简化的后处理快速景深效果

余 伟[1] 张 扬[2]
[1](西华大学数学与计算机学院 四川 成都 610039)
[2](西南交通大学信息科学与技术学院 四川 成都 610031)

摘 要 基于几何光学中透镜成像模型,在保持景深真实特点的同时,采用后处理实现景深效果,提出一种易于在图形处理器上实现的简化算法,以进一步提高景深效果计算速度。相对于已有的后处理景深在图形处理器上的实现,该方法使用帧缓存对象(FBO)通过一遍模型绘制和一遍后处理实现景深效果。该算法可以在不改动固定图形管线渲染代码的情况下对给定场景实现快速的景深效果。实验表明,在通用的硬件条件下该后处理耗时 0.3 毫秒左右。

关键词 景深 后处理 实时绘制 虚拟现实 图形处理器

SIMPLIFIED ALGORITHM FOR POST-PROCESSING FAST DEPTH OF FIELD EFFECT

Yu Wei[1] Zhang Yang[2]
[1] (School of Mathematics and Computer Engineering, Xihua University, Chengdu 610039, Sichuan, China)
[2] (School of Information and Technology, Southwest Jiaotong University, Chengdu 610031, Sichuan, China)

Abstract Based on imaging model of geometric optics, a simplified algorithm which fits for the implementation on graphics processor units was proposed in this paper to further improve computation speed for large depth of field effect by adopting post-processing method to realise the effect of DOF while keeping the reality characteristic of the DOF. Compared with current implementation of post-processing DOF effect on graphics processor units, only one-off model rendering and one-off post-processing are needed in this algorithm with FBO (Frame Buffer Object) to realise DOF effect. It can be used to implement fast DOF effect on a given model rendering without changing any original code of fixed graphics pipeline. Experiment demonstrated that our post-processing costs about 0.3 ms under general hardware conditions.

Keywords Depth of field (DOF) Post-processing Real-time rendering Virtual reality Graphics processor unit

0 引 言

照相机和人眼成像系统可以根据观察物体相对于观察者位置的远近做适当调节,使处于焦距位置的物体,可以清晰地在底片和视网膜上成像;而那些更远或者更近的物体,由于没有处于焦距位置,出现成像模糊的现象,这种现象称为景深 DOF(Depth of Field)现象,它是光成像中固有的现象。景深在摄影和电影中作为一种艺术表现手法,有重要的应用,是计算机虚拟现实,真实感绘制中研究的重要问题。

在计算机中实现景深效果,主要有以下一些方法:1)光线跟踪的方法。该方法模拟多条光线在场景中传播的方法,计算透镜成像的真实效果[3,4]。这种方法虽然可以生成真实的模糊效果,但是复杂的算法使得光线跟踪在现有的图形系统中实现起来仍旧十分耗时。2)多遍绘制的方法。该方法采用多遍绘制的方法生成景深效果,最终渲染出来的图像由一系列图像积累综合而成,如文献[1,2]使用图形硬件中的积累缓存综合而成的图像。该种方法需要对场景绘制数遍,一旦场景复杂,将会影响渲染速度。另外,该方法没有精确的透镜模型,生成的效果缺乏真实感。3)后处理的方法。对图形系统渲染的结果后处理,能够快速生成较真实的景深效果。该方法根据透镜成像模型采样图像滤波方法模糊像素点,是比较实用的方法[5-8]。使用图形处理器的片元程序,在图形管线末端对渲染图像后处理可以快速生成景深效果。

在大场景中,绘制速度成为考虑算法是否实用的重要因素。本文基于渲染后处理的方法,通过简化真实透镜成像模型,在图形处理器上实现一遍后处理的快速景深效果。和文献[1,2]相比,速度得到提高,本文实验中有具体比较;和文献[6-8]相比,本文方法模块性更强,计算过程更简单,从而可以进一步提高速度。

1 几何光学模型

如图 1所示的薄透镜成像模型,满足薄透镜光学成像公式(1),其中薄透镜焦距为 f,物距和像距分别为 u和 v。

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \tag{1}$$

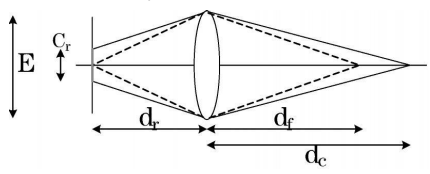

图 1 几何光学模型

收稿日期:2009-02-18。余伟,讲师,主研领域:图形图像处理,软件工程。

对于物距为 d_f 的透镜聚焦位置的物点可以在像距为 d_r 的成像屏幕上呈现清晰像点; 而位于非聚焦位置 d_c 的物点, 像距大于 d_r, 所以该点在成像屏幕上呈现半径直径为 C_r 的光斑, 一个模糊的像点, 该光斑称为散光圈 COC(Circle of Confusion)。

设透镜的直径为 E, 由式(2)可以计算出散光圈的直径 C_r,

$$C_r = |V_d - V_f|(E - V_d) \tag{2}$$

其中, $V_d = \frac{fd_c}{d_c - f}$, $V_f = \frac{fd_f}{d_f - f}$。

根据以上光学模型和参数关系, 我们可知。1)像点模糊程度取决于 C_r 大小; 2)C_r 与物距在光器件聚焦位置, 数值关系发生大的变化。根据这些特性, 我们对上述景深模型进行简化。

2 模型简化及其在图形处理器上的实现

2.1 COC算法的简化

近似计算散光圈由式(3)所示, 散光圈半径和景物深度关系在一定范围内简化成线性关系。

$$C'_r = |(d'_c - d'_f) \cdot \alpha| \tag{3}$$

$$\alpha = \begin{cases} \alpha_1 & (d'_c > d'_f) \\ \alpha_2 & (d'_c < d'_f) \end{cases} \tag{4}$$

式中 C'_r 表示屏幕坐标下以像素为单位的散光圈模糊直径。d'_c 和 d'_f 取自渲染的深度缓存, 其取值范围为 $[0, 1]$。由于该数值直接取自固定图形管线的深度缓存中, 无需计算, 所以提高了计算效率。该数值虽然不是真实场景的远近值, 但是反映出物点距离观察者的远近关系。为了接近真实的光扩散模型, 近景模糊和远景模糊采用不同的斜率参数 α, 通常情况下 $\alpha_1 < \alpha_2$。

2.2 防止颜色渗漏的像素加权采样

在渲染图像的后处理中, 根据 C'_r 值决定该物点的模糊程度, 通常采用卷积模板滤波对像素点模糊。而考虑到在图形处理器实现卷积滤波对于较深点的模糊需要多遍绘制才能完成, 这样会导致性能下降。所以我们采用基于泊松分布的采样模板。C'_r 值在这里表示成像素采样的最大半径如图 2所示。

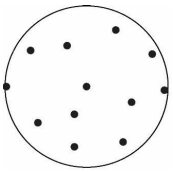

图 2 采样模糊

采样点首先在单位圆中定义符合泊松分布的坐标位置, 然后根据中心像素点的散光圈大小缩放该采样圆, 最后根据公式(5)决定最终颜色值。

$$Color = \sum_{i=1}^{n}(\beta_i C_i) \Big/ \sum_{i=1}^{n}\beta_i \tag{5}$$

其中 $Color$ 表示模糊后的颜色值, C_i、β_i 分别表示采样颜色点及其权值参数, 一般情况下当前点的权值略大于相邻采样点的权值。

后处理的景深效果模拟中, 常常会产生颜色渗漏的问题, 即前景物体的颜色在采样滤波中渗漏到后景物体中, 如图 3所示。通过深度值比较, 设定采样权值的方法可以解决该问题。最终像素值在 SM D形式的片元程序下由以下算法决定。

1) 计算采样点和当前点深度差 ds;

图 3 设置参数 MaxDis前的效果

2) $\beta_i = (ds > MaxDis)\ ?\ 0 : 1$;

3) 按照公式(5)加权计算最终值 $Color$。

设置了采样深度比较参数后减少颜色渗漏的放大效果如图 4所示。

图 4 设置了参数 MaxDis值后效果

2.3 图形处理器上的实现

现代图形处理器的可编程性为实现后处理的景深效果提供高性能的硬件加速功能。我们在 OpenGL平台下以 Cg编写的一个片元程序实现了上述算法, 其核心代码如下:

```
//中心点和场景聚焦深度的深度差
float dr= centerdepth-df;
//散光圈最大半径
float2 CoC= abs( dr* a);
CoC= clamp(CoC, 0, MaxCr);
Float Cr= (dr< 0)? CoC. x: CoC. y;
float offsetdepth;
float4 offsetcolor;
float dWeight;
float4 resultColor;
//计算中心点颜色贡献, RGBA四通道颜色
float4 accuColor= centercolor* dWeight;
float totalContributation= dWeight;
//泊松分布采样模糊
for( i= 0; i< 7; i= i+ 1)
{ offsetcolor= texRECT( texcolor, coPoisson[ i]* Cr+ texCoord);
offsetdepth= texRECT( texdepth, coPoisson[ i]* Cr+ texCoord). x;
//防渗漏深度差比较
dWeight= ( abs( offsetdepth-centerdepth) > MaxDis)? 0: 1;
accuColor= accuColor+ offsetcolor* dWeight;
totalContributation= totalContributation+ dWeight;
}
resultColor= accuColor/ totalContributation;
return resultColor;
```

该实现步骤如图 5(c)所示, 首先进行场景的离屏渲染, 利用原有的场景绘制代码渲染场景至帧缓存对象 FBO(Frame-buffer Object)中, 并设置颜色纹理和深度缓存纹理。场景绘制后设置屏幕绘制状态。然后, 用颜色和深度纹理作为输入纹理, 做图像的景深后处理, 实现上述算法。后处理中主要通过传递片元程序的参数, 控制景深效果。简化的模型不以模拟光照透镜模型为实现方法, 而是仅仅以深度缓存和像素模糊为调整对象, 通过调节参数达到满意的效果。

本文的方法和已有的基于图形硬件加速方法的处理过程比较如图 5所示。其中图 5(a)为文献[1, 2]的处理过程, 需要对一个场景模型绘制多遍; 图 5(b)为文献[7, 8]的处理过程, 需要在模型绘制时加入特定的顶点处理和片元处理程序, 在图像后

处理时再进行一次处理;本文的景深实现方法如图 5(c)所示,由于后处理集中与一遍片元程序处理中,所以无需对原有的模型绘制程序作任何更改,简化了处理过程,同时增强了模块性。

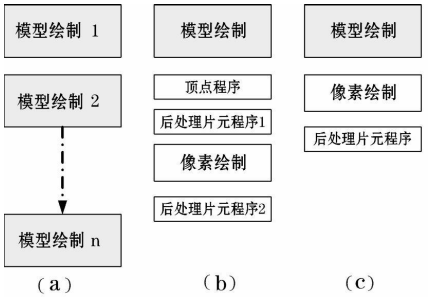

图 5　处理过程比较

表 1列出了参数化景深效果用到的重要设置参数。

表 **1**　参数表

参数名	功能描述
α	散光圈扩散斜率
d_c	像素深度
d_f	聚焦深度
MaxDis	最大采样深度差
MaxCr	最大散光圈半径
coPoisson	泊松分布采样坐标

3　实验和比较

和 OpenGL RedBook[2]中的景深例子比较,我们在 128bit 128MB Geforce 5700 AGP4X图形加速卡; PIII 1GHz CPU; 256MB内存; windows XP操作系统下实验。采用参数化设置,简化了景深效果的光学模型,从而达到提高速度、加速绘制的目的,从表 2中看出,本文方法大致比多遍绘制快 8–9倍。性能比较见表 2和图 6。

表 **2**　性能比较

场景像素大小	积累缓存方法耗时(us)	本文方法耗时(us)	加速比
300* 200	151986	19590	7.8X
600* 400	162877	19598	8.3X
800* 600	192243	19656	9.8X
1024* 768	236639	19853	11.9X

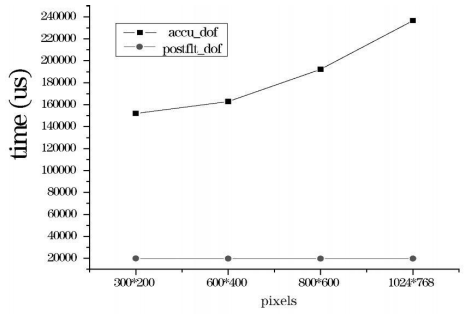

图 6　性能比较图

在本文方法的实验中,模型绘制和像素绘制后处理耗时由表 3所示。从该表可看出,后处理的时间大约为一遍绘制时间的 1/50,由于后处理在图形处理器上执行非常快以致超出时间测试函数的精度,所以本表最后一列已经反映不出后处理耗时随像素规模的变化趋势。

表 **3**　模型绘制和后处理耗时比较

场景大小	模型绘制耗时(us)	后处理耗时(us)
300* 200	19590	333
600* 400	19598	354
800* 600	19656	324
1024* 768	19853	349

效果展示见图 7–图 10。

图 7　无景深效果的渲染

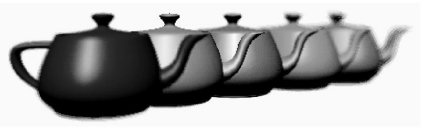

图 8　采用积累缓存的景深实现效果

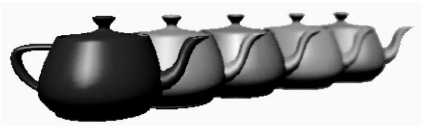

图 9　本文方法的远景模糊景深效果

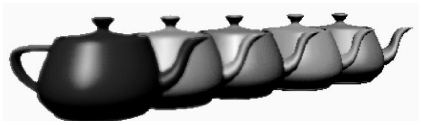

图 10　本文方法的近景模糊景深效果

4　结　论

本文介绍了一种使用简化的后处理景深效果在图形处理器上实现方法,速度快、模块化强。因此适合于处理现有大量的没有景深效果的固定图形管线下的场景渲染代码,在通用的硬件条件下实现快速的景深效果。

参　考　文　献

[1] Haeberli P, Akeley K. The Accumulation Buffer: Hardware support for High-Quality Rendering [J]. Computer Graphics, 1990, 24(4): 309-318.

[2] Shreiner D, 等. OpenGL Programming Guide [M]. 邓郑祥, 译. 北京: 人民邮电出版社, 2005: 328-330.

[3] Cook R L, Porter T, Carenter L. Distributed Ray Tracing[J]. Computer Graphics, 1984, 18: 137-145.

[4] 吴向阳, 鲍虎军, 陈为, 等. 采用正向光线跟踪的照相机成像实时模拟[J]. 计算机辅助设计与图形学学报, 2005, 17(7): 1427-1433.

[5] Rokita P. Generating Depth-of-field Effects in Virtual Reality Applications[J]. IEEE Computer Graphics and Applications, 1996, 16(2): 18-21.

(下转第 230页)

标准参考图像(cameraman, 256×256),采用高斯滤波模糊化来模拟受损图像,图 2(b)是对图像的左半部分模糊化;图 2(c)是对图像的右半部分模糊化。目的就是通过融合得到清晰的、含有更多信息量的图像。图(2)(d)、(e)、(f)分别为三种方法的融合结果,表 1给出了融合算法的性能评价。图 3给出了另一组多聚焦图像(标准参考图像为 peppers, 128×128)的融合实验结果,表 2给出了相应的融合算法的性能评价。

图 2 不同规则的融合图像效果比较(cameraman)

图 3 不同规则的融合图像效果比较(peppers)

由表 1和表 2可知,改进规则的融合效果优于传统的选择和加权平均规则,融合图像和参考图像的均方误差最小,融合图像的峰值信噪比最大。

表 1 三种融合规则的性能比较(cameraman)

融合方法	MSE	PSNR(dB)
选择规则	5.6897	40.58
加权平均规则	8.1888	39.00
改进规则	4.6508	41.46

表 2 三种融合规则的性能比较(peppers)

融合方法	MSE	PSNR(dB)
选择规则	6.2534	40.17
加权平均规则	11.7766	37.42
改进规则	5.4964	40.73

4 结 论

局部能量准则是一种常用的小波图像融合规则,但传统的局部能量准则对窗口内像素所包含的独立信息考虑不足,每个像素所包含的正负极性符号信息被丢失。窗口的中心像素是十分重要的像素,本文利用中心像素所包含的相位特征,增加融合决策过程中的信息量。以窗口中心像素的小波系数正负号相位特征为指导,提出了一种基于局部能量的选择和加权平均相结合的新融合规则。当待融合的二个小波系数正负极性符号相同时,采用局部能量极大值准则选择融合图像小波系数;当正负极性符号相反时,根据局部能量之间的差距大小,分别采用局部能量极大值准则和加权平均规则。实验结果表明,新规则的融合效果优于传统的局部能量极大值准则和加权平均规则。此外,新规则计算简单,使用方便,通用性强,是一种有效、实用的图像融合规则。

参 考 文 献

[1] David L H. An Introduction to Multisensor Fusion[J]. Proc of the IEEE, 1997, 85(1): 6-23.
[2] 赵天昀. 基于方差的图像融合[J]. 河南理工大学学报: 自然科学版, 2007, 26(3): 302-306.
[3] 曾基兵, 陈怀新, 王卫星. 基于改进局部梯度的小波图像融合方法[J]. 电视技术, 2007, 31(8): 18-20.
[4] 侯建华, 熊承义, 喻胜辉. 基于小波局部能量特征的多分辨率图像融合[J]. 中南民族大学学报, 2006, 25(2): 47-50.
[5] 苗启广, 王宝树. 基于小波变换与局部能量的多聚焦图像融合[J]. 计算机科学, 2005, 32(2): 229-232.
[6] 周锐锐, 陈振华, 毕笃彦. 一种基于局部能量的图像融合方法[J]. 中国体视学与图像分析, 2006, 11(3): 226-229.

(上接第 217页)

[6] Bertalm′ioM, Fort P, S′anchez-Crespo D. Real-time, Accurate Depth of Field using Anisotropic Diffusion and Programmable Graphics Cards[C]//Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission, Sept 2004: 767-773.
[7] Riguer G, Tatarchuk N, Isidoro J. Real-Time Depth of Field Simulation[OL]. ShaderX2, Wordware 2003. http://www.ati.com.
[8] Scheuermann T, Tatarchuk N. Improved Depth of Field Rendering[OL]. ShaderX3, Charles River Media 2004. http://www.ati.com.