一种基于亮度和深度信息的实时景深渲染算法

赵东阳，陈一民，李启明，刘　燕，黄　晨，徐　升，周明珠
(上海大学 计算机工程与科学学院, 上海 200072)

摘　要: 在研究透镜成像模型与针孔成像模型的基础上，提出了一种基于*亮度和深度信息*的实时景深渲染算法，*首次把景深渲染应用到增强现实系统*。使用 OpenGL 和 Visual Studio2008 实现整个算法。通过建立三维空间坐标系，*实时追踪和获取真实场景深度信息*，把生成虚拟物体融入到真实场景中。随后，依据*亮度信息对虚实融合场景进行模糊处理*。最后进行图像融合，把模拟的景深效果输出到屏幕。实验结果表明，新算法更接近于真实摄像机拍摄的景深效果图，适用于增强现实系统。

关键词: 景深; 实时性; 增强现实; 亮度信息
中图分类号: TP391.9　　**文献标识码:** A　　**文章编号:** 1004-731X (2012) 08-1612-06

Luminance and Depth Based Real-time Depth of Field Rendering Algorithm

ZHAO Dong-yang, CHEN Yi-min, LI Qi-ming, LIU Yan, HUANG Chen, XU Sheng, ZHOU Ming-zhu
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract: According to *the luminance and depth information*, *a real-time depth of field rendering algorithm of Augmented Reality* was proposed based on the research of lens image and ideal camera model. The algorithm with OpenGL and VisualStudio2008 were realized. At first, virtual objects were merged into the real scenes by establishing three-dimensional coordinate system for *real-time tracking and access to the real scene depth information*. Next, the merged image was rendered, which depended on *the brightness and the depth of the scene*. Last, the images were merged and the simulated effect of depth was output to the screen. The result shows that the algorithm is much closer to reality, and it can be used to the AR system.
Key words: depth of field; real-time; Augmented Reality; luminance information

引　言

增强现实（Augmented Reality，AR）是在虚拟现实(Virtual Reality，VR）基础上发展起来，通过计算机系统提供的信息增加用户对现实世界感知的技术，并将计算机生成的虚拟物体、场景或系统提示信息叠加到真实场景中，从而实现对现实的“增强”。然而，目前关于景深问题的研究，绝大多数都集中在虚拟现实系统中，并且研究的算法几乎都没有考虑到亮度对景深的影响，在增强现实系统中，几乎都未引入景深效果，整个虚拟场景成像前后都是清晰的，因而造成整个场景显得不够自然、真实，缺乏沉浸感，易造成眼睛不适应与疲劳。加入景深效果，可以提高系统的真实感和沉浸感，提供给用户一些场景深度信息，并真正的实现虚实融合。目前在计算机图形学领域已经提出了许多关于景深渲染的算法，文献[1]对目前的景深渲染算法做了分类和详细比较。累积缓存的方法[2]（accumulation buffer method）计算任务繁重，渲染效率低。周强[3]等人通过均值滤波技术实时的模拟了景深效果。但该算法容易造成强度泄露（intensity leakage）和模糊不连续性（blurring discontinuity）问题。Bertalmı´o 等人[4]采用各向异性分散（anisotropic diffusion）成功的解决了强度泄露问题，但计算弥散圆面积时复杂度急剧增加，不能够满足增强现实系统的实时性要求。文献[5,6]依据深度值把针孔成像图像分解成多个子图像，对每个子图像通过快速的傅里叶变换等进行模糊处理，在生成速度上很难满足增强现实的实时性要求。

收稿日期: 2010-08-19　　**修回日期:** 2011-01-04
基金项目: 国家科技支撑计划课题 (2006BAK13B10); 上海市国际科技合作基金项目 (09510700900); 上海市重点学科建设项目 (J50103); 上海市科学技术委员会资助项目 (115115034400).
作者简介: **赵东阳**(1984-)，男，河南南阳人，硕士生，研究方向为增强现实，多媒体技术；**陈一民**(1961-)，男，上海人，博士，教授，博士生导师，研究方向为增强现实与虚拟现实、网络与多媒体技术、机器人控制技术；**李启明**(1982-)，男，山东五莲人，博士生，研究方向为增强现实、多媒体技术; **刘燕**(1985-)，女，江苏南通人，硕士生，研究方向为虚拟仿真，机器人控制; **黄晨**(1985-)，男，江苏常州人，硕士生，研究方向为增强现实；**徐升**(1989-)，男，浙江东阳人，硕士生，研究方向为增强现实；**周明珠**(1985-)，女，湖南邵阳人，硕士生，研究方向为机器人。

由于实时性及计算过于复杂等原因，目前增强现实系统中，几乎未引入景深效果。另外景深渲染算法也几乎没有考虑亮度对景深有影响。因此，本文将大量的代数运算从 CPU 转移到 GPU，极大地减轻了 CPU 的负担，较好满足了增强现实系统实时性的要求，把景深渲染首次应用到增强现实系统，同时对场景渲染时考虑了亮度对景深的影响，弥补了先前景深渲染算法的不足，进一步提高增强现实系统的沉浸感。

1 系统结构

本文研发的视频透视式增强现实系统整体结构如图 1 所示。主要包括实时视频采集系统、虚拟场景生成系统、多通道交互系统、通讯模块系统和增强现实显示系统。本文研究的景深渲染算法主要体现在场景融合这一部分，正是展示平台研究开发中要解决关键问题之一。该系统软件结构如图 2 所示。由于先前国内外学者研究的景深渲染算法都是在虚拟现实的基础上做研究，而对于增强现实景深渲染方面的研究基本上没有，因此本文在增强现实景深渲染方面进行了相关的研究工作。图 2 中摄像机采集子系统实现对真实场景信息的采取，并依据真实场景信息建立三维空间坐标系；虚拟物体生成子系统生成虚拟物体；景深渲染子系统实现对场景的景深渲染。在先前的增强现实系统中，由于未引入景深，造成场景前后的物体都是清晰的，使得场景显得不够自然真实，缺乏沉浸感。加入景深，提高了场景内物体的深度信息和场景的真实性和沉浸感，较好的解决了虚实场景“无缝”融合。

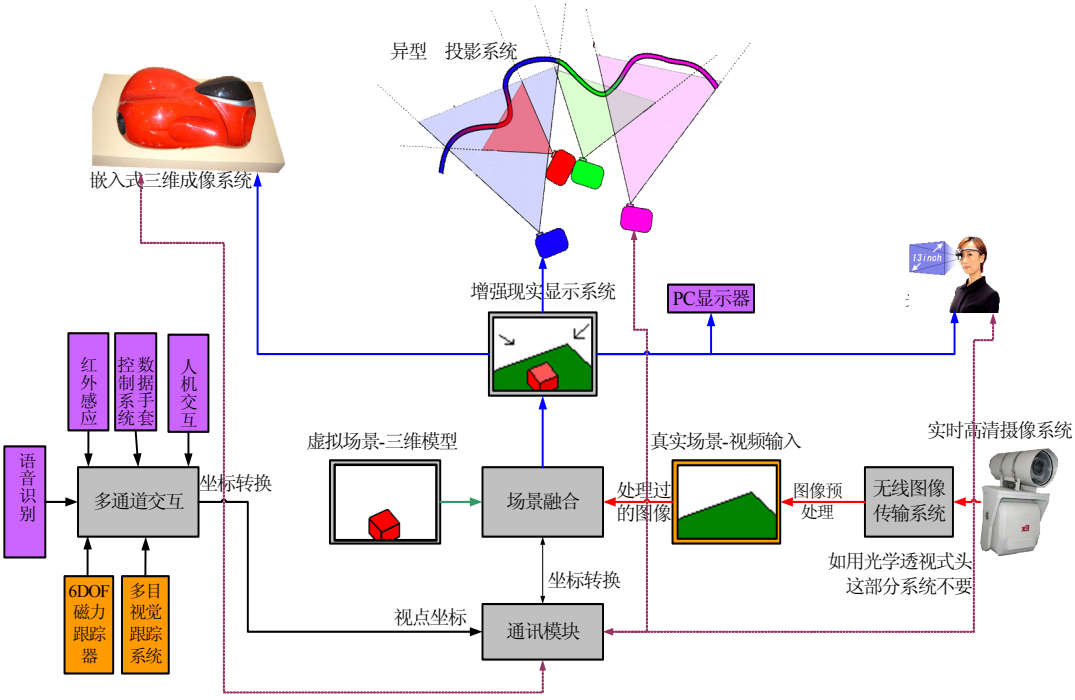

图 1 增强现实系统整体结构图

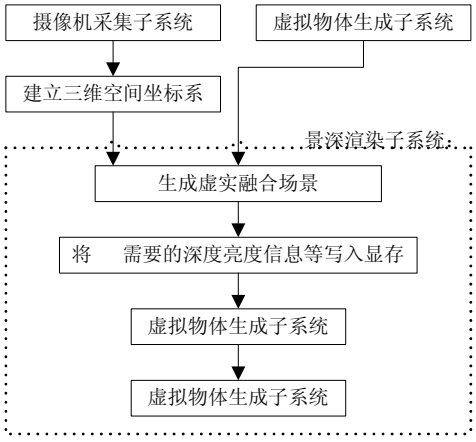

图 2 增强现实系统的软件结构

2 景深和弥散圆的计算

景深是人眼视觉系统中成像的重要特征。在摄像机系统中的聚焦面前后，光线开始聚集和扩散，点的影像逐渐模糊，形成一个扩大的圆，即弥散圆。在现实当中，观 拍摄的影像是以某种方式(比如放大成照片等)来观察的，人的肉眼所感受到的影像与放大倍率、投影距离及观看距离有很大的关系，如果弥散圆的直径小于人眼的鉴别能力，在一定范围内实际影像产生的模糊是不能辨认的，这个不能辨认的弥散圆就称为容许弥散圆。在焦点前后各有一个容许弥散圆，这两个弥散圆之间的距离就叫景深，即：在被摄主体(聚焦面)前后，其影像范围仍然有一段成像清晰，这段成像清晰的范围就是景深。在景深范围之内的物体都是清晰的，在景深范围

之外，物体要有梯度的模糊。

在现实中，成像系统把三维空间景物成像在二维的感光器件上，依据光学成像原理，聚焦准确的点在成像面上是理想的点，聚焦面前后的点在成像面上形成一个模糊圆，即弥散圆。由于人眼分辨率的限制，当弥散圆小于一定程度时人眼是无法辨认的，这个无法辨认的弥散圆就是容许弥散圆（permissible circle of confusion）。成像平面前后各有一个容许弥散圆，这两个弥散圆之间的距离成为焦深，焦深所对应的物平面的前后距离成为景深，如图 3 所示。

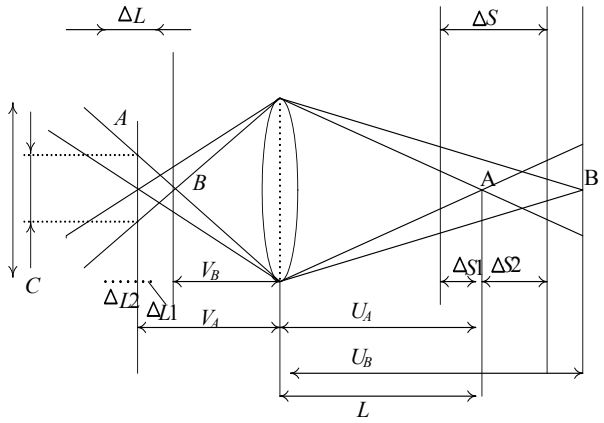

图 3 景深和弥散圆计算示意图

物点 A 经过透镜折射成像于成像面 A'点，物点 B 经过透镜在成像面上形成一个直径为 C 的弥散圆，透镜的焦距 F，直径为 D，由相似三角形的性质，则弥散圆直径 C 可用公式(1)计算:

$$C=\frac{V_A}{U_A}\frac{|U_B-U_A|}{U_B}D \tag{1}$$

根据透镜成像公式(2)可得公式(3)：

$$\frac{1}{f}=\frac{1}{u}+\frac{1}{v} \tag{2}$$

式中：u 为物距，v 为正焦相距，f 为透镜焦距。

$$C=\frac{F}{U_A-F}\frac{|U_B-U_A|}{U_B}D=\frac{F^2}{U_A-F}\frac{|U_B-U_A|}{NU_B} \tag{3}$$

式中：U_A 为 A 点的物距，U_B 为 B 点的物距，F 为透镜焦距，N 为光圈，$N=F/D$ 。因此可得景深的计算公式如下：

$$\Delta L1=\frac{NCL^2}{F^2+NCL} \tag{4}$$

$$\Delta L2=\frac{NCL^2}{F^2-NCL} \tag{5}$$

$$\Delta L=\Delta L1+\Delta L2=\frac{2F^2NCL^2}{F^4-N^2C^2L} \tag{6}$$

式中：$\Delta L1$ 为前景深，$\Delta L2$ 为后景深，ΔL 为景深，C 为容许弥散圆的直径，N 为光圈，L 为物距，F 为焦距。

3 基于亮度和深度信息的景深渲染算法

一幅具有景深效果的图像，其最大的特点是在聚焦面图像纹理清晰，在超过聚焦范围外的物体纹理模糊，事实上摄像机拍摄下来的具有景深效果的图像如图 4 所示，这几幅图像是 GC-655P-G 工业摄像机拍摄的，从图 4 可以看出，真实的景深效果图不仅和距离有关系，同时和像素的亮度有关系，反映在图像上即：在同等距离下，图像的模糊是从亮度高的像素点渗透到周围的像素点，周围的像素点亮度越低，渗透的效果越严重；图像的模糊和聚焦距离是成正比的关系，即离焦点越远的地方，模糊的效果越严重。因此，对于以前只考虑图像的深度信息的景深渲染算法，不能很好的适用于增强现实统。所以我们提出了一种基于图像亮度和深度信息的景深渲染算法。

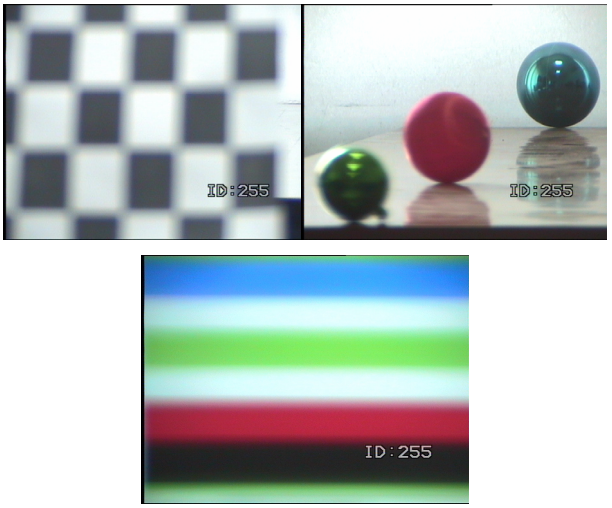

图 4 摄像机拍摄下景深图

3.1 真实场景信息采集及三维空间坐标建立

我们首先对所使用的摄像机进行标定，通过 OpenCV 自带的相机标定函数，可以求出所用相机转换矩阵为：

$$\begin{bmatrix} 990.21 & 0.00 & 337.80 \\ 0.00 & 981.31 & 180.23 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

因此，在现实场景中我们只要测得拍摄场景的物点到摄像机的距离，就可以通过转换矩阵计算出拍摄场景的深度信息和景深范围。这为以后虚实融合场景的渲染提供了重要数据，同时，通过 RGB 图像到 YUV 图像的转换矩阵，我们就可以求出图像的亮度信息，为下一步的景深渲染提供重要信息。

在试验中，摄像机的光圈为 11，允许的弥散圆半径取常用值 0.035mm，焦距为 200mm，经过测量物体到镜头的距离为 2.5m 左右，根据公式(4)(5)(6)，我们可求出摄像机的前景深 $\Delta L1$ 为 58.7mm,后景深 $\Delta L2$ 为 61.6mm，这样摄像机

的清晰范围就是 2.4413m 到 2.5616m，通过摄像机的转换矩阵，可以求出虚实融合场景中的景深是 126.746mm，试验系统通过抓取实时场景的特征点建立的三维坐标系如图 5 所示：

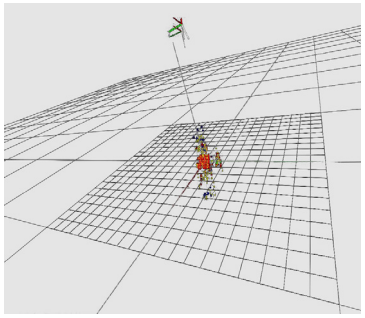

图 5　建立的三维坐标系

3.2 虚实融合场景的亮度和深度信息的获取

在增强现实系统中，透镜的焦距、光圈和聚焦面等参数以及摄像机拍摄的位置和方向是可以根据需要进行连续调节的，因此虚实融合场景中的虚拟物体各点的模糊度需要实时的计算，以便和真实的场景实现无缝融合。

设在真实场景中物体上点的坐标为 $pos(x_a, y_a, z_a)$，由 3.1 节得到的透射投影变换矩阵可知，物体上的点在光轴上的投影长度是 $L=z_a$，在对虚实融合场景渲染时，我们要把透镜的光圈 N，焦距 F，聚焦面 U，投影长度 L 等参数作为形参传递给渲染程序，物体上的点的弥散圆的直径由公式(3)可求得：

$$C_{blur} = \frac{F^2}{U - F}\frac{|L - U|}{NL} \tag{7}$$

真实场景中，设近剪截面是 z_{near}，远剪截面是 z_{far}，则物点的深度是 z，归一化为 $depth = z / z_{far}$，根据公式(3)，可以求出 c_{near} 和 c_{far}，由于成像最模糊的点必定位于这两个剪截面之一，故弥散圆的最大值 $c_{\max} = \max(C_{near}, C_{far})$，归一化模糊因子 $a_{blur} = c_{blur} / c_{max}$，同时根据第一步求出的投影变化矩阵，以及计算景深公式(4)、(5)、(6)，我们可以求出虚实融合场景中聚焦面的前后景深，而在这个景深范围之外的物体都是模糊的，且随着离聚焦面的距离越远，模糊的程度越明显。在渲染时，为了防止前景像素流入后景像素，把虚实融合场景的深度信息和模糊信息渲染到一张纹理里面，把虚实融合场景渲染到另外一张纹理里面，以供以后的处理使用，如图 6 所示，其中 a 里面存放是像素的亮度信息。

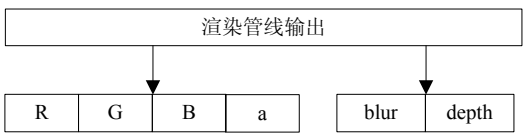

图 6　渲染管线输出

3.3 基于亮度信息的纹理采样模糊处理

处于景深范围之外的物点，经过透镜折射，在成像面上表现为一个模糊圈，在　幕上表现为多个像素组成的圆形区域，这是受到周围相邻区域像素影响的结果。因此，处于景深范围之外的物体各点，在　幕上成像表现为一个个模糊圈相互叠加而成为一个模糊区域。对模糊区域的像素，在针孔相机模型中，常采用平滑滤波处理而获得，常采用的平滑算子有高斯滤波、均值滤波等，这些方法没有考虑像素是如何相互影响的。事实上，真实的模糊效果不仅和物体的深度有关，还和物体的亮度有关，如图 4 所示。在模糊区域范围内，物体的模糊是从亮度高的点的颜色溢向亮度低的点的颜色，而不是像高斯滤波和均值滤波处理的那样，只考虑了周围像素点的影响，却没有考虑亮度的影响，依据事实，本文提出了一种基于亮度信息的模糊处理方法。模糊的依据是虚实融合场景的深度信息和亮度信息，亮度值高的块的颜色值流向亮度值低的块的颜色值，同时，依据深度值，离聚焦点近的地方，使之模糊度小，反之，使之模糊度大。

我们首先对虚实融合场景进行均值预模糊处理，在预处理过后，我们再考虑亮度对模糊的影响。由于物体模糊时颜色是从亮度高的点流向亮度低的点，因此，对每一个像素，我们求出其与周围八个像素的亮度差作为模糊的依据，如图 7 所示：记 $\Delta_i = color_1.light - color_i.light (2 \le i \le 9)$，则 Δ_i 是中心像素点（此处是①）与周围像素点的亮度差。如果 Δ_i 小于零，则说明像素①受到像素 i 的影响，如果 Δ_i 大于零，则说明不会受到像素 i 的影响。记 Δ_I 是所有负的 Δ_i 的和，则像素 i 对像素①的影响比例因子是：

$$\Delta = \begin{cases} |\Delta_i / \Delta_I|, (\Delta_i < 0); \\ 0, \quad (\Delta_i \ge 0); \end{cases} \tag{8}$$

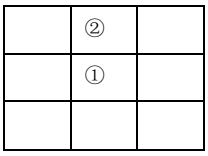

图 7　渲染影响示意图

在算法实现上，程序主要由 CPU 计算和 GPU 计算两部分组成。CPU 主要负责真实场景获取、建立三维空间坐标系和真实场景中物体各点弥散圆的直径的计算。GPU 主要负责虚实融合场景的融合和景深渲染。计算机在绘制图形时，必须为每个像素保存数据，包括颜色、深度值和亮度信息等。显示内存正是作为存储像素信息之用，在图形学中称为缓存。CPU 不能直接操作缓存中的数据，必须将 GPU 中的数据　贝到内存，处理完成后再将数据写入缓存，由于这需要大量的时间开销，所以必须尽可能减少 CPU 和 GPU 之间的通信。本系统使用 CUDA 对 GPU 进行编程，一个并行

化的程序会被许多个 thread 同时来执行。数个 thread 可以组成一个 Block,在同一 Block 中的 thread 可以使用同一份共享内存。数个 Block 可以组成一个 Grid。

图 8 为一个 block 的处理块。其中黄框内是数据区域，红虚线框内是一个 block 的数据区域，蓝区为待计算的像素点。绿框示意为一个 thread 的处理区域。以图 8 的设计方式计算一个 800x800 的数据规模，需要一个 grid，该 grid 包括 5000 个（100x50）block，每个 block 包括 128 个（1x128）thread，每个线程处理一个待计算的像素点。每 block 包括 5 块共享存储区域 ，总计 9728 字节，其中，存储像素点 RGB 信息的存储区为 4608（16x24x3x4）字节,存储像素点弥散圆半径的存储区为 1536（16x24x4）字节，存储像素点亮度信息的存储区为 1536 字节（16x24x4），存储待计算的像素点的计算结果为 1536（8x16x3x4）字节，存储待计算的像素点的中间值为 512（8x16x4）字节。

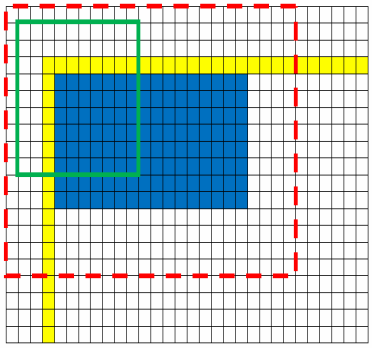

图 8 Block 示意图

3.4 图像的融合

经过 3.2 和 3.3 节处理，我们得到了一幅模糊的场景图和一幅清晰的场景图，直接利用 OpenGL 提供的 blend 功能对这两幅图像融合就可以得到具有景深效果的虚实融合三维场景图，融合公式如下：

$$color_i = \alpha \times blur + (1-\alpha) \times original \tag{9}$$

加权平均融合系数 α 取决于该点与聚焦平面的位置关系。α 的取值范围为 0~1.0，原则上离聚焦平面越远，α 的值就越大，即该点渲染过后的颜色值中其本身的颜色 A 占的比重就越小，相应的周边像素点占的比重就越大，看起来就越模糊。反之亦然。

公式(9)中，α 为归一化模糊因子。要注意的是，做图像融合时，在景深范围内的物点，由于人眼看起来是清晰的，因此我们可以适当的缩小一下模糊因子，在这里经过多次试验，我们把模糊因子变为原来的 2/3，同时为了不给用户造成图像突然变模糊的感觉，我们把景深边界周围的像素做了平滑模糊处理。融合前的图像如图 9 所示，渲染后的图像如图 10 所示，融合后的图像如图 11 所示。

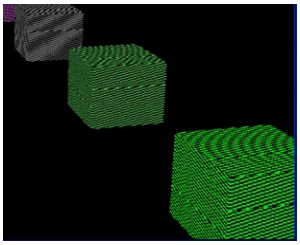

图 9 处理前的图像

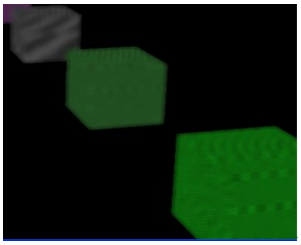

图 10 本文算法渲染后的图像

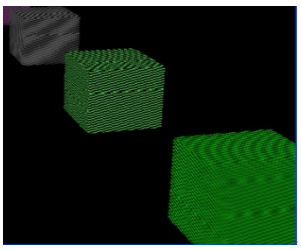

图 11 两者融合后的图像

4 模拟效果实例及实验对比

实验在此配置环境下进行（CPU：Intel Core Duo，主频：2.80GHz；内存：2G；显卡：Geforce 9600 GT，分辨率为 1280*1024；）。在虚拟现实系统中，帧速率可达到 48fps，在增强现实实验中，采用 GC-655P-G 工业摄像头和 V211 视频采集卡，帧速率可到达 21fps。经实验，如果采用 DirectShow 来获取视频，帧速率只有 16fps,当使用采集卡自带的 SDK 开发包，帧速率可以达到 25fps。如果采用 CPU 做渲染，帧速率是 14fps.而当我们采用 CUDA 对 GPU 编程时，帧速率可以达到 21fps。图 12 是采用高斯滤波处理的结果，图 13 是采用本文算法处理的结果，图 14 和图 15 是把本算法应用到增强现实系统的效果，从图 12 和图 13 对比的结果可以看出，在加入了亮度对景深的影响后，离焦区域内亮度高的点获得了更大的扩散空间，从而改进了以前的算法忽略了亮度信息这一不足，更真实的模拟了景深效果。图 14 是整个区域内全部在聚集区域的效果图，图 15 是真实球和虚拟球全部在聚焦区域外的效果图，其中后两幅图像中的蓝色球是由计算机虚拟生成。图 16 是摄像机拍摄的真实景深图像。从图 14、图 15 和图 16 的对比可以看出，提出的算法较好模拟了景深效果图。

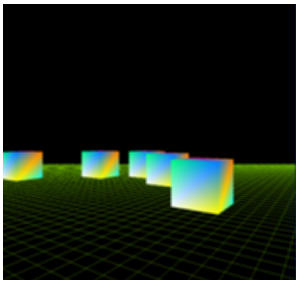

图 12 高斯算法效果图

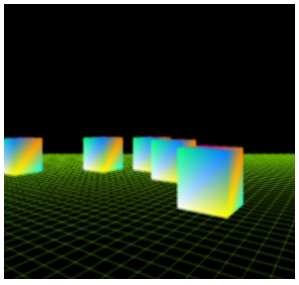

图 13 本文算法效果图

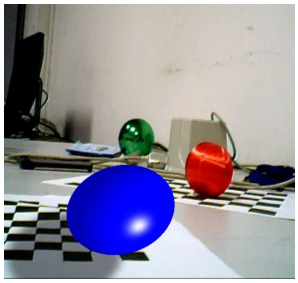

图 14　未加景深的效果

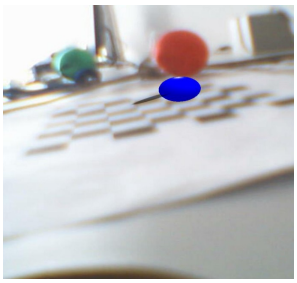

图 15　加入景深的效果

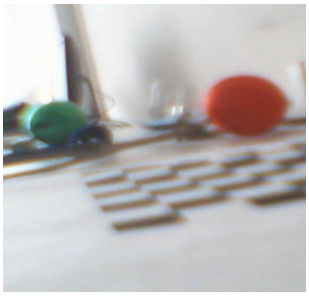

图 16　摄像机拍摄景深图

5　结论

经过研究前人的算法和　细分析真实摄像机成像的景深效果，针对先前的景深渲染算法忽略了亮度对景深的影响，本文提出把亮度信息考虑到景深效果模拟中，改进了以往只考虑深度对景深影响的不足，并且本文首次把景深渲染应用到增强现实系统，为提高用户的沉浸感，解决用户眼睛不适和真正实现虚实融合做出了一定贡献。实验结果表明：本文算法能较好的模拟真实摄像机拍摄的景深效果图，更适用于增强现实系统的应用。同时，由于把大量的浮点运算从 CPU 转移到了 GPU，在虚拟现实中，本文算法的帧速率达到 48fps，在增强现实中，本文算法的帧速率达到 21fps（实际需要 25fps），因此可较好满足增强现实实时性的要求。如何进一步提高增强现实系统的渲染速率，达到全帧速率是今后的研究重点。

参考文献：

[1] Ungkil Lee, Jounghyun Kim, Seungmoon Choir. Real-Time Depth of- Field Rendering Using Anisotropically Filtered Mimap Interpolation [J]. Visualization and Computer Graphics (S1077-2626), 2009, 15(3): 453-464.

[2] P Harebell, K Akeley. The Accumulation Buffer: Hardware Support for High-Quality Rendering [C]// Proc. ACM SIGGRAPH ’90. USA: ACM, 1990: 309-318.

[3] 周强, 彭俊毅, 戴树岭. 基于可编程图形处理器的实时景深模拟 [J]. 系统仿真学报, 2006, 18(8): 2219-2221, 2238. (ZHOU Qiang, PENG Jun-yi, DAI Shu-ling. Programmable GPU Based Real-time Simulation of Depth of Field [J]. Journal of System Simulation (S1004-731X), 2006, 18(8): 2219-2221, 2238.)

[4] M Bertalmı´o, P Fort, D Sa´nchez-Crespo. Real-Time, Accurate Depth of Field Using Anisotropic Diffusion and Programmable Graphics Cards [C]// Proc. Second Int’l Symp. 3D Data Processing, Visualization and Transmission (3DPVT’04). USA: IEEE, 2004: 767-773.

[5] S Lee, G J Kim, S Choi. Real-Time Tracking of Visually Attended Objects in Virtual Environments and Its Application to LOD [J]. Visualization and Computer Graphics (S1077-2626), 2009, 15(1): 6-19.

[6] M Kraus, M Strengert. Depth-of-Field Rendering by Pyramidal Image Processing [J]. Computer Graphics Forum (S1467-8659), 2007, 26(3): 645-654.

(上接第 1611 页)

[6] Higashi K, Nakasuka S, Sugawara Y, Sahara H. Thermal control of Panel Extension Satellite (PETSAT) [C]// 25th International Symposium on Space Technology and Science (2006-j-02). Japan: ISTS, 2006.

[7] Yoshiki Sugawaraa, Hironori Saharab, Shinichi Nakasukab, *et al.* A satellite for demonstration of Panel Extension Satellite (PETSAT) [J]. Acta Astronautica (S0094-5765), 2008, 63(1): 228-237.

[8] 刘俊, 林　宗, 刘小平, 等. ADAMS 柔性体运动仿真分析研究及应用[J]. 现代制造工程, 2004, (5): 53-55.

[9] 王成, 王效岳. 虚拟样机技术及 ADAMS [J]. 机械工程与自动化, 2004, 16(6): 66-70.

[10] 白争锋, 田浩, 赵阳. 基于虚拟样机的太阳帆板展开动力学仿真 [J]. 系统仿真学报, 2009, 21(13): 3976-3977. (BAI Zheng-feng, TIAN Hao, ZHAO Yang. Dynamics Simulation of Deployment of Solar Panels in Different Layouts Based on ADAMS [J]. Journal of System Simulation (S1004-731X), 2009, 21(13): 3976-3977.)

[11] 白争锋, 赵阳, 田浩. 太阳帆板故障模式展开动力学仿真[J]. 系统仿真学报, 2007, 19(13): 3067-3072. (BAI Zheng-feng, ZHAO Yang, TIAN Hao. Dynamics Simulation of Deployment of Solar Panels in Fault Modes [J]. Journal of System Simulation (S1004-731X), 2007, 19(13): 3067-3072.)

[12] 李金玉,　志践, 李　. 基于ADAMS的齿轮啮合过程中齿轮力的动态仿真[J]. 机械, 2005, 3(32): 15-17.