文章编号:1671-4598(2014)05-1516-03　　中图分类号:TP302　　文献标识码:A

基于 DCT 并行加速算法图像渲染平台系统设计

于艳东
(集宁师范学院 计算机系,内蒙古 乌兰察布 012000)

摘要:此次主要研究了基于 GPU 的集群渲染系统平台设计;为了提高平台的工作效率、增强集群渲染系统平台的数据传输能力,提出了一种采用 DCT 变换的方法来加速图像渲染速度;该方法利用 DCT 变换算法加速图像的实时压缩,加入 CPU 监控器和任务分配器模块,让 GPU 和 CPU 共同承担了绘图和渲染的目的,这样有效地降低处理流程对 CPU 的占用,实现了三维绘图和特效渲染的加速;为了验证平台的有效性以及图像压缩处理的效果,做了相应的功能验证;对 640×480 的 RCB 图像使用上述压缩方法和 JPEG 标准库在不同压缩设置下进行实验;仿真实验结果表明所提方案具有更高的压缩效率。

关键词:集群渲染;DCT 变换;图像压缩;JPEG

Accelerating Application of Rendering Platform Based on DCT Parallel Algorithm

Yu Yandong
(Computer Science Department, JiNing Normal College, Wulanchabu 012000, China)

Abstract: This paper mainly researches the design of cluster rendering system platform based on GPU. In order to improve work efficiency, enhance the platform data transmission capability of cluster rendering system platform, this paper presents a DCT transform to accelerate the rendering speed. The method using DCT transform algorithm to accelerate the image real-time compression, joined the CPU monitor and the task distributor module, GPU and CPU shared the drawing and the purpose of rendering, so reduce the occupancy of the CPU treatment process, the accelerated 3D graphics and special effects rendering. Finally, with the JPEG Library in different compression settings were compared, the simulation results show that the proposed scheme has higher compression efficiency.

Key words: cluster rendering; DCT transform; image compression; JPEG

0 引言

随着计算机技术的不断发展进步,在三维仿真中图形渲染技术已经得到越来越多的应用。尽管对于图形的软/硬件处理方式都有很大提高,但是,一般性能的渲染平台已无法满足高度复杂三维场景对于真实感、实时性的要求[1]。衡量图形渲染平台性能的一个重要指标是对图形图像的高质量处理能力,海量数据图像降低了系统平台的实时出具处理性能,因此,高效率的图像压缩编码方法是目前的研究热点[2-3]。

已有一些研究学者提出了提高图像压缩编码效率的算法[4],这些算法都是通过 CPU 计算,CPU 的高负荷运行使得无论是真实感还是实时性方面,均无法满足高质量处理的要求。为了解决这个问题,NVIDIA 公司提出了采用图形处理单元(GPU)芯片的方法,使之在计算机整体架构中专门用于处理图形图像[5]。并且,Cabral 等人提出了一种基于三维纹理映射硬件的绘制方法,用以解决复杂图形绘制方面的效率问题[6]。目前,随着 GPU 的高速发展和普及应用,基于 GPU 的图形图像处理方法得到越来越广泛的关注。

基于 GPU 的实时图像处理方法,本文提出了采用 OpenGL 的扩展功能对图形接口进行构建,增强集群渲染系统的数据传输能力。该方案可对普通的图像压缩算法进行并行加速,降低处理流程对 CPU 的占用,提高平台的效率,实现基于 GPU 的三维绘图和特效渲染。

收稿日期:2014-01-08; **修回日期**:2014-03-08。

作者简介:于艳东(1979-),女,内蒙古乌兰察布市人,工程硕士,讲师,主要从事计算机科学教育、计算机软件技术等方向的研究。

1 系统体系结构

1.1 图形系统的硬件

本文所设计的渲染平台硬件组成如图 1 所示。其中 GPU 为支持 OpenGL 的图形处理器,负责渲染列表计算处理任务;CPU 为 32 位,可采用 ARM9 等嵌入式处理器,用以执行图形应用程序;DMA 可将渲染列表发送至 GPU;VGA/LCD 控制器主来生成显示器的扫描时序信号。系统平台配备两块帧缓存,一块用于显示,一块用于保存 GPU 的绘制结果,通过两者之间的乒乓操作可以确保整个平台的高效运行。

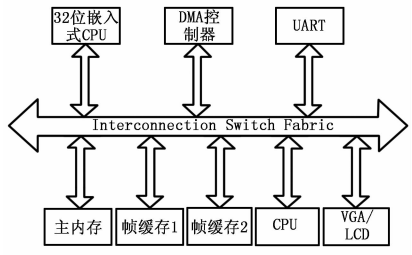

图 1 系统平台的硬件组成

GPU 的模块如图 2 所示。其中,存储器接口用于外接显存;CPU 接口用于接收渲染列表。不同的 CPU 总线规格可以开发相应的 CPU 接口,GPU 可添加到系统平台中,对图形图像的运行程序进行加速,增强了 GPU 的通用性。

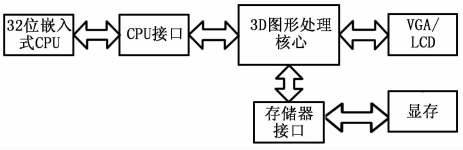

图 2　GPU 连接图

1.2　图形管线

组成 GPU 的核心是图形管线，为了降低平台的设计难度，图形管线中没有纹理贴图这一级。具体结构如图 3 所示。

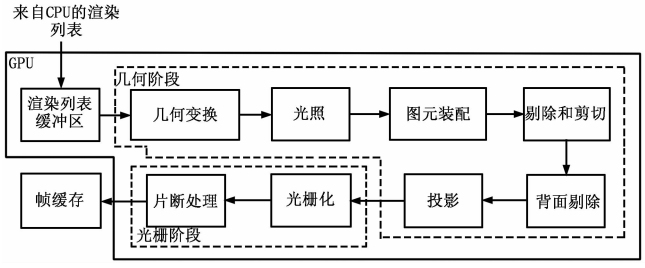

图 3　形管线的结构

从图 3 看出，来自 CPU 的渲染列表保存于 GPU 的渲染列表缓冲区内。图形管线的第一级是对组成物体的顶点进行变换，然后确定视觉颜色；图形管线的第二级根据三维场景中光源的设置对顶点的颜色进行重新计算；图形管线的第三级对独立的顶点进行组装，形成新的图元，进而降低硬件实现的复杂度；图形管线的第四级将视景外的图元进行剔除和剪切，只保留视景体内部的部分；图形管线的第五级对背面朝向观察者的三角形图元再次进行剔除，减轻管线的计算负担；图形管线的第六级将三维空间中的图元投影到二维平面中，对平面到视口的坐标进行转换；图形管线的第七级是光栅化，用离散的点阵来表示图元；图形管线的第八级对光栅化后的片段数据进行深度测试，最终通过深度测试的片段数据会被显示出来。这八级即为 GPU 体系的关键组成部分。

2　实时图像压缩和图形处理

图形渲染平台的性能是通过处理图像的效率来体现的。为了提高平台的运行效率，需要对须向进行压缩编码。

2.1　DCT 计算

基于 GPU 的实时图像压缩方法是通过检测目前通用的 JPEG 算法，使用并行计算架构（CUDA）技术在 GPU 上进行并行计算。CUDA 的作用是可以直接把 OpenGL 渲染输出的图像作为 CUDAkernel 的输入。对于不能进行并行计算的步骤，仍使用 CPU 进行计算。对于海量数据的原始图像（如 RDG 格式图像），首先将其转换为 YCrCb 色彩空间，则有

$$Y = 0.299R + 0.587G + 0.114B - 128 \tag{1}$$

$$Cb = -0.16874R - 0.33126G + 0.5B \tag{2}$$

$$Cr = 0.5R + 0.41869G - 0.08131B \tag{3}$$

由上面 3 个式子可以看出，每个像素的转换均为独立的过程，因此，可以通过 CUDA 进行并行计算。颜色转换完成后，需要继续进行 DCT 计算。当图像块为 $N \times N$ 时，二维 DCT 的计算公式为

$$T(u,v) = \sum_{x=0}^{N}\sum_{y=0}^{N} f(x,y)g(x,y,u,v) \tag{4}$$

式（4）中的 $g(x, y, u, v)$ 计算公式为

$$g(x,y,u,v) = \left\{\alpha(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\right\} \left\{\alpha(v)\cos\left[\frac{(2y+1)v\pi}{2N}\right]\right\} \tag{5}$$

且有

$$\alpha(u),\alpha(v) = \begin{cases} \sqrt{\dfrac{1}{N}} & u,v = 0 \\ \sqrt{\dfrac{2}{N}} & u,v = 1,2,\cdots,N \end{cases} \tag{6}$$

（4）、（5）和（6）3 个式子虽然可以对像素进行二维 DCT 变换，但是复杂度较高，限制了可并行性。为此，将二维 DCT 拆分为一维，每一行的一维 DCT 用一个 thread 来表示，则 YCrCb 分量可表示为

$$S_i = \frac{\alpha(i)}{2}\sum_{x=0}^{7}\cos\left[\frac{(2x+1)i\pi}{2N}\right] \tag{7}$$

上式中的（i）定义与（6）式相同。

2.2　编码设计

本文采用了霍夫曼编码方式，可有效进行无损压缩，并可去除大量的冗余信息。这里对实际编码进行并行，可分为两个阶段：第一个阶段是给没个像素一个 thread，并查找码值；第二个阶段是个循环过程，将第一阶段中的各个码值组合在一起。

2.3　光栅化处理阶段

由前面可知，图形管线的第七级是光栅化，用离散的点阵来表示图元。在投影显示阶段，除了需要对图像进行压缩外，还需要对图形进行处理。第七级中进入的是顶点数据，发出的是片段数据，则对光栅化引擎的端口程序设计为：

```
class SOM_rasterize_engine
: public sc_module                                    ①
{
public:
sc_port<read_if> projection_pipe_read;                ②
sc_port<write_if> fragment_pipe_write;                ③
……
};
```

②中的 projection _ pipe _ read 为投影和光栅化两个引擎之间的 Fifo 通道指针，后者通过该指针来读取前者的处理结果。③中的 fragment _ pipe _ write 为光栅化和片段处理两个引擎之间的 Fifo 通道指针，该端口被光栅化引擎用来将处理结果写入后续管线。对于片段数据流，其格式如表 1 所示。

表 1　片段数据渲染列表

Screen space X(Integer)
Screen space Y(Integer)
Screen space Z(Integer)
Color R
Color G
Color B
Color A

2.3　图像压缩

在满足数据精度要求的前提下，采用阈值化的方式，有效地利用压缩小波系数重构数据，需要为压缩的小波系数建立索

引。本节提出了为小波系数建立快照索引的方法，该方法用索引变量的 $N-1$ 个比特位的状态标识 $N-1$ 个细节分量值是否为 0。如果小波系数中第 i 个该细节分量值为 0，则索引变量的第 i 个比特位置 1。

压缩的小波系数可以表示为如下的三元组＜*average*，*index*，*coefficients* [] ＞：

(1) *average* 为小波系数中所有数据的整体均值。

(2) *Index* 是由 $N-1$ 个比特位组成，第 i 个比特位表示第 i 个细节分量值是否为 0。

(3) *Coefficients* [] 为记录未压缩的细节分量值。

建立快照索引的过程及算法如下：

将 *index* 的初值置 0，压缩小波系数时，依次扫描近似的小波系数中的第 i 个细节分量值，如果值为 0，*index* 的第 i 位保持不变，否则将细节分量值写入 *coefficients*，同时将 *index* 的第 i 位置 1。在重构数据时，只要根据 *index* 的第 i ($i=1$，…，$N-1$) 个比特位的状态 (0 或 1)，即可判定第 i 个小波系数是否为 0，不为 0 的分量值可从 coefficients 中获得。

以上节的 W' 为例，当对 W' 取阈值 2 时，即可使大部分细节分量为 0 即为 { [340，341]， [0]， [0，0]， [0，2.121 3，0，－2.828 4]}。故对细节分量取阈值后得到的压缩的小波系数可以表示如下：

$\hat{W}$ = {341,110 000 101,[340,341,2.121 3,－2.828 4]}

故可以看到最后只需要保存 4 个小波数据，仅为原始数据的 4/9＝44.4％。

算法详细描述了索引的建立过程如下。

算法 *Index _ coefficients* () 子函数

输入：近似的小波系数 W'{W'[0],W'[1],...,W'[N] }

输出：压缩的小波系数 $\hat{W}$

函数功能：将近似的小波系数 W' 压缩为压缩的小波系数 $\hat{W}$

为压缩的小波系数建立索引算法描述：

(1)$\hat{W}$. average＝W′[0]
(2)$\hat{W}$. index＝0
(3)for i in 1 to N－1
(4)if W′[i]！＝0
(5)将$\hat{W}$. index 的第 i 位置 1
(6)将$\hat{W}$ [i]加入$\hat{W}$. coefficients[]
(7)end if
(8)end for

容易看出，算法的时间复杂性为 O (N)。

3 仿真实验结果

为了验证平台的有效性以及图像压缩处理的效果，做了相应的功能验证。对 640×480 的 RCB 图像使用上述压缩方法和 JPEG 标准库在不同压缩设置下进行实验。图 4 (a) 是大小为 900 k 的原始图像，图 4 (b) 是压缩质量系数为 100 时候的效果图。

采用 JPEG 标准库方法对图 4 (a) 中的图像做压缩质量系数为 100 的压缩实验，图 5 为采用两种方法的对比图。实验结果表明，在相同压缩比下，本文所述方法具有更高的压缩效率。

(a) 900k的原始图像　(b) 压缩质量系数为100时候的效果图

图 4　压缩效果图

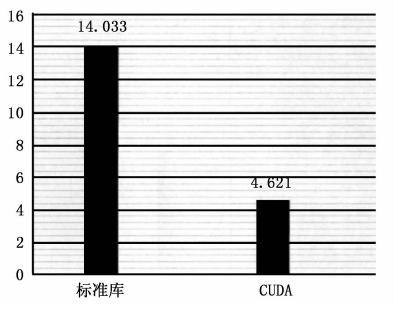

图 5　压缩效率对比

4 结论

本文基于 GPU 的实时图像处理方法，构建了集群渲染系统平台，介绍了系统架构，详细说明了硬件结构的功能和执行流程。在此基础上，提出了集群渲染系统的同步运行方法，可以实时的对图像进行压缩处理。最后，通过实验检测结果可以看出，本文所述方法与传统的 JPEG 标准库相比具有更高的压缩效率。

参考文献：

[1] 王永滨，石民勇，洪志国. 网络环境下集群渲染技术综述 [J]. 微电子学与计算机，2008，25 (9)：81－83.
[2] 张敏海，吴新开，张婷婷. 基于 JPEG 压缩编码算法的数字图像处理系统 [J]. 计算机系统应用，2012，21 (10)：135－138.
[3] 郑美芳，高晓蓉，王　黎，等. 赵全轲基于 JPEG 标准的图像压缩 DCT 变换 [J]. 信息技术，2008，(11)：118－120.
[4] Nilsson P. Hardware accelerated image compositing using OpenGL [A]. Proc of Usenix' 04 Annual Technical Conference [C]，Santiago Freenix Track，2004.
[5] 吴恩华，柳有权. 基于图形处理器 (GPU) 的通用计算 [J]. 计算机辅助设计与图形学学报，2004，16 (5)：601－612.
[6] 张庆丹，戴正华，冯圣中，等. 基于 GPU 的串匹配算法研究 [J]. 计算机应用，2006，26 (7)：1735－1737.
[7] 杨　珂，罗　琼，石教英. 图形处理器在数据库技术中的应用 [J]. 浙江大学学报：工学版，2009，43 (8)：1349－1360.
[8] 邱宇峰，曾国荪. 一种基于 GPU 的粒子系统火焰模拟 [J]. 计算机科学，2009，36 (12)：238－242.
[9] 杨志义，朱娅婷，蒲　勇. 基于统一计算设备架构技术的并行图像处理研究 [J]. 计算机测量与控制，2009，17 (4)：734－737.