和 D,则该滤波区域的均值为 $(A-B-C+D)/4$,即为 4.5.

数据s[x,y]

1	2	3
4	7	0
4	5	0

区域求和表t[i,j]

A 1	3	6 B
5	14	17
C 9	23	32 D

图 3　区域求和表计算过程

Fig. 3　Summed area table calculation process

1.3　基于泊松圆盘采样的遮挡物查找

在遮挡物查找过程中,需要多次采样求遮挡物的平均遮挡深度. 采样数目越多,软阴影的质量越好,但是过多的采样数会降低算法的性能. 泊松圆盘采样模式(poisson disk pattern)[8] 就是目前非常好的一种图像采样方式. 通常采用掷标法(dart throwing)来生成泊松圆盘序列,所有采样点分布在半径为 1 的单位圆盘上,得到的所有采样

点之间的距离都大于一个最小距离(记为阈值),该阈值设置在0.05~1.0最佳.在遮挡物查找过程中,将泊松圆盘采样点与遮挡物查找区域半径相乘作为纹理坐标偏移量,再对处理后的纹理坐标进行采样得到阴影图中相应纹理坐标下的深度值.通过上述得到阴影图中相应坐标下的深度值后与接受体当前点的深度值进行比较,从而计算遮挡物的平均遮挡深度.

1.4 基于深度梯度法的自阴影处理

为了处理自阴影现象,需要动态计算深度偏移值.深度梯度法能够有效地处理自阴影问题,消除阴影粉刺或偏离现象.对于给定的函数 $d(x,y)$,在位置(x,y)处的梯度矢量如公式(2).

$$\boldsymbol{G}[d(x,y)] = \begin{bmatrix} \dfrac{\partial d}{\partial x} \\ \dfrac{\partial d}{\partial y} \end{bmatrix}. \tag{2}$$

计算深度偏移值时,必须把屏幕空间转换到纹理空间,转换如公式(3):

$$gradient = \begin{bmatrix} \dfrac{\partial d}{\partial u} \\ \dfrac{\partial d}{\partial v} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} \end{bmatrix} - T \begin{bmatrix} \dfrac{\partial d}{\partial x} \\ \dfrac{\partial d}{\partial y} \end{bmatrix}. \tag{3}$$

其中,*gradient* 即为转换到纹理空间的深度梯度;$\boldsymbol{G}[d(x,y)]$是屏幕空间的深度梯度矢量;中间是纹理空间的雅克比矩阵.

将深度梯度值与偏移量 *uv_offset* 进行点乘,就可以得到深度偏移值,从而计算最终的深度值,如公式(4):

$$d = d_0 + dot(uv\_offset, gradient). \tag{4}$$

将该值与相应纹理坐标下采样得到的深度值进行比较,就可以有效消除阴影粉刺或偏离现象.

2 基于GPGPU的区域求和表生成

2.1 区域求和表生成方法

区域求和表生成有两种方法:逐行方法(line-by-line)和回溯倍增方法(recursive doubling).回溯倍增法[9-10]可以在GPU端并行计算,算法中前缀求和是基于Hillils和Steele提出的扫描算法,对于一个二进制运算符⊕和一个 n 元数组 $A=\{a_0,a_1,\cdots,a_{n-1}\}$,进行前缀求和得到的结果每个元素都是其所有前缀和其自身经过⊕运算得到的结果,记为 $B=\{a_0,(a_0\oplus a_1),\cdots\cdots,(a_0\oplus a_1\oplus\cdots a_{n-1})\}$.

2.2 区域求和表在GPU上的实现

(1)将生成的阴影图分成大小相同的块,每块对应GPGPU的一个线程组,如图4所示.笔者采用的GPU是NVIDIA GeForce GT440,每个线程组包含的线程数限制在1 024内.对于高和宽分别为 h 和 w 的阴影图,假设每个分块大小为16*32共512个像素点,则需要分成的线程组数为$(w/16)*(h/32)$,如果分块的尺寸不能被线程组的尺寸整除,则实际分配的线程组数应该为$(w/16+1)*(h/32+1)$[11].

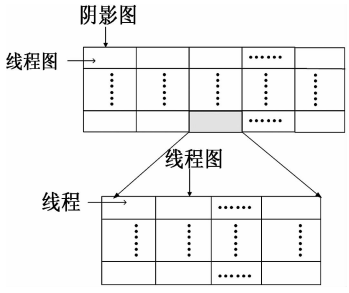


图4 阴影图分块

Fig. 4 Shadow map block

(2)将阴影图的每个分块数据读入线程组的共享存储器中,使得线程组内的所有线程可以同时读取分块数据;利用线程对分块的每一行进行并行前缀求和运算,并利用线程同步机制待所有线程执行完毕后再将最终结果写入类型为RW-Texture2D的全局存储器中.

(3)将前面计算得到的每个分块结果的每行最后一个元素值即分块总和,从全局存储器读到共享存储器中,利用线程对这些分块总和进行并行前缀求和运算.将对应行分块总和i加到分块i+1的对应行的所有值上,并将最终结果写入全局存储器中,就完成了阴影图中所有行的前缀求和运算.

(4)对上一步的结果进行转置运算,对转置后的阴影图数据重复上面三步,进行所有列的运算,再将最终结果进行转置便生成了区域求和表,如图5所示.

3 基于GPGPU的软阴影算法实验结果分析

3.1 PCSS算法规则采样效果

软阴影的渲染速率和阴影质量与采样模式以及采样数等因素密切相关.图6所示为场景使用PCSS算法在规则采样模式下,分别利用25,49,81个不同数目的采样点对场景进行阴影渲染的细节特写,阴影图大小为1 024*1 024.

当采样数为25时,阴影中有明显的带状条纹,随着采样数目的增多,阴影效果越来越好.当

采样数为 81 时,阴影边缘已经比较柔和.但是,随着采样数目的增多,算法的效率却在降低,如表 2 所示.

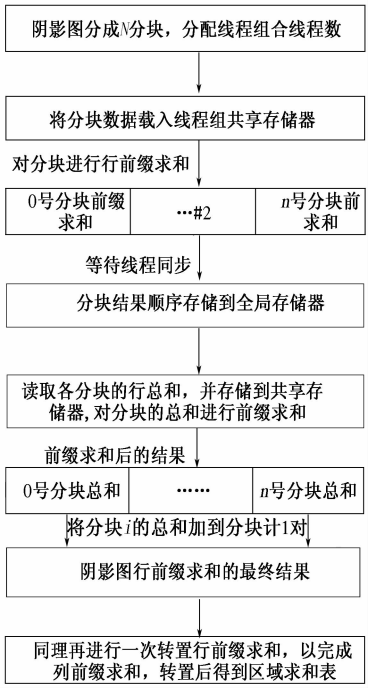
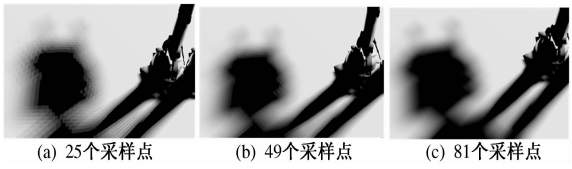


图 5　基于 GPGPU 的区域求和表计算

Fig. 5　Summed area table calculation based on GPGPU



(a) 25个采样点　(b) 49个采样点　(c) 81个采样点

图 6　不同采样数阴影效果细节对比

Fig. 6　Detail contrast of different sampling number

表 2　PCSS 算法性能分析

Tab. 2　Performance analysis of PCSS algorithm

采样数	绘制效果	渲染帧速/(f·s^{-1})
25	较差	108.39
49	一般	81.99
81	较好	58.23

3.2　改进算法泊松圆盘采样效果

为解决阴影质量与渲染效率的矛盾,使用泊松圆盘采样代替规则采样模式.图 7 所示为在两种不同采样模式下的软阴影渲染效果,遮挡物查找阶段使用的采样点数均为 64.

图 7(a)、(b)为 PCSS 算法在规则采样方式下渲染的软阴影效果,过滤阶段采样数分别为 64,361;图 7(c)图为 PCSS 算法在泊松圆盘采样方式下渲染的软阴影效果,过滤阶段采样数为 64.从图中可以看出,图 7(a)图中软阴影边缘仍有部分带状走样;(b)图过滤阶段使用较多的采样点,软阴影边缘较柔和;而(c)图使用泊松圆盘采样,过滤阶段仅使用 64 个采样点,得到的软阴影与(b)的效果差不多,算法性能如表 3 所示.

(a) 64个规则采样点　(b) 361个规则采样点　(c) 64个泊松圆盘采样点

图 7　不同采样模式对软阴影效果的影响

Fig. 7　Effect of the different sampling patterns

表 3　规则采样与泊松圆盘采样效果对比

Tab. 3　Comparison of two sampling methods

采样模式	过滤采样数	帧速/(f·s^{-1})	阴影质量
规则采样	64	69.24	一般
	361	47.85	较好
泊松圆盘采样	64	63.17	较好

3.3　改进算法自阴影效果处理

针对自阴影现象采用深度梯度法进行处理,效果如图 8 所示.在图 8(a)中出现了自阴影现象,如模型足部与阴影之间有缝隙,出现了阴影移位,这主要是由于常量深度偏移值选择不当造成的;而图 8(b)中,采用深度梯度法动态计算偏移量,有效地消除了自阴影现象,软阴影效果较好.

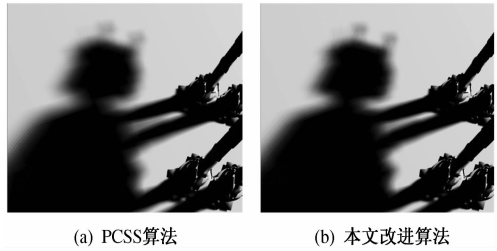
(a) 使用常量深度偏移值　(b) 使用深度梯度法

图 8　自阴影现象的消除

Fig. 8　Elimination of self shadow

3.4　改进算法与原算法整体性能的对比

PCSS 算法和本文算法渲染的效果比较如图 9 所示.从图 9 中可以看出,本文算法渲染的软阴影边缘柔和,效果更加清晰,完全满足了软阴影真实感渲染的需求.



(a) PCSS算法　(b) 本文改进算法

图 9　PCSS 算法与本文改进算法渲染的软阴影效果

Fig. 9　Rendering soft shadows of PCSS and improved algonthm

改进算法与原算法性能对比如表4所示.

表4 PCSS算法与改进算法性能对比

Tab.4 Comparison of PCSS algorithm and improved algorithm

算法	阴影图大小	渲染帧速/(f·s^{-1})
PCSS算法	512*512	69.24
	1 024*1 024	88.94
本文算法	512*512	157.21
	1 024*1 024	205.39

4 结论

鉴于阴影对增强三维场景真实感渲染的重要性,在PCSS算法的基础上提出一种改进的软阴影算法.利用深度梯度法得到动态的深度偏移值消除阴影粉刺和偏离问题;在过滤阶段利用VSM算法代替PCF算法,并结合区域求和表对阴影图进行预先滤波;利用GPU通用计算的优势生成区域求和表算法,并和传统的利用Direct3D的像素着色器生成区域求和表算法进行了对比.结果表明,改进后的软阴影算法具有更好的阴影质量和更高的性能,加速比是原PCSS算法的3倍以上,充分体现了本文算法的优越性.

参考文献:

[1] 唐滔.面向CPU-GPU异构并行系统的编程模型与编译优化关键技术研究[D].长沙:国防科学技术大学计算机学院,2012.

[2] 韩博,周秉锋.GPGPU性能模型及应用实例分析[J].计算机辅助设计与图形学学报,2009,21(9):1219-1226.

[3] TIM H. Real shadows, real time[R]. Iris: Iris Universe,1991:23-31.

[4] BAVOIL L, CALLAHAN STEVEN P, CLAUDIO T. et al. Robust soft shadow mapping with backprojection and depth peeling[J]. Journal of Graphics Tools, 2008, 13(1):19-29.

[5] APPEL A. Some techniques for shading machine renderings of solids[C]//AFIPS 1968 Spring Joint Computer Conf. San Francisco, California, 1968, 32: 37-45.

[6] LIU Lu, XIAO Shuang-jiu. Real-time soft shadows for large-scale virtual environments[C]//2011 International Conference on Multinedia Technology (ICMT). Hangzhou, China:IEEE Press,2011:5464-5467.

[7] FERNANDO R. Percentage-closer soft shadows[C]// ACM SIGGRAPH 2005 Sketches and Applications. New York:ACM Press, 2005:35

[8] DUNBAR D,HUMPHREYS G. A spatial data structure for fast poisson-disk sample generation[J]. ACM Transactions on Graphics,2006,25(3):503-508.

[9] NEHAB D, MAXIMO A, RODOLFO LIMA, et al. GPU-efficient recursive filtering and summed-area tables[J]. ACM Transactions on Graphics, 2011, 30 (6):1-11.

[10] FRANKLIN C C. Summed-area tables for texture mapping[C]//SIGGRAPH 1984:Proceedings of The 11th Annual Conference on Computer Graphics and Interactive Techniques. Minneapolis, Minnesota:ACM Press, 1984:207-202.

[11] 孙伟东,马宗民.一种适合于GPU计算的并行后缀数组构造算法[J].小型微型计算机系统,2011,32 (5):830-836.

Study of Traditional Soft Shadow Algorithm Optimization Technology in the GPGPU Framework

GAO Ming-lei, ZHAO Xin-can, ZHAN Yun

(School of Information Engineering, Zhengzhou University,Zhengzhou 450001,China)

Abstract: Aiming at the realistic need of shadow rendering, on the basis of the PCSS algorithm, this paper puts forward a soft shadow generation algorithm based on GPGPU. It samples a region by poisson disk pattern instead of regular sampling mode, uses depth gradient to deal with the phenomenon of self-shadowing, uses the VSM algorithm to filter the shadow, and uses the summed area table to filter the shadow map dynamically, which can improve the shadow quality and rendering efficiency in a certain extent.

Key words: GPGPU;PCSS; summed area table