

基于 GPU 的球面深度图实时绘制

朱 鉴¹⁾ 吴恩华^{1),2)}

¹⁾(澳门大学科技学院电脑与资讯科学系电脑图形及多媒体实验室 澳门)

²⁾(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)

摘 要 提出了一种 GPU 加速的实时基于图像的绘制算法. 该算法利用极坐标系生成对物体全方位均匀采样的球面深度图像; 然后根据推导的两个预变换公式将单幅球面深度图像预变换到物体包围球的一个与视点相关的切平面上, 以生成中间图像; 再利用纹理映射生成最终目标图像. 利用现代图形硬件的可编程性和并行性, 将预变换移植到 Vertex Shader 来加快绘制速度; 利用硬件的光栅化功能来完成图像的插值, 以得到连续无洞的结果图像. 此外, 还在 Pixel Shader 上进行逐像素的光照以及环境映射的计算, 生成高质量的光照效果. 最终, 文章解决了算法的视点受限问题, 并设计了一种动态 LOD(Level of Details)算法, 实现了一个实时漫游系统, 保持了物体间正确的遮挡关系.

关键词 GPU(图形处理单元); 球面深度图像; 基于图像的绘制; 预变换

中图法分类号 TP391

DOI号: 10.3724/SP.J.1016.2009.00231

GPU Based Real-Time Rendering of Spherical Depth Image

ZHU Jian¹⁾ WU En-Hua^{1),2)}

¹⁾(Computer Graphics and Multimedia Laboratory, Department of Computer and Information Science,
Faculty of Science and Technology, University of Macau, Macao)

²⁾(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190)

Abstract This paper presents an image-based real-time rendering algorithm on GPU. By the algorithm, we uniformly sample an object in all directions in a polar coordinates system, to construct a Spherical Depth Image. With two deduced Warping Equations, we then pre-warp the image onto a view-dependent tangent plane of the bounding sphere of the object to get an intermediate image, which can be further rendered onto the target image plane using standard texture mapping. By exploiting the inherent parallelism of modern programmable GPU, we transport the pre-warping process into Vertex Shader. Furthermore, the hardware pipeline's rasterization function is utilized to conduct the image re-sampling efficiently to generate hole-free rendering results. Besides, Per-pixel lighting and Environment mapping effect is implemented in Pixel Shader to get high quality lighting effect. Finally, we solve the problem of limited field of view, and design a LOD (Level of Detail) algorithm to implement a real-time walkthrough system, in which all the rendered objects are occluded by each other correctly.

Keywords GPU(Graphics Processing Unit); spherical depth image; image-based rendering; pre-warping

收稿日期:2008-07-29;最终修改稿收到日期:2008-09-11. 本课题得到国家“九七三”重点基础研究发展规划项目基金(2009CB320802)、国家“八六三”高技术研究发展计划项目基金(2008AA01Z301)、国家自然科学基金重点项目(60833007)、澳门大学研究基金以及研究生奖学金资助. 朱 鉴,男,1982年生,博士研究生,主要研究方向为基于图像的绘制、图形硬件加速与基于物理的动画. E-mail: rockyzhu@163.com. 吴恩华,男,1947年生,研究员,博士生导师,主要研究领域为计算机图形学、可视化与虚拟现实.

1 引言

长期以来,实时绘制高度真实感的图像是计算机图形学研究者的一个目标.为了生成高度真实感的图像,往往需要建立复杂的场景模型,且需要比较高级的绘制算法,如光线跟踪、辐射度和光子映射等,从而难以达到实时绘制的效果.近年来,基于图像的绘制(简称 IBR)已成为研究热点之一.与传统的真实感图像生成技术相比,其最大的优点在于成像复杂度几乎与场景复杂度无关,因而成为在复杂场景中实现实时漫游的有利工具.此外,随着现代图形硬件 GPU 的不断发展^[1],一些新的硬件特性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.

本文提出一种基于 GPU 的球面深度图像绘制算法.该算法的绘制流程是一个两步的过程:先实行预变换(pre-warping),再进行纹理映射.首先生成一幅特殊的深度图像(下称球面深度图),该深度图是对整个三维物体的一次均匀全方位采样,它实际是对全光函数(plenoptic function)^[2]的一次离散均匀采样,绘制时只需要读入这样一幅简单的二维图像,而不是整个三维模型,解决了浮雕纹理映射(RTM)^[3]中浮雕纹理对三维物体采样不均匀以及双投影柱面纹理映射(DPCM)^[4-5]中双投影柱面深度图(DPCI)对物体采样不完全的问题.

本文第 2 节介绍相关工作;第 3 节描述球面深度图像的生成;第 4 节描述球面深度图像绘制算法;第 5 节解决视点受限问题;第 6 节描述实时漫游系统的实现;第 7 节给出实验结果及与相关方法的比较;第 8 节总结并提出未来工作.

2 相关工作

RTM^[3]是对三维映射^[6]的一个优化处理,它把三维映射公式变换成简单的预变换和普通的二维纹理映射.与视差映射^[7]相似,二者都通过预变换来模拟三维视差效果.RTM 的预变换和图像插值重构都是通过软件实现的,导致 RTM 方法难以实时绘制.RTM 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,增加了绘制开销.此外,RTM 方法需在相邻面之间进行多次的预变换,否则就会在棱和角处产生空洞,影响了绘制速度.RTM 中视点位置是受限的,当视点进入包围盒正

方体时,产生的中间图像将会出现错误的遮挡关系甚至空洞.为此,Sheng^[8]提出了虚平面映射的方法,但是虚平面的建立增加了绘制的额外开销,并且同样有 RTM 的图像插值、多个面预变换等问题.

硬件加速的 RTM^[9]将预变换工作交给图形硬件支持的像素着色引擎去完成.绘制时,通过使用多重纹理映射直接完成成像.虽然该方法的显示速度很快,但其成像粗糙且不完整.条纹纹理映射^[10]首先为源深度图像中每个像素生成一个条状的纹理(条纹),即根据一个像素及其邻近像素的深度值,沿着深度方向为该像素插值生成多个具有不同深度值的点,它们一起构成该像素的条纹,然后绘制时利用图形硬件的纹理映射直接处理这些条纹.条纹纹理映射采用直线段作为绘制元语,但是图形硬件对点和线的光栅化效率要远低于多边形,而本文采用的是基于三角形的硬件光栅化.

基于硬件的逐像素位移映射^[11],采用图形硬件中的三维纹理来加快后向映射的搜索速度.该方法存储空间耗费大,难以处理复杂物体.Wang 等^[12]把位移映射作为光线跟踪问题来解决,由于需要处理五维函数,所以要求巨大的预计算和存储量.DMesh 等^[13]根据计算出的三维空间对应点坐标来三角化参考图像为一个三维网格,并且基于某种误差准则,生成了一个近似的简化网格.这个粗糙的简化网格必然带来近似误差,难以表示表面几何细节.实时浮雕映射^[14]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.线性搜索需要一个固定的搜索步长,为了表达丰富的表面细节,必须减小步长、增加搜索次数,影响了绘制速度.

DPCM^[4-5]以 DPCI 为绘制元语,在绘制的第 1 步预变换阶段,将源像素预变换到一个与视点相关并且与圆柱体相切的竖直平面上,容易在多处发生插值填补,引起中间图像严重变形,使结果图像显得粗糙.更严重的是,由于柱面深度图只是对物体四周的采样,而顶部与底部信息丢失,当视点位于物体上下方时结果图像会出现空洞.空洞的填补可以通过绘制额外的浮雕纹理来完成,但是增加了绘制开销.此外,该方法有着严重的视点受限问题,本文将在第 5 节展开详细讨论.

3 球面深度图

我们知道三维空间中的点 $P(x, y, z)$ 可以用极坐标 (θ, φ, r) 来表示,其转换公式为

$$\begin{cases} x=r \times \cos(\theta) \times \cos(\varphi) \\ y=r \times \sin(\varphi) \\ z=r \times \sin(\theta) \times \cos(\varphi) \end{cases} \quad (1)$$

其中 $\theta \in [0, 2\pi]$, $\varphi \in [-\pi/2, \pi/2]$. 如图 1 所示, θ 为 P 点在 XZ 平面的投影 P' 到 X 轴的角度, φ 为 P 点到 XZ 平面的仰角. 那么如果我们在一个待绘制物体内部建立极坐标系, 且当作纹理坐标系, 对 θ 和 φ 以固定步长沿着垂直于 (θ, φ) 平面平行于 r 轴的方向进行采样, 会得到一张二维的纹理图像(球面深度图像). 该图像上任何一个像素 P_s 的坐标 (u_s, v_s) , 与其空间对应点 P 的极坐标中的 (θ, φ) 对应相关. 假设该二维纹理图像的分辨率为 $m \times n$, 则其对应的转换关系为

$$(u_s, v_s) = (\theta/2\pi \times m, (\varphi/\pi + 1/2) \times n) \quad (2)$$

该彩色图像的 RGB 通道可以保存物体的颜色信息, α 通道保存对应点极坐标中的 r 值. 也可以用一幅单独的灰度图来保存 r 值. r 值即为三维映射^[6]与 RTM^[3]中所谓的深度值, 其意义为采样点到球心的距离. 这样该二维深度图成为对整个三维物体的一次全方位采样, 所以它实际是对全光函数^[2]的一种离散采样, 也是该原始物体的一种近似表示, 表示的精确度正比于该图像的分辨率.

针对计算机生成的三维模型和现实世界中的物体, 有不同的球面深度图像生成方法.

对于计算机生成的三维模型, 本文采用对模型进行虚拟拍照的方法以得到球面深度图像, 具体方法为: 首先求出模型的包围球, 然后将包围球按所需的分辨率划分为正交的经纬线网格, 每个网格对应球面深度图的一个像素. 从每个像素发出一条连接包围球球心的直线, 求得直线与模型的交点, 计算距该球面最近的交点与球心的距离即该像素的深度(r 值), 同时获取该交点的颜色值以及法向量. 该虚拟拍照的方法类似于鱼眼照相机^[15]的工作原理.

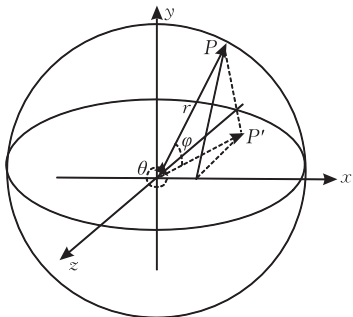


图 1 直角坐标和极坐标的转换关系图

对于现实世界中的物体, 本文采用与 RTM 中一样的方法先获取原始物体的 6 幅正投影深度图, 然后用重投影(re-projection)的方法来获取球面深度图. 其二维解释如图 2 所示, P 为物体上的采样点, 其在右侧浮雕纹理上的对应像素为 P_1 , 对应的球面深度图像上的像素为 P_2 . 我们需要根据像素 P_1 来计算像素 P_2 在球面深度图上的坐标位置. 设 P_1 与 P_2 的坐标分别为 (u_1, v_1) , (u_s, v_s) , 源浮雕纹理和目标球面深度图的分辨率分别为 $m_1 \times n_1$, $m_2 \times n_2$. 易知 $PP_1 = d_1$ 为像素 P_1 对应的深度值, OP 为像素 P_2 对应的深度值 r 值, AP_1 的长度等于像素 P_1 的横坐标, 即 u_1 , $AD = OD = m_1/2$ (所有长度以像素为单位), 从而得到

$$\begin{cases} \tan(\theta) = \frac{PC}{OC} = \frac{P_1D}{OD - CD} = \frac{AD - AP_1}{OD - PP_1} = \frac{m_1/2 - u_1}{m_1/2 - d_1} \\ r = OP = PC / \sin(\theta) = (m_1/2 - u_1) / \sin(\theta) \end{cases} \quad (3)$$

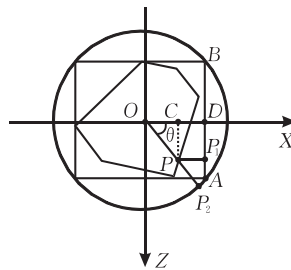


图 2 浮雕纹理重投影的二维解释

联立式(2)和(3), 最终我们可以轻松求得 u_s 和 r . 很容易将此二维情形推广到三维. 三维情形中 u_s 的求取和二维的一样, 因为同一竖直方向上的点其对应的 θ 值相同, 而对于 v_s 只需在 YZ 平面重新计算 P 的仰角 φ , 最后 r 的值则结合 θ 和 φ 即可求得.

本文用第二种方法, 根据已有的 6 幅 Venus 头像浮雕纹理(分辨率为 256×256), 得到了一组分辨率为 512×512 的球面深度图(图 3). 其中图 3(a)保存了颜色信息, 而图 3(b)则保存了其相应的深度信息, 即 r 值(点到球心的距离), 图 3(c)为对应的法向量图.

值得一提的是, 上述计算球面深度图的过程是一个预处理过程, 并不影响实时显示效果. 从后续的讨论可以知道此球面深度图对于后续绘制的预变换阶段是十分有利的, 能够减少计算量, 加快绘制速度. 另外, 对于现实世界的物体, 可以通过 3D 扫描仪获取物体的三维模型, 再用上述虚拟拍照的方法来生成所需球面深度图.

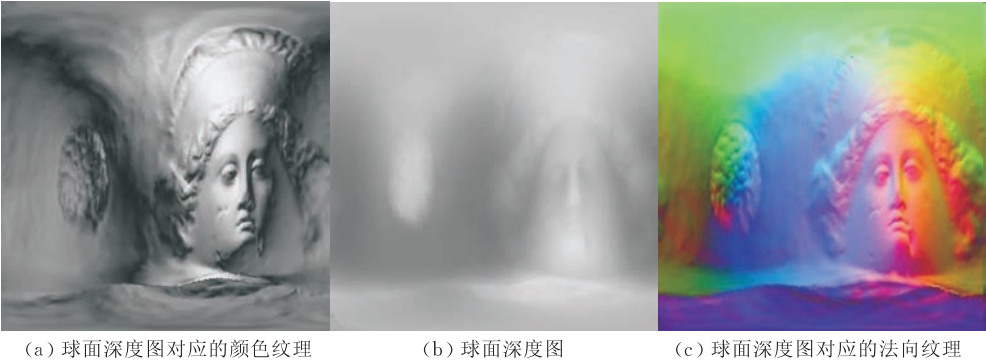


图 3

4 球面深度图的绘制

4.1 算法流程

本小节描述参考球面深度图像在 GPU 上的绘制流程. 算法的输入为球面深度图像和目标视点位置. 对参考球面深度图像进行绘制通过两遍来完成. 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, 该中间图像成像于物体包围球的一个切

平面上, 并且和视点相关, 我们把它实现在 Vertex Shader 中. 由于原图像和中间图像的采样率不同, 会在中间图像像素之间产生空洞, 我们利用图形硬件的光栅化功能来进行图像的插值重采样, 以生成平滑无洞的目标图像. 逐像素光照和环境映射计算在 Pixel Shader 中得以实现, 以生成高品质的光照效果. 第 2 遍绘制把第 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 4 显示了两遍绘制的流程图.

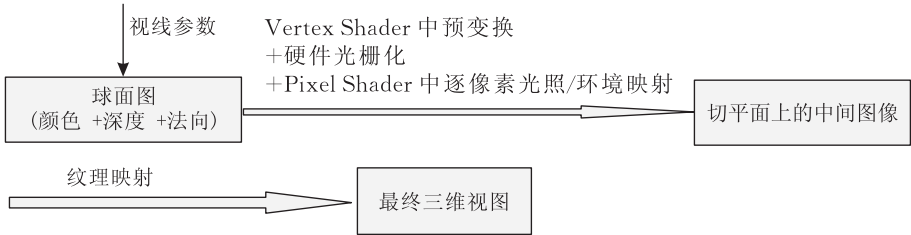


图 4 本文算法的两阶段绘制流程

4.2 映射公式的推导

本文采用的预变换的核心思想和 RTM 是一致的: 对源图像上的每一个像素 (u_s, v_s) , 经预变换后它在中间图像上的对应点 (u_i, v_i) , 再经二维纹理映射变换到目标图像上的点, 与 (u_s, v_s) 经三维变换 (3D-warping) 后得到的目标图像上的点一致. 不同的是, RTM 的中间图像成像于与视点相关的正方体表面上, DPCM^[4-5] 的中间图像成像于圆柱体的与视点相关的切平面上, 而本文则把像素预变换到物体包围球的一个与视点相关的切平面上. 该预变换是一个前向映射的过程, 其三维立体解释见图 6.

物体上的点 P 位于其包围球内, C_i 为目标视点位置, 射线 OC_i 与水平面的夹角为 β , 与球面的交点为 P_9 (极点 epipole^[6]). 我们的目标是把 P 投影到过点 P_9 的切平面上, 即与球面相切于球心和视点的连线与球面交点的切平面. 其二维解释如图 5 所示, P_s 为 P 在其球面深度图对应的像素, C_i 同图 6, P 在目标视平面的成像点为 P' . 预变换的结果是把 P_s

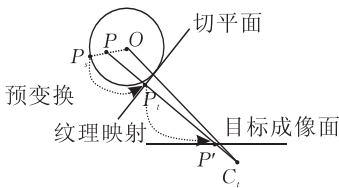


图 5 预变换过程的二维平面解释图

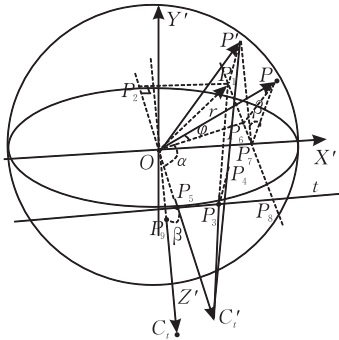


图 6 预变换过程的三维立体解释图

变换到 P_i , P_i 位于球面的切平面, 且为 PC_i 连线与切平面的交点.

回到图 6, 让 OC_i 在竖直方向绕 O 点旋转 β 角, 直到 C_i 到达 C'_i , 使得 OC'_i 位于水平面, 交球面于点 P_5 , 且此时 P 同步旋转到达 P' . 我们以 OC'_i 为 Z' 轴, 位于水平面且垂直 OC'_i 于 O 点的直线为 X' 轴, 竖直方向为 Y' 轴建立坐标系. 不难发现把 P 投影到过点 P_9 的切平面上, 等价于把 P' 投影到过点 P_5 的切平面上, 两次投影得到的对应点在相应的切平面上的局部坐标是相等的. 而後者的计算却要简单得多. 如图 6, 设 P' 与 C'_i 的连线交过点 P_5 的切平面于 P_4 , P_4 即为 P' 在过 P_5 的切平面上的投影位置.

设 P_1 为 P' 在 $X'Z'$ 平面的垂直投影, P_1 与 C'_i 的连线交过 P_5 的切平面于 P_3 , P_1 在 Z' 轴的投影为 P_2 . 设直线 t 为球面的一条切线, 且过 P_5 , 那么 P_3 必在直线 t 上. 在过 P_5 的切平面内如果以 P_5 为原点, t 为 X 轴, 过 P_5 且垂直于 t 的直线为 Y 轴建立局部坐标系, 则 Y 轴平行于 Y' 轴和 P_4P_3 , 那么 P_4 在此坐标系下的坐标为 (P_3P_5, P_4P_3) , 设为 (u_i, v_i) . 从而对于点 P' 的预变换实际上转化为求取 P_3P_5 和 P_4P_3 .

设 P_6 为 P 在 $X'Z'$ 平面的垂直投影, P 点极坐标为 (θ, φ, r) , 则 $\angle POP_6 = \varphi$, $PO = r$. P_6 到 Z' 轴的偏转角 $\angle P_6OP_5$ 设为 α , 即 P 与极点 P_9 的极坐标中 θ 角之差. 直线 P_1P_6 交 X' 轴、直线 t 分别于 P_7, P_8 . 由于 P' 实际上为 P 绕 X' 轴旋转 β 角而得到, 故 $\angle PP_7P' = \beta$, 且直线 P_1P_6 平行于 Z' 轴. 由图 6 可知

$$disparity = \frac{C'_iP_4}{C'_iP'} = \frac{C'_iP_3}{C'_iP_1} = \frac{C'_iP_5}{C'_iP_2} = \frac{d-R}{d-OP_2} \quad (4)$$

其中, d 为视点与球心的距离, 即 OC'_i 的长度, R 为球的半径. 注意 OP_2 的长度有正负之分, 指向正 Z' 轴时为正, 否则为负. 而 $disparity$ 则类似于三维映射^[6]中的同名称的变量, 它能够反映场景中的点距离视点的深度信息. OP_2 可由下面的推导求得

$$\begin{aligned} OP_2 &= P_7P_1 = P_7P' \times \cos(\angle P_8P_7P') \\ &= P_7P' \times \cos(\angle P_8P_7P + \beta) \\ &= P_7P \times \cos(\angle P_8P_7P + \beta) \\ &= P_7P \times (\cos(\angle P_8P_7P) \times \cos(\beta) - \\ &\quad \sin(\angle P_8P_7P) \times \sin(\beta)) \\ &= P_7P_6 \times \cos(\beta) - PP_6 \times \sin(\beta) \\ &= OP_6 \times \cos(\alpha) \times \cos(\beta) - r \times \sin(\varphi) \times \sin(\beta) \\ &= r \times (\cos(\varphi) \times \cos(\alpha) \times \cos(\beta) - \sin(\varphi) \times \sin(\beta)) \end{aligned} \quad (5)$$

根据相似三角形的性质, 我们有

$$\begin{aligned} P_3P_5 &= disparity \times P_1P_2 = disparity \times P_7O \\ &= disparity \times OP_6 \times \sin(\alpha) \\ &= disparity \times r \times \cos(\varphi) \times \sin(\alpha) \end{aligned} \quad (6)$$

综合式(4)~(6), 最终可求得

$$\begin{aligned} P_3P_5 &= u_i = ((d-R) \times r \times \cos(\varphi) \times \sin(\alpha)) / \\ &\quad (d - r \times (\cos(\varphi) \times \cos(\alpha) \times \\ &\quad \cos(\beta) - \sin(\varphi) \times \sin(\beta))) \end{aligned} \quad (7)$$

同理可得

$$\begin{aligned} P_4P_3 &= v_i = ((d-R) \times r \times (\cos(\varphi) \times \cos(\alpha) \times \\ &\quad \sin(\beta) + \sin(\varphi) \times \cos(\beta))) / \\ &\quad (d - r \times (\cos(\varphi) \times \cos(\alpha) \times \\ &\quad \cos(\beta) - \sin(\varphi) \times \sin(\beta))) \end{aligned} \quad (8)$$

从而我们得到了在球面切平面上的映射公式. 实际绘制时我们可以做出适当优化, 并且该公式在以球面深度图作为输入时可以大大简化计算, 我们将在 4.3 节展开详细讨论.

4.3 在 Vertex Shader 中的预变换

为了有效地在 GPU 上进行预变换, 首先把参考球面深度图像进行三角化. 把参考球面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. 绘制时先根据当前视点位置来求得 4.2 节讨论的 β, d 以及极点(图 6 中为 P_9) 在球面深度图的坐标 (u_e, v_e) . 对像素 P , 设其坐标为 (u_s, v_s) , 对应深度值为 r , 参考图像的分辨率为 $m \times n$, 则对于式(7)和(8), 我们有

$$\begin{cases} \alpha = (u_s - u_e) \times 2\pi/m \\ \varphi = v_s \times \pi/n \end{cases} \quad (9)$$

由于观察到式(9)中 u_s, v_s, u_e 均是离散的整数值, 取值范围取决于图像的分辨率, 我们可以事先保存好一份表, 表长正比于图像的分辨率大小. 每次读入一个新的像素, 可以直接由像素坐标经简单计算来查表索引求得对应的 α 以及 φ 值, 从而避免了繁琐的世界直角坐标到极坐标的转换, 大大简化了计算.

对于变换式(7)和(8), β 和 d 仅随视点变化而改变, 不依赖于特定的顶点, R 在整个绘制过程固定不变, 从而每次绘制可以把 β, d 和 R 当成固定常量传递给 Vertex Shader. 式(7), (8)中其他的 3 个变量 α, φ 和 r , 可以当成三角形的顶点传递给 Vertex Shader, 而其颜色和法向量可以由其对应的纹理坐标 $(u_s/m, v_s/n)$ 由纹理硬件来获取.

此外, 我们观察到式(7)和(8)有共同的公因子 p :

$$p = ((d-R) \times r) / (d - r \times (\cos(\varphi) \times \cos(\alpha) \times$$

平面 1 上不会有点 P_i 的成像,其原因同样是因为 P'_i 位于视平面 1 的后方,从而,DPCM 方法中,视点不仅不可以进入圆柱体,而且也不可以进入圆柱体的正上方或者正下方,这样的视域受限是比较严重的。

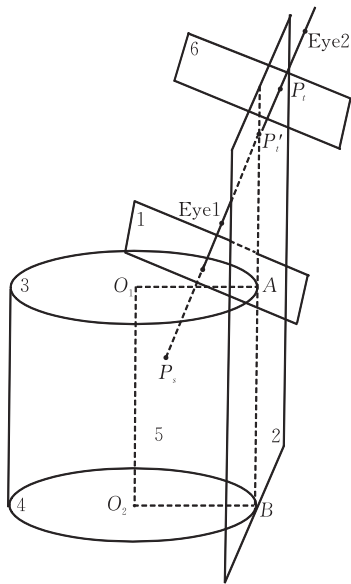


图 8 DPCM 算法的视点受限三维解释

为了解决此问题,Sheng^[8]针对 RTM 提出了虚平面变换.根据视点的变化,该方法动态地建立一个位于视平面前方的虚平面,然后把浮雕纹理上像素点预变换到此平面上,而不是 RTM 中的浮雕纹理平面自身,避免了上述问题的出现.而对于本文提出的算法,我们可以更加方便地建立所谓的“虚平面”.如图 7,视点 C_i 位于球面内,射线 OC_i 的延长线交球面于 O_1 (极点),其反向延长线交球面于 O_2 ,我们称之为次极点.视点进入包围球时,我们只需把参考图像上的像素预变换到过次极点(O_2)而非极点(O_1)的切平面即可,图中即为平面 t_1 .比如对于点 P_1 ,取而代之,我们将其投影到平面 t_1 上,而非 t_2 ,生成中间图像上的像素 P_{t2} ,此时由于 t_1 位于视平面 t_3 的前方而避免了上述问题的发生.我们总是可以找到这样一个切平面作为中间成像面,且其预变换公式和 4.2 节中一样,只需修改部分参数值即可.第 7 节图 10(d)、(e)即为视点处于球面内且十分靠近物体,采用上述方法绘制的结果,而 DPCM 方法无法在该视点下生成完整图像。

6 基于球面深度图的实时漫游系统

本文提出的算法可以应用于多个基于球面深度

图表示的物体的实时漫游系统。

为了同时绘制多个物体,除了要产生正确的自遮挡图像外还必须保持各物体间正确的遮挡关系.RTM^[3]采用了 McMillan 的极线原则排序法^[6]组织像素进行绘制,该方法本质上是一个画家算法^[16](painter's algorithm),保证了所有的点按照由远到近的顺序来绘制.但是它不能用于本文的球面深度图,这是因为 McMillan 的方法是基于平面成像面推导得出来的,而本文的参考图是依据球面极坐标系采样而得到的.此外,对于每个参考图像上的像素传递给 Vertex Shader 的是 (α, φ, r) ,并非其空间对应点的实际三维直角世界坐标,这样图形硬件(Z-buffer)对深度值的比较结果必然是错误的,然而利用硬件的可编程性可以在 Vertex Shader 中手动设定深度值.如图 6,对于 P 旋转后得到的点 P' ,我们可以利用其在 Z' 轴上的投影点 P_2 ,根据 $C_i P_2$ 的长度,以及 $zNear$ ^[16]和 $zFar$ ^[16]的值来手动计算出一个 $[0, 1]$ 之间的深度值再传递给 Z-buffer,这样就能够确保正确的深度检测,使得中间图像保持正确的遮挡关系.由式(4)知道, $C_i P_2$ 的长度即为公因子 P 的分母,只需在计算式(10)时保存即可, $zNear$ 和 $zFar$ 的值固定,整个绘制流程不变。

为了实时地绘制多个基于球面深度图表示的物体,考虑到绘制时无需输入复杂的几何模型,而只是简单的二维图像,我们可以方便地使用动态的 LOD (Level of Detail) 技术.对于离视点较远的物体,我们使用较低的分辨率进行绘制,从而可以在绘制的第一阶段大大减少待变换的像素个数,减少了计算量,同时又不影响最终的图像质量.首先,对每一个待绘制的物体,生成不同分辨率的球面深度图,其原理类似于目前广泛应用的 mipmap 纹理映射技术^[17].绘制时,首先计算视点和物体包围球切平面之间的距离,图 6 中即为 $C_i P_9$ (或 $C'_i P_5$),其值为 $d-R$,该值的计算不会增加任何额外的开销,再计算出该切平面与视线方向的夹角 t ,而值 $(d-R) \times \cos(t)$ 反比于第二阶段绘制时切平面在成像面上的投影面积 S ,从而我们可以根据 $(d-R) \times \cos(t)$ 的值来决定最终使用哪一级别的分辨率.该值越大,物体在视平面投影面积越小,使用的参考图像分辨率就越低。

我们实现了一个基于球面深度图像的实时漫游系统,绘制效果如图 13 所示,除背景外所有的 Venus 头像和佛像都是用本文算法并结合上述深度遮挡测试和 LOD 技术来绘制.当输出图像分辨率为 1024×512 时,其实时绘制速度可以达到 43.231fps.

7 实验结果比较

我们在 Windows XP 系统实现了本文的算法. 系统的硬件配置为 Pentium(R)4 3.20GHz 和 512MB 主内存, 显卡采用 nVidia GeForce 6600. 首先用 C 实现了第 3 节讨论的球面深度图生成算法, 程序的输入为 6 幅浮雕纹理(RT)及其对应的法向量图, 生

成的结果图像见图 3. 接着用 OpenGL 和 Cg 实现了本文的绘制算法, 绘制程序以球面深度图作为输入. 绘制结果见图 10~图 12. 同时, 我们在同一平台与基于多边形、RTM 方法以及 DPCM^[4-5] 方法进行了比较, 其中本文和 DPCM 算法输入图像分辨率一致, 而 RTM 方法的输入为生成本文球面深度图所用浮雕纹理, 输出图像分辨率均为 512×512, 因而比较是合理的. 绘制速度见表 1.

表 1 同一平台下各个绘制算法的比较

绘制方法	输入		绘制速度/fps
多边形方法	多边形数量	67170	35.625
本文的方法	球面深度图	分辨率 512×512	74.214
本文方法+Per-pixel Lighting/ Environment Mapping	球面深度图+Normal 贴图	分辨率 512×512	37.432 ^① /40.136 ^②
RTM	6 幅浮雕纹理(RT)	分辨率 256×256	45.717
DPCM 绘制算法	DPCI+上/下底面浮雕纹理	分辨率 512×512(DPCI)	67.249 ^③ /56.783 ^④

注: ①表示 per-pixel lighting; ②表示 Environment mapping; ③表示单幅 DPCI; ④表示 DPCI+上/下底面浮雕纹理.

由表 1 中数据可知, 本算法的绘制速度是传统的基于多边形的方法的 2 倍, RTM 方法的 1.6 倍, 即使是加入逐像素的光照计算^①时的速度都可以与传统的方法相当. RTM 中繁琐的插值和多个面的变换占据了绘制时间, 而本文的方法采用了 GPU 加速. 比单幅 DPCI 绘制时快 15% 左右^③, 而单幅 DPCI 绘制时的结果会出现空洞^[4-5] (图 9), 原因在于 DPCI 不是对三维物体的全方位采样. 为了填补空洞, 该算法需要将上下底面浮雕纹理再绘制一次, 此情况下本文算法的绘制速度是其 1.3 倍^④. 本文的算法绘制的结果图像是完整的, 并无空洞, 成像质量高(图 10(d)、(e)), 绘制速度也快. 硬件加速的 RTM^[9]虽然绘制速度比 RTM 快, 但空洞填补只是

简单的赋值, 成像十分粗糙且不完整. 从文献[4]我们知道逐像素位移映射^[11]和浮雕映射^[14]等后向映射方法比本文和 DPCM 的前向映射方法要慢很多. DMesh^[13]中大量顶点的几何变换需要通过矩阵乘法来完成, 成为绘制瓶颈, 且采用了粗糙的简化网格, 绘制结果并不理想.

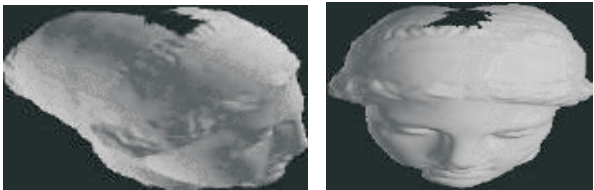


图 9 视点位于物体上方时 DPCM 方法^[4-5]绘制的结果

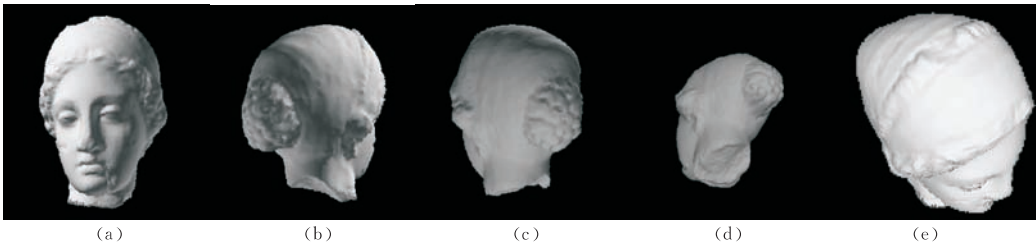


图 10 不同视点位置和视线方向下本文算法的绘制结果



(a) 颜色纹理 (b) 绘制结果1 (c) 绘制结果2

图 11 本文算法结合 Pixel Shader 中逐像素光照的绘制结果



图 12 本文算法结合 Pixel Shader 中环境映射的绘制结果

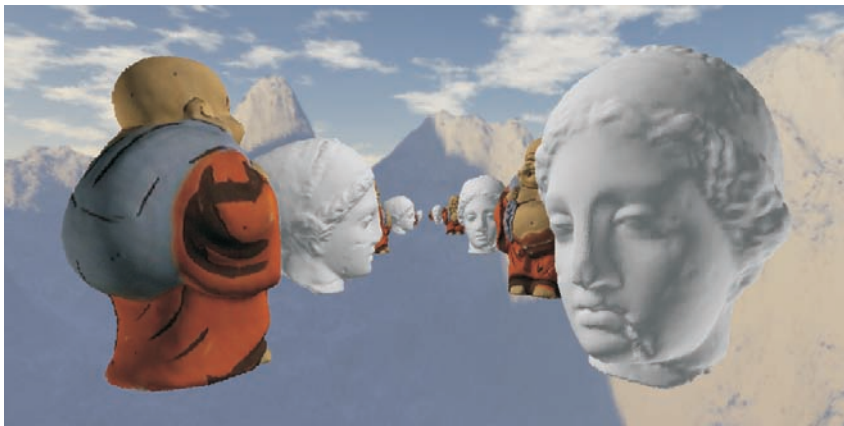


图 13 基于球面深度图的实时漫游系统(43. 231fps)

8 总结与未来工作

本文提出了一种 GPU 加速的基于图像的绘制算法: 首先利用极坐标系生成对物体全方位均匀采样的球面深度图, 解决了 RTM^[6] 方法采样不均匀以及 DPCM^[4-5] 方法采样不完全(即绘制结果有空洞)的问题. 绘制时将参考球面深度图像预变换到物体包围球切平面上, 再利用纹理映射生成最终图像. 我们利用现代图形硬件的可编程性和并行性来加速预变换过程, 硬件的光栅化功能完成图像的插值, 解决了 RTM 软件实现的预变换和图像插值重构的绘制瓶颈问题. 此外, 球面深度图本身可以减少变换次数, 简化变换计算, 从而大大提高了绘制速度. 最终, 我们的算法解决了 RTM 和 DPCM 等方法的视点受限问题, 实现了一个实时漫游系统. 实验结果表明, 本文的方法以单幅图像作为输入可以生成高质量绘制结果, 并且成像速度也很快.

同 RTM 和 DPCM 一样, 本文的算法只适用于单层物体, 对于复杂的物体绘制结果可能会出现错误的自遮挡甚至空洞. 一个未来工作是将复杂物体分割为多个适合于球面深度图表示的简单模型, 然后用第 6 节的方法来实时绘制; 另一未来工作是拓展球面深度图的数据结构, 类似于层次深度图(LDI)^[18], 单个像素位置保存多个像素值, 形成对复

杂物体的完整表示, 绘制时根据像素深度值自动决定各个像素的可见度.

参 考 文 献

- [1] Macedonia M. The GPU enters computing's mainstream. IEEE Computer Society, 2003, 36(10): 106-108
- [2] Adelson E H, Bergen J R. The plenoptic function and the elements of early vision//Computational Models of Visual Processing, Chapter 1. Cambridge, Mass: The MIT Press, 1991
- [3] Oliveira Manuel M, Bishop Gary, McAllister David. Relief texture mapping//Proceedings of the SIGGRAPH 2000. New Orleans, Louisiana, USA, 2000: 359-368
- [4] Li Bao-Quan, Liu Xue-Hui, Wu En-Hua. Real-time rendering depth images on GPU by forward warping. Journal of Software, 2007, 18(6): 1531-1542(in Chinese with English Abstract)
(刘保权, 刘学慧, 吴恩华. 基于 GPU 的实时深度图像前向映射绘制算法. 软件学报, 2007, 18(6): 1531-1542)
- [5] Li Xiao-Ying, Liu Bao-Quan, Wu En-Hua. Full solid angle panoramic viewing by depth image warping on field programmable gate array. International Journal of Virtual Reality, 2007, 6(2): 69-77
- [6] McMillan L, Bishop G. Plenoptic modeling: An image-based rendering system//Proceedings of the SIGGRAPH'95. Los Angeles, CA, USA, 1995: 39-46
- [7] Tomomichi K, Takahei T, Inami M, Kawakami N, Yanagida Y, Maeda T, Tachi S. Detailed shape representation with parallax mapping//Proceedings of the 11th International

- Conference on Artificial Reality and Telexistence (ICAT 2001). Tokyo: ACM Press, 2001: 205-208
- [8] Sheng Bin, Wu En-Hua. Walking into images: Virtual plane mosaics for plenoptic modeling//Proceedings of the Virtual Reality Conference (VR'07). Charlotte, North Carolina, USA, 2007: 187-194
- [9] Fujita M, Kanai T. Hardware-assisted relief texture mapping//Navazo I ed. Proceedings of the Eurographics 2002. Saarbrücken: Eurographics Association. Saarbrücken, Germany, 2002: 257-262
- [10] Li Kui-Yu, Wang Wen-Cheng, Wu En-Hua. Bar texture mapping. Journal of Software, 2004, 15: 179-189 (in Chinese with English Abstract)
(李奎宇, 王文成, 吴恩华. 条纹纹理映射. 软件学报, 2004, 15: 179-189)
- [11] Donnelly W. GPU Gems. 2nd Edition. New York: Addison-Wesley, 2005: 123-136
- [12] Wang L F, Wang X, Tong X, Lin S, Hu S M, Guo B N, Shum H Y. View-dependent displacement mapping. ACM Transactions on Graphics (Proceedings of the SIGGRAPH 2003), 2003, 22(3): 334-339
- [13] Pajarola R, Sainz M, Meng Y. DMesh: Fast depth-image meshing and warping. International Journal of Image and Graphics (IJIG), 2004, 4(4): 1-29
- [14] Policarpo F, Oliveira M M, Comba J L D. Real-time relief mapping on arbitrary polygonal surfaces. ACM Transactions on Graphics (Proceedings of the SIGGRAPH I3D 2005), 2005, 24(3): 155-162
- [15] Xiong Y, Turkowski K. Creating image-based VR using a self-calibrating fisheye lens//Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97). San Juan, Puerto Rico, 1997: 237-243
- [16] Hearn Donald, Baker M Pauline. Computer Graphics with OpenGL. 3rd Edition. San Francisco, Prentice Hall, Pearson Education, 2004
- [17] Williams L. Pyramidal parametrics. SIGGRAPH83 Proceedings, Detroit, Michigan, 1983, 17(3): 1-11
- [18] Shade Johnatan et al. Layered depth images//Proceedings of the SIGGRAPH 1998. Orlando, Florida, USA, 1998: 231-242



ZHU Jian, born in 1982, Ph.D. candidate. His research interests include image-based rendering, graphics hardware acceleration and image processing.

WU En-Hua, born in 1947, Ph.D., professor. His research interests include realistic image synthesis, scientific visualization and virtual reality.

Background

Real time rendering of highly realistic scene has been a goal pursued by computer graphics researchers for decades. The parallelism of programmable hardware provides us a way to speed up, and image-based rendering techniques provide us a way to produce realistic images.

Relief texture mapping (RTM) can efficiently represent the 3D surface details of objects, but its multi-pass pre-warp and interpolation for the hole-filling by software dominates the rendering time, which is computationally expensive and time-consuming. Moreover, In RTM an object representation consists of six relief textures associated with the cubic surfaces. This kind of representation has to be greatly over-sampled at the edges and the corners of the bounding box, which increases much unnecessary overheads.

Fujita et al. use hardware-assisted per-pixel shader functions to accelerate relief texture mapping by offloading most of the original computations from CPU to GPU. However, it simply fills in the holes, caused by different sampling rate between source and target images, with nearby pixels; the quality of rendered images is usually poor.

Donnelly implemented per-pixel displacement mapping in graphics hardware using 3D texture to speed up searching for the backward mapping method, but 3D texture requires such large volume of storage that it is difficult to represent large-scale scenes. Bar texture mapping uses lines as rendering primitives, but hardware rasterization is more efficient to

polygons, in our method we use triangles for rasterization.

Double Projective Cylindrical texture Mapping (DPCM) uses Double Projective Cylindrical Image (DPCI) for object representation, and puts the whole pre-warping process into Vertex Shader on GPU. As a result, its rendering speed is faster. However, DPCI is not a complete representation of the rendered object, there may be holes at the resultant images due to lack of source information. Even worse, DPCM has heavily limited field of view.

To solve these problems, the authors proposed a GPU accelerated real-time depth image based rendering algorithm. The main contribution of this paper is as follows. The proposed Spherical Depth Image is sampled completely and uniformly, so the resultant image is complete without holes and the rendering overheads are minimized. By transporting most of the computation into GPU, we can greatly increase the rendering speed. Finally, we solve the problem of limited field of view. Experiment results show that it is suitable for massive real-time walkthrough system.

This work is supported by National Key Fundamental Research and Development Project (973) with Grant No. 2009CB320802, National High-tech R&D Program (863 Program) with Grant No. 2008AA01Z301, National Natural Science Key Foundation of China with Grant No. 60833007, the research grant and postgraduate studentship of University of Macau.