基于 CUDA 技术模拟雷达余辉的方法

谢永亮,汤晓迪,刘尚富,曾海兵
(海军蚌埠士官学校,安徽 蚌埠 233000)

摘 要:分析了目前基于光栅显示器模拟雷达余辉的方法。针对实现逼真余辉效果存在的主要瓶颈,通过采用 CUDA 技术可以解决模拟余辉时庞大的计算量的问题。主要采用 CPU+GPU 的编程模式模拟余辉,在 GPU 中为每一个像素点创建一个并行执行的线程来完成整个屏幕像素的数据处理,使得余辉效果逼真、画面流畅、扫描速度大幅提高。

关键词:CUDA;CPU+GPU;雷达余辉;光栅扫描
中图分类号:TN391 **文献标识码**:A **文章编号**:1674-7720(2011)08-0061-03

Radar persistence vision simulative method based on CUDA

Xie Yongliang, Tang Xiaodi, Liu Shangfu, Zeng Haibing
(Naval Petty Officer Academy, Bengbu 233000, China)

Abstract: Raster-scan based persistence vision simulative methods are analyzed in this paper, and the choke point of those methods is presented. Based on CUDA simulative method, the problem of gigantic data operation could be solved easily. CPU+GPU programme method is adopted to simulate radar persistence vision, thread is created to deal with the data operation for each pixel which executes at the same time. Vivid persistence display and fluent image are got, and the scan speed can be promoted greaterly.

Key words: CUDA; CPU+GPU; persistence vision of radar; raster scan

在建立雷达虚拟操作系统或维修训练系统时,显示器的仿真效果直接影响模拟器的训练效果。目前制约余辉实现的主要瓶颈是余辉效果带来的庞大的计算量,使得效果较好的余辉扫描线转速难以超过 10 转/s,如果要提高转速,则需要以牺牲显示画质为代价。基于光栅扫描余辉模拟的主流方法有画线法、固定扇扫法、逐点消隐法,由于前两者图像易出现辐射状花纹及扫描速率不稳定,因此后者的应用较多,效果也明显强于前者[1]。本文在逐点消隐法的基础上应用 CUDA 技术,解决了运算量巨大的问题,在光栅显示器上得到了余辉效果逼真、画面流畅的余辉图形。

1 余辉仿真的瓶颈

传统的雷达 P 显采用示波管作为显示终端,其内部荧光材料具有指数型衰减的余辉效应,电子束扫描线圆周扫过屏幕将留下逐渐消隐的余辉[2]。但光栅显示器无法自动产生荧光粉的余辉效应,因此必须人为地模拟余辉效应。

软模拟通常采用光栅显示器,用计算机编程实现。光栅扫描显示器具有高亮度、高稳定度、大容量显示的图文处理能力、丰富的色彩及多灰度等级的优点。一般采用以下三种方法实现[3-4]。

(1)画线法较容易实现,原理是在屏幕上以画直线的方式画出每一角度的扫描线,形成每次画一个扇面的灰度递减的直线簇。但是当程序运行时,扫描线轨迹不断地在屏幕上转动,该方法不能无缝地覆盖整个扇扫区域,从而产生一个辐射状的固定花纹。

(2)固定扇扫法是在画线法基础上改进的一种仿真方法,控制扇形区域的圆心角,依次使不同扇形区域亮度减少。它虽然消除了辐射状花纹,但在没有目标到有目标信号时,由于数据量的增加会造成扫描线的转速不同。

(3)逐点消隐法,主要原理是将每个方位像素的亮度逐次递减,即每个点都必须被修改,这样整个屏幕画面亮度逐渐衰减。其产生的余辉效果比较逼真,扫描线转速也较稳定。

模拟逼真的余辉效果,一般采用逐点消隐法,十分逼真的余辉仿真需要非常高的数据吞吐率,要求在每一显示帧的时间内(一般为 60 Hz 的倒数约 16 ms)对屏幕

中所有像素进行一次衰减运算。以公认的高效算法,即查表法为例:对于一个像素点而言,最少需要1次读和2次写操作,分辨率为1 024×1 024的屏幕中会有1 024×1 024个像素点参与雷达回波的显示,数量约为1 M。即在16 ms的时间内需要进行1 M次读操作和2 M次写操作,分给每个像素点的时间为16 ns。由于Windows属于通用型操作系统,硬件操作过程极其复杂,无论如何也无法在16 ns内完成1次读和2次写操作。需要说明的是,现有的用PC实现的余辉仿真算法都是以牺牲画质为前提条件的,例如有的算法降低角度分辨率,有的算法只运算部分像素。

2 瓶颈的解决方案

为了解决此瓶颈,本文将国外主要应用于3D游戏设计的CUDA技术移植到余辉的模拟上。CUDA(统一计算设备架构)是NVIDIA公司在2007年推出的针对GPGPU(通用计算GPU)的一个全新构想,使专注于图像处理的GPU超高性能在数据处理和科学计算等通用计算领域发挥优势[5]。

GPU特别适合并行数据运算问题,同一个程序可操作许多并行数据元素,并具有高运算密度(算术运算与内存操作的比例),且在高密度运算时,GPU访问内存的延迟可以被掩盖。目前高端GPU计算性能已达到Teraflops(每秒万亿次浮点运算)级别,其运算速度远远高于CPU的速度[6-7]。2008年初国内建成的首套实验系统,其计算性能的理论峰值124 Teraflops,可用峰值82 Teraflops。

但是常规的GPU通用计算还存在以下问题[7]:编程过于繁杂,难以学习与使用,在非图形领域应用很不充分;GPU编程缺乏灵活性,对GPU性能的发挥有很大的限制。

而CUDA采用GPU+CPU的方式,通过标准C语言将GPU的众多的计算特性结合到一起,由线程来创建应用程序。程序代码在实际执行中分为两种,一种是运行在CPU上的主机代码,另一种是运行在GPU上的设备代码。它类似于CPU上的多线程程序,但与仅能有很少线程同时工作的多核CPU相比,GPU可以同时执行成千上万个线程[8-9]。CPU程序以异步的方式调用GPU核程序,GPU作为CPU的协处理器(CoProcessor)提供服务。

当前CUDA提供的主要功能如下[7]:

(1)在GPU上提供标准C编程语言。

(2)为在支持CUDA的NVIDIA GPU的并行计算提供统一的软硬件解决方案。

(3)支持CUDA的GPU能进行并行数据缓存和线程执行管理。

(4)经过优化的,从CPU到支持CUDA的GPU的直接上传、下载通道。

(5)CUDA驱动与DirectX和OpenGL等图形驱动程序兼容。

为了解决巨大计算量的问题,主要采用CPU+GPU的编程模式来模拟余辉,在GPU中为每一个像素点创建一个线程独立进行亮度衰减处理。由于每个像素的线程并行执行,完成整个屏幕像素的数据处理几乎不需要计算时间,真正花费时间的是画面绘制和翻转。因此绘制画面在后台表面进行,绘制完成后翻转到前台显示,这样绘制和显示可以同时进行,既为画面的绘制留足了时间,又能得到流畅不闪烁的画质。

3 采用CUDA技术来实现余辉效果

为了产生不同方位的扫描线,将方位、距离进行量化,由于扫描区域的分辨率为1 024×1 024,因此半径为512像素。由于扫描半径为512个像素,理论上只要角度量化数N大于3 217就不会出现显示死地址的现象[10],方位上量化为4 096个等分。这样初始生成一个4 096×512个像素的圆域。雷达P显中采用的是极坐标系,而在光栅显示器中采用的是直角坐标,通过坐标变换,将建立一张坐标变换表,如表1所示。

表1 极坐标—直角坐标查询表

r	θ			
	0	1	…	4 095
0	(512,512)	(512,512)		(512,512)
1	(512,511)	(512,511)		(512,511)
…	…	…	…	……
511	(512,1)	(513,1)		(511,1)

通过查表可以避免坐标变换带来的正余弦计算,方便地在极坐标和直角坐标间转换,从而节省大量的运算时间[11]。考虑到近距离区域,多个角度的距离单元会对应相同的像素点,首先为每个像素点定义一个属性的结构体:

```
typedef  struct
{    WORD  x;                          //屏幕直角坐标x
     WORD  y;                          //屏幕直角坐标y
     WORD  ScanlinePtIndex;            //该点在扫描线上的
                                       //距离索引
     BYTE  MapTo2Pt;                   //该点与同一条扫描上的
                                       //点是否重合
     BYTE  RadEnd;                     //标记该条扫描线处理完毕
}RADIUSPOINT;
```

为圆域内的点分配内存空间:

RADIUSPOINT m_pRadPtToLintPtMap=new RADIUSPOINT[4 096×512]。

对于同一条扫描线上相邻的两点,如果直角坐标相同就把MapTo2Pt设为1,标记为相同的点;如果相邻两点的直角坐标不相同,则把距离索引值赋给ScanlinePtIndex,每条线最后一个点设置RadEnd为1来标记每条线处理已完毕。对于相邻两条线上的点,如果当前线上点与前一条线上相邻4个点的直角坐标相等,设置为m_pPixelOverlap[i]=1,否则设为0。

考虑到余辉呈指数型衰减，而指数运算需要花费大量的时间，对于计算机，其最快的操作是取值和赋值，为了提高光栅扫描雷达显示系统的实时性，需要提高单位时间内能够处理的像素点个数。于是对指数运算采用查表法以提高速度，维护一张按角度划分的指数型衰减因子表 m_wAttenuation[4 096]以进行数值的取值和赋值操作。

同时还要建立一个 Brightness[4 096×512]的亮度表，来存储每个像素对应的 RGB 颜色值。

以上这些工作在程序的初始化中即完成，一经完成即可在后续的程序中直接调用。

通过 CUDA 编程时，GPU 可看作为可以并行执行非常多个线程的计算设备，执行并行计算的线程被组织成线程块（Block），每个线程块可以包含多达 512 个线程，而线程块又组成了栅格（Grid）。GPU 可以支持成百上千万个并行线程，于是可以为每个像素点开一个线程，这样每个像素点可以并行处理，能极大地提高对整个屏幕像素的处理速度，为 CPU 留出足够多的时间去处理其他相关的任务。

定义线程块 Block 包含的线程维数：

```
dim3 threads(BLOCK_SIZE,BLOCK_SIZE);
```

定义栅格 Grid 包含的线程块数：

```
dim3 grid(Width/threads.x,Height/ threads.y);
```

每个像素点对应的线程处理工作如下：

由于某型雷达转速为 10 转/min，相当于每次更新的扫描线数应为 4 096×10/60/1 000=0.683 条/ms，像素处理在 GPU 中并行进行，对 CPU 的占用率几乎为零，所消耗的时间主要是 Direct3D 纹理的绘制和表面的翻转，大约为 16 ms，因此每次更新的扫描线数目约为 16×0.683=10.928，即每次更新 11 条。将当前要更新的扫描线上的像素点设为初始亮度，其后的每条扫描线上的像素点的亮度按与当前扫描线角度差 m_anglediff 取 m_wAttenuation[m_anglediff]的亮度进行衰减。由于近距离区域多个角度的距离单元对应相同的像素点，因此中心部位被消隐的次数明显要比其他部位多，导致效果有些失真。于是需要对这些坐标相同的点进行处理，对于属性 MapTo2Pt 为 1 的点，比较坐标相同的点处于不同距离时的亮度，取其大者赋值给亮度表 Brightness[4 096×512]。对于属性 m_pPixelOverlap 为 1 的点，比较处于各个角度时的亮度，取其大者赋值给亮度表。这样对于同一个点只显示一次且取其最亮者显示，较好地避免了中心部位被消隐次数过多的情况。

对于实现余辉等级的情况，只需要调制 m_wAttenuation 的大小就可以方便地调节余辉等级。如果需要提高转速，只需增大每次更新的扫描线数目即可，且基本不会影响程序运行速度。

通过 CPU+GPU 组合的方式模拟不同等级余辉效果如图 1、图 2 所示，此时对应的 CPU 占用率几乎为零，如图 3 所示。该方法得到的余辉效果逼真、画面流畅、扫描速度达到了预定的 10 转/s 的要求，且 CPU 占用率极

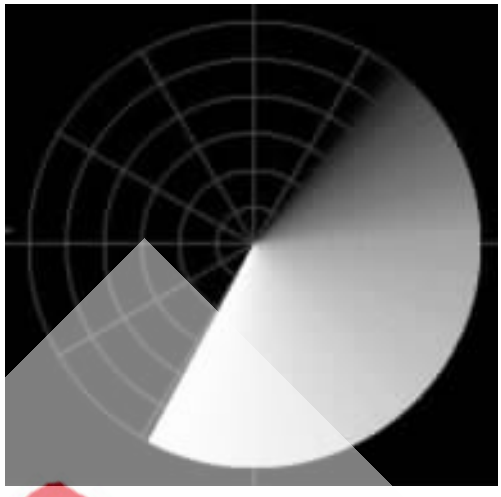

图 1 短余辉效果示意图

图 2 长余辉效果示意图

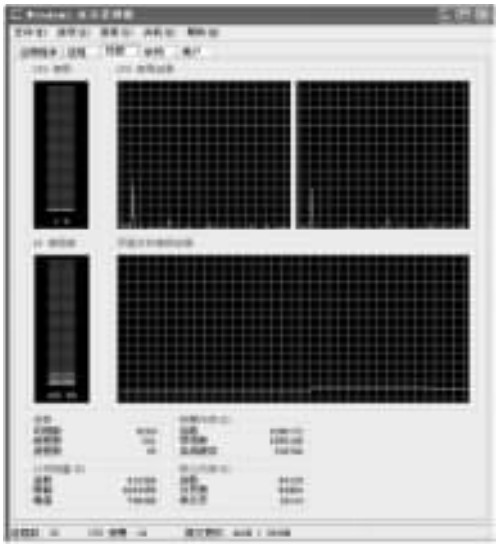

图 3 程序运行时 CPU 的占用率

低，并不妨碍 CPU 处理其他数据。

当把每次需要更新的扫描线数目增多时，由于 GPU 能并行高速处理每个像素点，扫描的速度能迅速提升而不影响显示画质，在程序调试时，可以验证当扫描速度到 45 转/min 时，画面依然流畅且占用的系统资源少。

余辉实现的逼真程度很大程度上决定了雷达模拟器的效果，本文就当前余辉模拟存在的瓶颈提出了一种基于 CUDA 的解决方案，采用“CPU+DPU”编程的方法，很好地解决了数据吞吐量巨大的问题。此方法模拟的余辉易于与雷达回波信号叠加，便于程序的扩展，可以应用于模拟器的设计及雷达技术的研发。

参考文献

[1] 朱兵.基于余辉地址表的雷达显示余辉模拟方法[J].舰船电子对抗,2007,30(3):37-39.

[2] 张泽润.船舶导航雷达[M].北京:人民交通出版社,1990.

[3] 樊世友,杨作宾.基于余辉模型的 P 型雷达显示器计算机仿真[J].计算机仿真,2003,20(4):6-8.

[4] 刘翠海,温东.光栅扫描显示器上实现 PPI 雷达长余辉仿真[J].计算机仿真,2002,19(2):25-27.

[5] RUEDA A J,ORTEGA L.Geometric algorithms on CUDA [J].GRAPP,2008,39(6):59-60.

[6] Wu Enhua,Liu Youquan.General purpose computation on GPU[J].Journal of Computer-aided Design & Computer Graphics,2004,16(5):601-611.

[7] 多相复杂系统国家重点实验室多尺度离散模拟项目组.基于多 GPU 的多尺度离散模拟并行计算[M].北京:科学出版社,2009.

[8] 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报,2004,16(5):601-611.

[9] RANDINA F.GPU 精粹——实时图形编程的技术、技巧和技艺[M].姚勇,工小琴,译.北京:人民邮电出版社,2006.

[10] 徐展翼,欧阳宁,韩传久.高速即工光栅扫描显示系统坐标转换设计与实现[J].桂林电子工业学院学报,2003,23(1):14-18.

[11] 刘翠海,王文清,袁满.一种支持雷达 P 显仿真的实时坐标变换策略[J].系统仿真学报,2002,14(9):57-60.

(收稿日期:2010-11-25)

作者简介:

谢永亮,男,1984 年生,硕士研究生,主要研究方向:雷达信号处理及技术保障。