ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.6, June 2007, pp.1531−1542
DOI: 10.1360/jos181531

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 GPU 的实时深度图像前向映射绘制算法*

刘保权[1,3+], 刘学慧[1], 吴恩华[1,2]

[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 100080)

[2](澳门大学 科学技术学院 计算机和信息科学系,澳门)

[3](中国科学院 研究生院,北京 100049)

Real-Time Rendering Depth Images on GPU by Forward Warping

LIU Bao-Quan[1,3+], LIU Xue-Hui[1], WU En-Hua[1,2]

[1](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macao)

[3](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62522028, Fax +86-10-62563894, E-mail: lbq@ios.ac.cn, http://lcs.ios.ac.cn/~lbq/

Liu BQ, Liu XH, Wu EH. Real-Time rendering depth images on GPU by forward warping. *Journal of Software*, 2007,18(6):1531−1542. http://www.jos.org.cn/1000-9825/18/1531.htm

Abstract: This paper presents a new pipeline for rendering depth images entirely on GPU (graphics processing unit). The implementation exploits inherent parallelism of GPU to speed up the rendering of depth images. By the scheme, a novel forward 3D warping method is proposed for vertex shader to obtain high rendering performance. Furthermore, the hardware pipeline's rasterization function is utilized to conduct the image re-sampling efficiently to generate holes free rendering results. Per pixel lighting effect is implemented in pixel shader to get high image quality. The rendering shows rapid performance at full screen resolution, with correct self-occlusions and accurate silhouettes. Moreover, a real-time walkthrough system is implemented for the objects based on cylindrical depth image rendered by view-dependent dynamic LOD (level of detail) representation at runtime.

Key words: graphics hardware; GPU (graphics processing unit); real-time rendering; depth image; image-based rendering (IBR); per-pixel lighting

摘 要: 提出一种完全基于 GPU(graphics processing unit)的实时深度图像绘制流程.该方法利用 GPU 的并行计算特性对深度图像的绘制过程进行加速.推导出一种在 vertex shader 上进行的三维前向映射方法,对输入像素进行前向映射,以得到更高的绘制性能,并利用图形硬件流水线的光栅化功能高效地进行图像的插值重构,以得到连续无洞的结果图像.在 pixel shader 上进行逐像素的光照计算,生成高品质的光照效果.实验表明,该方法可以高速地进行满屏绘制,准确地保留物体轮廓信息和正确的遮挡关系.还实现了基于该方法的实时漫游系统.该系统能够实时地绘制多个基于柱面深度图像表示的对象,并能对其进行视相关的动态 LOD(level of detail)操作.

关键词: 图形硬件;GPU(graphics processing unit);实时绘制;深度图像;基于图像的绘制;逐像素光照

* Supported by the National Natural Science Foundation of China under Grant Nos.60473105, 60573155 (国家自然科学基金); the National Basic Research Program of China under Grant No.2002CB312102 (国家重点基础研究发展计划(973))

Received 2005-11-23; Accepted 2006-04-27

中图法分类号: TP391 文献标识码: A

纹理映射长期以来一直是现代图像绘制系统的一个重要部分,在增强场景真实感方面起着重要作用.纹理映射是通过模拟不同材质外表颜色来增强表面的二维细节,所以不能表示三维表面细节.而物体的三维表面细节可以通过深度图像来增强.深度图像是一个标量的距离场,可以表示物体表面的三维细节变化.然而,绘制深度图像远比纹理映射要困难得多,因为深度图像的绘制不允许像二维纹理映射那样直接逆映射,原因是多个深度采样点会映射到同一目标图像像素.所以,逆映射绘制算法都需要在参考图像空间进行搜索,以找到离目标视点最近的对应采样点.而这种搜索是非常耗时的.

随着现代图形硬件 GPU 的不断发展,一些新的硬件特征可以被用来进行绘制的加速,从而使真实感图形最终能够实现实时交互的绘制.正如 Macedonia[1]所说:“GPU 已经进入计算的主流.”我们的目标就是利用 GPU 的并行计算能力来加速深度图像的绘制过程,同时保持较高的真实感效果.这项研究可以充分挖掘硬件的新特性和新的应用前景.本文提出一种利用 GPU 的深度图像前向映射算法,对深度图像进行实时绘制.我们把深度图像视为一个规则的二维网格,每个参考像素都是一个网格点,在 vertex shader 上进行图像的前向映射.由于前向映射算法不是一对一映射,故映射结果会有不连续的空洞产生,我们利用图形硬件流水线的光栅化功能高效地进行图像的插值重构,以得到连续、无洞的高品质结果图像.

本文的方法是对 Relief Texture Mapping(RTM)[2−4]的一种扩展.与 RTM 方法一样,是通过绘制深度图像来表示三维物体的表面细节,如图 1 所示.RTM 方法的绘制瓶颈是软件实现的前向映射和图像插值重构,占了 94.1%的总计算时间.为此,本文的方法首先修改前向映射公式,使其容易在 GPU 上实现;然后,我们利用硬件流水线的光栅化功能来进行图像的插值重构,这样就解决了 RTM 方法的两个绘制瓶颈问题.我们的方法和 RTM 一样,也是一个两步的过程:先实行前向映射,再进行纹理映射,如图 2 所示.第 1 步对参考图像像素 X_s 进行平移,得到中间图像像素 X_i.我们的前向映射是在二维参考图像平面中进行的,中间图像和参考图像位于同一平面.第 2 步把中间图像作为纹理进行传统的纹理映射生成目标图像.

(a) Depth image (a) 深度图像 (b) Normal map (b) 法向图 (c) Color texture (c) 颜色纹理 (d) Rendering result (d) 输出图像

Fig.1 Input of our system is in (a), (b), and (c), the rendering result is shown in (d)

图 1 系统输入为(a),(b)和(c),输出图像为(d)

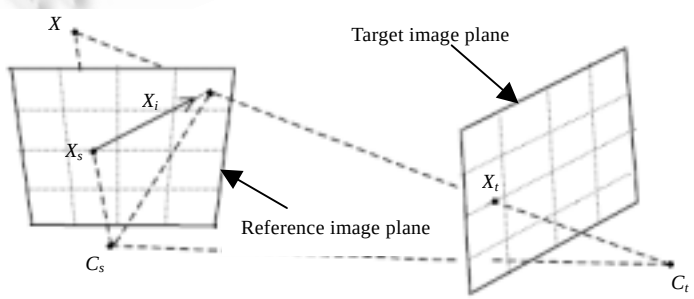

Fig.2 The two-step rendering procedure: Pre-warping followed by conventional texture-mapping

图 2 两步绘制过程:先实行前向映射,再进行纹理映射

为了对完整的三维物体进行绘制,我们用柱面深度图像来表达完整的三维物体,并扩展平面深度图像绘制算法,使之能够对柱面深度图像进行绘制.

本文的主要贡献包括:

1) 把深度图像的前向映射公式应用到 vertex shader 上,利用 GPU 的并行计算能力取得快速的绘制性能.
2) 在 vertex shader 中的前向映射由一个 2D 平移过程完成,所以不需要进行模视变换和投影变换等矩阵乘法,也不需要 Z buffer 测试,所以不会产生“顶点变换瓶颈”.
3) 图形硬件的光栅化操作被用来进行高效的图像插值重构,以产生连续无洞的结果图像.由于 RTM 方法的瓶颈在于使用软件方法进行图像插值重构,故我们解决了 RTM 方法的瓶颈.
4) 扩展平面深度图像绘制方法到柱面深度图像的绘制,以有效地表示完整的三维对象.实现了基于该方法的实时漫游系统,该系统能够对包含多个对象的场景进行实时绘制,并进行视点相关的动态 LOD(level of detail)控制.

本文第 1 节介绍相关工作.第 2 节描述平面深度图像的绘制算法.第 3 节扩展平面深度图像绘制方法到柱面深度图像的绘制.第 4 节给出实验结果及其与相关方法的比较.第 5 节对本文工作进行总结.

1 相关工作

Bump mapping[5]是通过扰动法向量来模拟物体的表面细节.由于实际表面并没有被改动,故无法正确地绘制自遮挡和物体轮廓,从而无法正确取得物体的三维视差效果.模拟物体复杂表面的最直接方法是建立表面细节的三维几何模型.Displacement mapping[6−8]就是通过把表面按照高度函数细分为大量的小三角形来进行绘制.Displacement mapping 改变了底层表面的实际几何结构,故可以产生正确的自遮挡、自阴影和物体轮廓.然而,Displacement mapping 要求绘制大量的小三角形,要对所有小三角形的顶点进行模视变换和投影变换等几何变换,而几何变换要通过 4×4 矩阵乘法来完成,所以顶点的几何变换成为绘制的瓶颈.虽然在本文的方法中也要用到很多小三角形,然而它们都在二维空间,不需要进行三维几何变换,只是在 vertex shader 中对每个顶点进行一个简单的参考图像平面内的二维平移.

Relief texture mapping(RTM)[2−4]与 parallax mapping[9]很相似,它们都通过图像的前向映射来模拟物体的三维视差效果.RTM 的前向映射和图像插值重构都是通过软件实现的,这形成了该方法的速度瓶颈.结果 RTM 方法很难进行实时绘制,虽然它能够产生高品质的结果图像(正确地处理自遮挡和自阴影,产生准确的物体轮廓).

Fujita[10]提出了一种硬件辅助的 RTM 方法.该方法可以把一些高级绘制效果,如 reflection mapping,添加到 RTM 绘制中.然而在绘制速度上,它和原始的 RTM 方法没有差别.因为它也是通过软件方法来实现前变换和图像的插值重构,故没有从本质上解决 RTM 方法的速度瓶颈,只是给 RTM 添加了一些高级绘制效果.

Relief mapping[11]是一种后向映射方法.它先把视线向量变换到纹理空间,然后通过在纹理空间进行线性搜索来定位视线与物体表面交点的粗略位置,最后再通过二分查找来定位交点的精确位置.线性搜索需要一个固定的搜索步长,为了不错过小的表面细节,必须减小步长、增加搜索次数,这样就难以提高绘制速度.文献[12,13]同样也是相似的后向映射方法,实际上都采用了线性搜索方法求取视线与物体表面的交点.

近年来,Donnelly[14]实现了一种基于硬件的逐像素位移映射方法,采用图形硬件中的三维纹理来加快后向映射的搜索速度.在预计算中,先生成一个三维距离场作为三维纹理,然后在绘制时查找距离场来减少搜索的迭代次数.该方法存储空间耗费大,难以处理大型复杂物体.

文献[15,16]把位移映射作为光线跟踪问题来解决.这种方法预计算所有位置上的所有方向上的视线相交信息,并存储在一个五维函数中,用 3 个位置变量和两个角度变量来进行索引.五维函数被预先压缩,运行时在 pixel shader 中进行解压缩.由于该方法要处理五维函数,所以要求巨大的预计算和存储量,难以处理复杂物体.

Dmesh[17]也是一种深度图像绘制算法.该方法先计算每个像素(i,j)的三维坐标 $P_{ij}\in R$;再三角化参考图像为一个三维网格(三维网格由大量的小三角形组成,小三角形的顶点即为 P_{ij});最后,通过绘制该三维网格得到目标图像.但是,绘制三维网格必须对每个顶点进行三维几何变换,而变换要通过 4×4 矩阵乘法来完成,所以,大量顶

点的几何变换成为绘制的瓶颈.为了提高绘制速度,Dmesh 方法基于某种误差准则,对原网格进行简化,生成了一个近似的简化网格,最后对该简化网格进行绘制.虽然简化网格比原网格的顶点少,所需要的变换也少,但是,这个粗糙的简化网格必然带来近似误差,因而难以表示丰富的表面几何细节.

2 基于 GPU 的单幅深度图像绘制算法

2.1 算法流程介绍

本节描述对参考深度图像在 GPU 上的绘制流程.绘制通过对参考图像进行基于新视点的前向映射来生成目标图像,运行时新视点可以交互式移动.算法的输入为参考深度图像和一个目标视点位置.输入的参考深度图像可以通过两种方法获得:(1) 用三维扫描仪对真实物体进行扫描得到深度图像;(2) 对计算机里的三维模型进行绘制所得到的 Z-buffer 即为深度图像.

根据新视点对参考深度图像进行绘制可以通过两遍来完成.在第 1 遍,参考图像像素 X_s 被映射到中间图像像素 X_i 处,这个映射是参考图像平面内的一个平移操作,它处理由于视点移动而产生的视差效果,我们把它实现在 vertex shader 中.由于原图像和中间图像的空间采样率不同,会在中间图像像素之间产生空洞,我们利用图形硬件流水线的光栅化功能来进行图像的插值重采样,以生成平滑无空洞的目标图像.逐像素光照计算在 pixel shader 得以实现,以生成高品质的光照效果.第 2 遍绘制非常简单,仅仅把第 1 遍的绘制结果作为纹理,映射到目标图像平面,以生成结果图像.图 3 显示了两遍绘制的流程图.

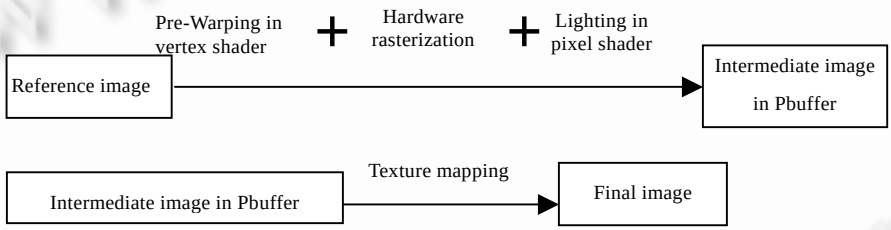

Fig.3 The flowcharts of the two rendering passes

图 3 两遍绘制的流程图

2.2 在vertex shader中进行前向映射

为了有效地在 GPU 上进行前向映射,首先需要把参考深度图像进行三角化.我们把参考深度图像看作是一个规则的二维网格,每个参考像素 X_s 都是一个网格顶点,这样就形成了一个规则的二维三角化网格.我们按照 McMillan 的遮挡兼容顺序[18,19]组织这些网格顶点进行绘制,这样就能保证不必进行深度比较,也能产生正确的遮挡关系.

然后,我们对 vetex shader 进行编程来实现对每个参考像素的前向映射.在 vetex shader 中,我们仅仅计算每个参考像素(u_s,v_s)在参考图像平面内的二维位移量$(\Delta u,\Delta v)$,位移后产生的中间图像是原图像在二维平面内的一个扭曲变形版本,中间图像像素可表达为$(u_i,v_i)=(u_s+\Delta u,v_s+\Delta v)$.有了这个位移量就能保证从目标视点看到的中间图像和把原图像直接进行 3D warping 所产生的目标图像是一致的.位移的计算方法采用 Image-Based Rendering using Image Warping,原理及其效果见参考文献[2,18].

我们沿着外极线来计算位移量$(\Delta u,\Delta v)$.图 4 表示在水平方向上对原像素 u_s 所做平移的平移量$\Delta u=u_i-u_s$.图 4 中,三角形 C_tFX 和三角形 u_iu_sX 是相似三角形,所以有

$$\Delta u/D_e=depth/(depth+Dt).$$

其中,D_e 是 u_e 到 u_s 的距离,$D_e=u_e-u_s$;D_t 是从目标视点 C_t 到原图像平面的距离,对一次绘制的所有参考像素来说,D_t 为常量;u_e 是外极点(epipolar)的水平方向坐标,u_su_e 为外极线;$depth$ 是参考像素 u_s 的深度.这样,我们就可以计算水平位移量Δu:

$$\Delta u=(u_e-u_s)\times depth/(depth+D_t)=(u_e-u_s)/(1+D_t/depth)=(u_e-u_s)\times Table(depth),$$

其中,$Table(depth)=1/(1+D_t/depth)$可以被预计算,并作为vertex texture传给vertex shader.用同样的方法可以计算出竖直方向的平移量Δv.

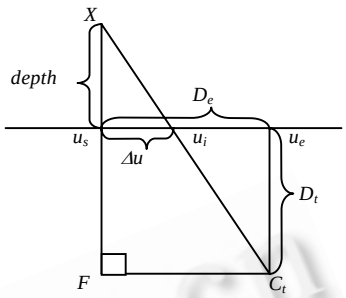

Fig.4 Top view of depth image pre-warping

图 4 水平方向上的位移计算

由此很容易看出本文的方法比 Dmesh 方法要快.这是因为我们仅仅在二维平面内对每个顶点作平移,而不是用矩阵乘法来作三维顶点变换.而且,我们也不需要在三维网格绘制中作必不可少的硬件深度比较(Z-Buffer 算法),这也会减少计算量,提高绘制速度.

在第 1 遍绘制过程中,我们关闭硬件深度比较功能;而在第 2 遍绘制中,我们再打开它,以便能够进行多幅深度图像的同时绘制,或深度图像和几何物体的混合绘制.我们可以很容易地计算出目标图像所对应的 Z-Buffer,即目标像素的正确深度值为 $Z_t=D_t+depth$.在第 1 遍绘制中,我们在 vertex shader 中计算出 Z_t,并把它作为顶点属性值传给 pixel shader,再输出到 Pbuffer 中,并在第 2 遍绘制中进行深度比较.有了正确的目标像素深度值 Z_t,就可以解决多幅深度图像同时绘制,或深度图像和几何物体在混合绘制时的相互遮挡问题(后文图 17 即为深度图像被基于三角形表示的茶壶所贯穿时的绘制效果).有了目标像素正确深度值的另一个好处是,可以用 shadow map 方法来投影阴影到结果图像中,以产生带阴影的绘制效果.

综上所述,我们对二维规则网格的每一顶点(u_s,v_s)做平移产生(u_i,v_i),这个平移是在原图像平面内的位移.由于各顶点位移量各不相同,于是我们就得到一个经过变形了的非规则二维网格,它将要被图形硬件光栅化成平滑无洞的中间图像.

2.3 基于图形硬件的图像重构

离散图像的空间变换包括两个过程:像素坐标变换和在输出图像像素网格上的图像重采样.前一节描述了像素坐标变换过程,即计算了参考像素的平移量.本节将讨论如何对图像进行插值重构,以解决由于原图像和目标图像采样率不同而引入的目标图像不连续(有空洞)问题.

我们利用标准图形硬件流水线的光栅化功能高效地进行图像的插值重构.图形硬件将对 vertex shader 所产生的非规则二维三角形网格进行光栅化,即用线性插值的方法来计算三角形内部所有采样点的属性值.很重要的一个属性值就是纹理坐标,它将被用来在 pixel shader 中对颜色纹理和法向纹理进行索引,以完成光照计算.图形硬件的光栅化功能可以确保所生成的目标图像平滑无洞.

2.4 在pixel shader中进行逐像素的光照计算

我们利用现代图形硬件的可编程性,在 pixel shader 中进行编程实现逐像素的 Phong 光照计算.如图 1 中的一个颜色纹理和一个法向纹理可以被纹理坐标进行索引,以产生高品质的光照效果,如图 1(d)所示.对每一像素片断的光照,计算过程如下:

1) 在纹理空间计算视线向量(VD),即从目标视点到像素片断的向量.把目标视点 C_t 投影到参考图像平面上,得到它的二维纹理坐标(u_e,v_e),和它到参考图像平面的距离:D_t.C_t 的三维纹理坐标就是$(u_e,v_e,-D_t)$.像素片断的三维纹理坐标是$(u_s,v_s,depth)$.所以,纹理空间的视线向量就是 $VD=(u_e,v_e,-D_t)-(u_s,v_s,depth)$.
2) 用像素片断的纹理坐标对颜色纹理和法向纹理进行索引,得到表面的属性值:漫反射颜色和法向量.
3) 使用视线向量和表面属性值在纹理空间进行 Phong 光照计算,得到该像素片断的最终颜色值.

由于我们的方法是一种前向映射方法,所以,在 pixel shader 中不需要任何搜索操作,使得 pixel shader 程序简短、快捷.而 Policarpo[11]和 Donnelly[14]是后向映射方法,需要在 pixel shader 中进行迭代搜索操作,以找到目标像素所对应的原像素,可见,后向映射方法的复杂度要远大于我们的方法.

2.5 第2遍绘制:纹理映射中间图像以生成结果图像

我们把第 1 遍绘制结果(即中间图像)存储在一个 Pbuffer 中,它是位于显卡本地的存储器上,不需要再读回主内存.然后,在第 2 遍绘制中直接把它作为纹理,映射到参考图像平面上的一个合适大小的矩形上,就得到最终的绘制结果.这个平面纹理映射过程实现了旋转、缩放和最终的滤波操作.生成的结果图像是遮挡正确的,并有准确的视差效果,使得移动的观察者能够感知到三维表面细节的正确起伏.绘制结果如图 1(d)和图 17 所示.

3 用柱面深度图像绘制整个物体

3.1 建 模

至此,我们已经描述了平面深度图像的绘制流程.本节我们将描述如何用这种两遍的绘制流程来对整个物体进行绘制,并实现一个实时漫游系统.我们采用柱面深度图像来对三维物体进行表达,使用与第 2 节相同的绘制流程,仅仅对映射公式略作修改.由于柱面深度图像与平面深度图像的数据结构一样都是一个简单的二维数组,故映射公式也很相似.

很多研究人员已经使用基于图像的技术来表示复杂形状物体[4,20].基于图像的绘制方法都是企图从一系列采样集来重构全光函数.采样集可以是一系列的参考图像,这些参考图像被进行映射,再合并,从而得到关于场景的任意视点的绘制结果.这些参考图像可以是普通的平面图像,也可以是球面、柱面等曲面图像.

正方形 6 个面上的 6 幅平面参考图像,由于存储简单(每幅平面图像都是一个二维数组)、处理方便,被广泛用来进行基于图像的建模(如 RTM 方法就采用正方形 6 个面上的参考图像对三维复杂物体进行建模).但是,这种方法并不是一种均匀采样方法,它在正方形的棱和角处都是非均匀采样.所以,RTM 方法不得不在绘制时对正方形相邻面之间进行多次的 pre-warping,否则就会在棱和角处产生空洞,最多需要进行 9 次这样的 pre-warping,从而严重地影响了 RTM 方法的绘制速度.为此,我们选择用柱面深度图像对物体进行建模,其优点是,柱面图像也可以像平面图像那样用二维数组来表示,同时在圆柱侧面上是完全的均匀采样.圆柱方案的最大问题是如何有效地表示上下视域的模型图像以及如何天衣无缝地与周边图像进行缝合过渡,我们将在第 3.3 节解决这个问题.

我们根据圆柱面参数方程的特点,设计了同时具有透视和平行两种投影方式的柱面相机模型,并称之为柱面双投影相机 DPCC(double projective cylindrical camera),它的投影面为圆柱面.在水平方向采用透视投影方式,光线交于投影中心一点(在圆柱的轴线上),如图 5(a)所示;而在竖直方向上采用平行投影方式,光线互相平行,如图 5(b)所示.

(a) Top view (perspective projection) (b) Side view (parallel projection)
(a) 俯视图(透视投影) (b) 侧视图(平行投影)

Fig.5 Double projective cylindrical camera

图 5 柱面双投影相机

这种 DPCC 模型不同于我们熟悉的柱面针孔相机模型,如柱面全景图相机模型[19](图 6(a)所示为一个用针孔相机模型产生的柱面全景图).并且,这种相机模型在现实世界中并不存在,但是,它却使得基于图像的绘制在存储和计算上都更加简单有效.

我们把具有这种投影方式的图像称为柱面双投影图像,简写为 DPCI(double projective cylindrical image).我们可以很容易地用圆柱面参数方程把它表示成二维数组,以便对三维物体进行建模.如图 6(b)所示为一个由 67 170 个三角形组成的三维物体的 DPCI.DPCI 的获取可以从三维物体包围盒上的 6 幅平面深度图像,在预处理阶

段经过 warping 得到.

DPCI 的每个像素 u_s 都有一个颜色值和一个深度值,还可以有一个纹理坐标来索引其他属性值.它的深度值如图 7 所示,C 是柱面中心;u_s 是像素坐标,其对应物体表面点为 B;像素 u_s 的深度值就是线段 u_sB 的长度 $depth$.

(a) Image projected by cylindrical pinhole-camera model
(a) 柱面全景图

(b) DPCI projected by DPCC
(b) 柱面双投影图像(DPCI)

Fig.6 Comparison of different cylindrical images
图 6 不同柱面图像的比较

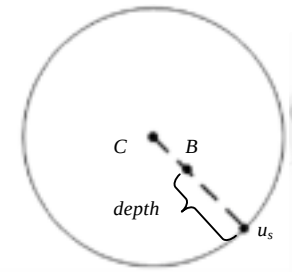

Fig.7 The pixel's depth value of DPCI
图 7 DPCI 的像素深度值

3.2 对DPCI的绘制算法

本节描述如何对 DPCI 进行绘制,以生成新图像.绘制流程仍然是一个两遍算法:第 1 遍进行前向映射和像素着色;第 2 遍是一个普通的纹理映射操作,其流程如图 8 所示.

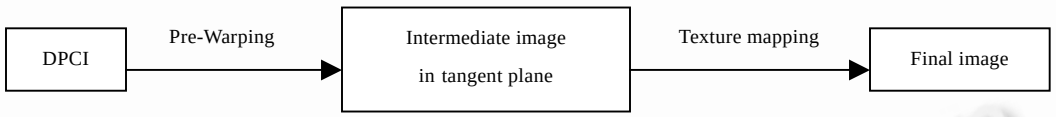

Fig.8 The flowchart of the two-pass rendering
图 8 两遍绘制算法流程图

图 9 是绘制过程的一个二维解释.由于柱面到柱面的映射公式比柱面到平面的映射公式要复杂,所以,我们选择把柱面图像前向映射到柱面的切平面上来,以提高绘制速度.切平面与柱面的交点即切点为 E,E 其实就是外极点(epipolar).前向映射的结果是在切平面上生成一个中间图像;在第 2 遍绘制中,再把这个中间图像作为纹理,用普通的纹理映射方法映射到目标图像平面,生成最终图像.

如图 10 所示,从 DPCI 到切平面的映射就是要从原像素 X_s 计算出切平面上的中间像素 X_i 的过程,使得从目标视点看到的中间图像和把原图像直接进行 3D warping 所产生的目标图像是一致的.也就是说,物体表面点 B、中间像素 X_i、目标像素 X_t 和目标视点 C_t 要 4 点共线.所以,中间像素 X_i 就是直线 C_tB 与切平面的交点,线段 X_sB 的长度就是原像素 X_s 的深度值,中间像素 X_i 可以很容易地根据图中的相似三角形关系求出来.

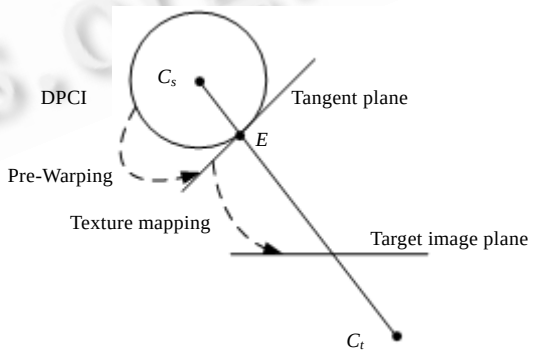

Fig.9 2-D illustration of the rendering procedure
图 9 绘制过程的二维解释

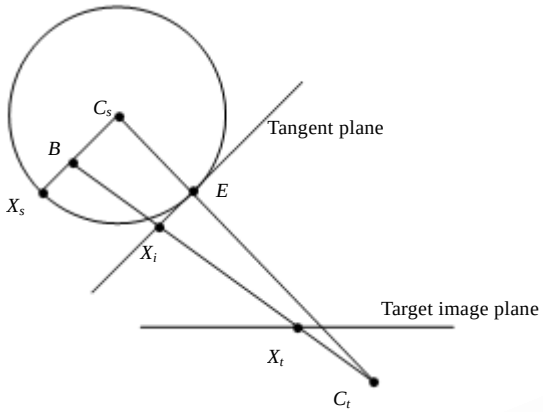

Fig.10 DPCI to tangent plane pre-warping

图 10 DPCI 到切平面的映射

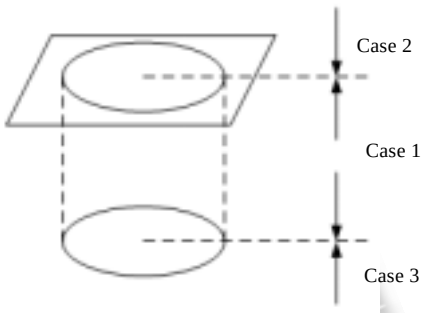

Fig.11 Three cases of the viewpoint's position

图 11 目标视点位置的 3 种情况

3.3 解决竖直方向视域问题

用柱面作投影面进行建模的一个缺点在于柱面的上、下两个底面处所引入的边界条件.因为,当目标视点位于柱面上方或下方时,不能绘制出完整的物体图像,如图 12(a)所示. 这时,根据视点位置,我们选择再绘制两个底面其中之一,以使绘制结果完整无误,这样就解决了可任意放置目标视点的问题.两个底面仅仅用普通的平面深度图像表示,我们可以计算中间图像对应的正确 Z-Buffer 值(记录每个中间像素到视点的距离),故很容易在第 2 遍绘制时打开硬件深度测试功能,把 DPCI 与上下底面深度图像所产生的中间图像,通过 Z-Buffer 的深度测试功能正确地合成起来(即仅保留距离视点最近的中间像素),从而不会产生拼接问题.如图 11 所示,目标视点有 3 种情况需要分别处理:

Case 1. 视点在柱面的高度范围内时,仅绘制一个 DPCI 就够了,可以得到完整的绘制结果;

Case 2. 视点在柱面的上方时,绘制一个 DPCI,再绘制一个上底面深度图像就够了.图 12 给出了绘制结果;

Case 3. 视点在柱面的下方时,绘制一个 DPCI,再绘制一个下底面深度图像就够了.

可见,在所有情况下,最多只需要绘制两个深度图像:一个是 DPCI;另一个是上、下两个底面其中之一.

(a) Rendered by a DPCI
(a) 仅绘制 DPCI 的结果

(b) Rendered by a top planar depth image
(b) 仅绘制上底面深度图像的结果

(c) The composition of the first two
(c) 对前两个的合成结果

Fig.12 Rendering DPCI with a top end cap

图 12 视点在柱面上方时的绘制过程

3.4 在DPCI上合成微结构

我们可以在 DPCI 图像上合成微结构(meso-structure),从而生成一些特效,如电影《指环王 III》中的"树人(tree man)".我们在两个尺度上表示物体几何形状:宏观结构用 DPCI 来表示大致的整体形状,而表面局部几何细节的微观结构用另一个深度图像作为纹理来表示.当我们绘制 DPCI 时,在 pixel shader 中采样这个微结构深度

纹理图像,并用采样结果来调制 DPCI 像素的深度值,然后再对调制结果做前向映射操作,生成最终图像.绘制结果可以准确地表现物体轮廓线和宏观、微观两个尺度上的几何形状,结果如图 13 所示.

Fig.13 Meso-Structure added to DPCI

图 13 在 DPCI 上合成微结构的绘制效果

3.5 视点相关的动态LOD表示

由于 DPCI 的数据结构是一个 $N\times N$ 的二维规则图像,所以很容易在运行时得到动态的多分辨率表示.通过使用动态 LOD 技术,我们可以实时地绘制很多个物体.对那些离视点较远的物体,我们用较低的分辨率进行绘制,不但减少了计算量,还能保证图像品质不受影响.

对场景中每一个用 DPCI 表示的物体,我们计算从视点到物体中心的距离 $Dist_{object2view}$,然后根据这个距离计算一个采样间隔 $Iter_{step}$ 对原 DPCI 进行采样,得到一个低分辨率的 DPCI,我们用这个低分辨率的 DPCI 来绘制原物体.$Iter_{step}$ 的计算方法是,由 $Dist_{object2view}$ 计算出一个目标像素在原图像中的投影面积 A,用这个投影面积 A 所覆盖的原像素个数的平方根作为采样间隔 $Iter_{step}$.当物体很近时,$Iter_{step}$ 等于 1,此时达到最高分辨率为,$N\times N$;而当物体较远时,较粗的分辨率为 $Nd\times Nd$,其中 $Nd=N\div Iter_{step}$,$Nd<N$,物体越远,Nd 越小.

我们实现了一个基于 DPCI 的实时漫游系统.该系统采用动态 LOD 技术进行全屏显示,可对很多个基于 DPCI 的物体进行实时绘制,绘制结果如图 14 所示.除了天空和地面以外,其他物体都采用基于 DPCI 的表示.

Fig.14 Real-Time walk through system in a scene full of image-based objects at 48.327 fps

图 14 基于 DPCI 的实时漫游系统(48.327 fps)

4 实验结果与讨论

我们用 OpenGL 和 Cg 在 Windows XP 桌面系统实现了本文的方法.系统的硬件配置为 Pentium IV 2.4 GHz 和 512M 主内存,显卡采用 nVidia GeForceFx5200.我们与基于多边形的绘制方法进行了比较,绘制速度见表 1,成像质量如图 15 所示.两种方法都采用了背面剔除和三角形条带方法,都未使用绘制列表,故本文所作的性能比较是公平的.

Table 1 Comparison of frame rate between DPCI based rendering and traditional polygon based rendering
表 1 本文的方法与基于多边形方法的绘制速度比较

Object	Polygon number	Resolution of DPCI	DPCI based rendering (fps)	Polygon based rendering (fps)	Ratio of fps
Venus	67 170	512×512 256×256	67.514 185.795	32.351	2.087 5.743
Man's head	78 157	512×512 256×256	67.514 185.795	28.346	2.382 6.554

(a) Traditional polygon method (b) DPCI based method (input DPCI is 256×256) (c) DPCI based method (input DPCI is 512×512)
(a) 普通的基于多边形的绘制 (b) 基于 DPCI 的绘制结果(分辨率 256×256) (c) 基于 DPCI 的绘制结果(分辨率 512×512)

Fig.15 Comparison of the rendering quality
图 15 绘制品质的比较

从以上绘制速度和成像质量的比较可以看出:本文的方法与基于多边形的绘制在成像质量上相仿;而在绘制速度上,本文的方法要比基于多边形的方法快很多倍.

条纹理映射[21]也是一种深度图像绘制方法,它采用直线段作为元语进行绘制.但是,图形硬件对点和线的光栅化效率要远远低于多边形.实验结果表明:本文的方法比该方法快约两倍.

我们还把本文的平面深度图像绘制与其他几种方法进行了比较(其中,Per-Pixel displacement mapping[14]和 relief mapping[11]为后向映射方法,而 DMesh[17]和本文的方法为前向映射方法),测试结果见表 2,输出分辨率都设成全屏模式:1024×768.实验结果说明,本文的方法要快很多.本文的方法平面深度图像绘制品质如图 16 所示,由于输入图像的分辨率都是 512×512,所以,图 16 中的不同图像的绘制速度相仿.

Table 2 Comparison of frame rate for planar depth image rendering
表 2 平面深度图像绘制的速度比较

Resolution of input image	Per-Pixel displacement mapping[14] (fps)	Relief mapping[11] (fps)	DMesh[17] (fps)	Our method (fps)
512×512	18.36	5.85	17.472	68.14

如图 17 所示为 DPCI 和基于多边形物体的混合绘制结果.场景中,茶壶和房间为标准的基于多边形的绘制;而人的头像为基于 DPCI 的绘制.绘制结果说明,基于多边形物体在 DPCI 中贯穿时,遮挡关系处理得很正确.

与文献[2,3]相同,DPCI 方法较适于与圆球同胚物体的绘制;而对于存在孔洞的拓扑复杂模型,在某些视点位置,其绘制结果可能会产生错误.一种解决方法就是先把拓扑复杂模型分割成多个与圆球同胚的简单模型,再对这些简单模型进行绘制,就能产生正确的绘制结果.

更多的绘制效果演示动画可以访问 http://lcs.ios.ac.cn/~lbq/Publications.htm.

5 结 论

本文提出了一种新的基于 GPU 的深度图像实时绘制流程.利用现代图形硬件的可编程性和并行性来加速深度图像的前向映射过程,还利用了硬件的光栅化功能来完成图像的插值重构.我们的绘制流程解决了 RTM 方法的速度瓶颈问题.实验结果说明,本文的方法可以生成连续、无洞的高品质绘制结果,并且成像速度要比最新

的几种深度图像绘制方法快很多.本文方法的应用包括基于深度图像的大规模实时漫游系统和游戏系统(如图 18 所示).

Fig.16 Rendering results of planar depth image

图 16 本文方法平面深度图像绘制品质

Fig.17 The hybrid rendering results of DPCI interpenetrated with standard triangle based geometry teapots

图 17 DPCI 和基于多边形物体的混合绘制结果

Fig.18 A full screen walking-through system based on image-based objects at 41.716 fps

图 18 基于深度图像的实时大规模全屏漫游系统(41.716 fps)

References:

[1] Macedonia M. The GPU enters computing's mainstream. IEEE Computer Society, 2003,36(10):106−108.

[2] Oliveira MM. Relief texture mapping [Ph.D. Thesis]. North Carolina: University of North Carolina, 2000.

[3] Oliveira MM, Bishop G, McAllister D. Relief texture mapping. In: Whitted T, ed. Proc. of the SIGGRAPH 2000. New Orleans: ACM Press, 2000. 359−368.

[4] Oliveira MM, Bishop G. Image-Based objects. In: Proc. of the '99 ACM Symp. on Interactive 3D Graphics. Atlanta: ACM Press, 1999. 191−198.

[5] James B. Simulation of wrinkled surfaces. Computer Graphics (Proc. of the SIGGRAPH'78), 1978,12(3):286−292.

[6] Cook RL. Shade trees. Computer Graphics (Proc. of the SIGGRAPH'84), 1984,18(3):223−231.

[7] Hirche J, Ehlert A, Guthe S, Doggett M. Hardware accelerated per-pixel displacement mapping. In: Proc. of the Graphics Interface. Waterloo: ACM Press, 2004. 153−158.

[8] Kautz J, Seidel HP. Hardware accelerated displacement mapping for image based rendering. In: Proc. of the Graphics Interface 2001. Toronto: Canadian Information Processing Society, 2001. 61−70.

[9] Tomomichi K, Takahei T, Inami M, Kawakami N, Yanagida Y, Maeda T, Tachi S. Detailed shape representation with parallax mapping. In: Proc. of the 11th Int'l Conf. on Artificial Reality and Telexistence (ICAT 2001). Tokyo: ACM Press, 2001. 205−208.

[10] Fujita M, Kanai T. Hardware-Assisted relief texture mapping. In: Navazo I, ed. Proc. of the Eurographics 2002. Saarbrücken: Eurographics Association, 2002. 257−262.

[11] Policarpo F, Oliveira MM, Comba JLD. Real-Time relief mapping on arbitrary polygonal surfaces. ACM Trans. on Graphics (Proc. of the SIGGRAPH I3D 2005), 2005,24(3):155−162.

[12] Yerex K, Jagersand M. Displacement mapping with ray-casting in hardware. In: Proc. of the SIGGRAPH 2004. Los Angeles: ACM Press, 2004.

[13] Delgass LL, McGhee K. Parallax searching and mesosurface shadowing. In: Proc. of the SIGGRAPH 2005. Los Angeles: ACM Press, 2005.

[14] Donnelly W. GPU Gems. 2nd ed., Addison-Wesley, 2005. 123−136.

[15] Wang LF, Wang X, Tong X, Lin S, Hu SM, Guo BN, Shum HY. View-Dependent displacement mapping. ACM Trans. on Graphics (Proc. of the SIGGRAPH 2003), 2003,22(3):334−339.

[16] Wang X, Tong X, Lin S, Hu SM, Guo BN, Shum HY. Generalized displacement maps. In: Proc. of the Eurograph Symp. on Rendering 2004. Norrköping: Eurographics Association, 2004. 227−234.

[17] Pajarola R, Sainz M, Meng Y. DMesh: Fast depth-image meshing and warping. Int'l Journal of Image and Graphics (IJIG), 2004, 4(4):1−29.

[18] McMillan Jr. L. An image-based approach to three-dimensional computer graphics [Ph.D. Thesis]. University of North Carolina, 1997.

[19] McMillan L, Bishop G. Plenoptic modeling: An image-based rendering system. Computer Graphics (Proc. of the SIGGRAPH'95), 1995, 39−46.

[20] Adelson EH, Bergen J. The plenoptic function and the elements of early vision. In: Computational Models of Visual Processing. Cambridge: MIT Press, 1991. 3−20.

[21] Li KY, Wang WC, Wu EH. Bar texture mapping. Journal of Software, 2004,15:179−189 (in Chinese with English abstract).

附中文参考文献:

[21] 李奎宇,王文成,吴恩华.条纹理映射.软件学报,2004,15:179−189.

刘保权(1975－),男,陕西西安人,博士生,主要研究领域为计算机图形学,可视化,虚拟现实.

刘学慧(1968－),女,博士,副研究员,CCF高级会员,主要研究领域为计算机图形学,虚拟现实.

吴恩华(1947－),男,研究员,博士生导师,CCF高级会员,主要研究领域为计算机图形学,可视化,虚拟现实.