第 42 卷 第 6A 期
2015 年 6 月

计 算 机 科 学
Computer Science

Vol. 42 No. 6A
June 2015

# 基于 CUDA 和 OpenGL 互操作的彩色图像 Sobel 边缘检测

李驰新 兰聪花

(兰州工业学院电子信息工程学院 兰州 730050)

**摘 要** 为了充分发挥 GPU 通用计算的能力,将以往由 CPU 承担的计算任务更多地移交给 GPU 完成,使用了基于 CUDA 和 OpenGL 的互操作技术,完全依靠 GPU 完成彩色图像的边缘检测和结果显示任务,CPU 只负责传递数据到 GPU,既提高了整个过程的计算速度,也最大程度地发挥了 GPU 的功效。实验结果表明,基于 CUDA 和 OpenGL 的互操作的图像处理技术是一种有效结合 GPU 并行处理能力和 GPU 图像显示技术的解决方案,与只使用 CPU 和只使用 GPU 并行计算的方案相比,本方法在处理高分辨率的图像时,可以获得 80 倍以上的加速比。

**关键词** 边缘检测,互操作,CUDA,OpenGL,Sobel

**中图法分类号** TP391.41,TP911.73 **文献标识码** A

## Sobel Edge-detection on Color Image Based on Interoperability between CUDA and OpenGL

LI Chi-xin LAN Cong-hua

(Electronics & Informatics College, Lanzhou Institute of Technology, Lanzhou 730050, China)

**Abstract** With the quick development of general-computing on GPU, many jobs which were once implemented by CPU now can be delivered to GPU. In this paper, depending on the interoperability between CUDA and OpenGL, jobs of edge-detection on color image and displaying result are all finished by GPU, the only job for CPU is delivering data to GPU. By doing so, computing speed is increased and efficiency of GPU is maximized. The experimental results indicate that interoperability between CUDA and OpenGL is an effective method which can combine parallel processing capability of GPU and displaying capability of GPU. Compared with other 2 kinds of scheme which only uses CPU to process image and only uses GPU to parallelly compute data, this kind of scheme can achieve 80 times speedup when processing high resolution image.

**Keywords** Edge-detection, Interoperability, CUDA, OpenGL, Sobel

最近几年,以 GPU 为核心的异构系统快速发展,GPU 可以承担的通用计算任务越来越多。由于图像数据的松耦合性,使得 GPU 在图像处理领域中获得快速普及[1]。目前 GPU 通用计算标准有 CUDA、OpenCL、C++AMP 等多种,基于 CUDA 的 Sobel 实现主要是针对灰度图像,对实现速度的测量集中于 kernel 函数[2,3],未考虑全部程序运行时间。CUDA 提供了针对 OpenGL 的完善支持,利用 CUDA 与 OpenGL 的互操作,以往经过 GPU 处理之后需要送回 CPU 的数据,可以省略回送步骤,直接由 GPU 在屏幕上显示,节省了数据处理时间,极大提高了程序所需要的全局速度。

## 1 边缘检测

边缘检测是图像处理最基本的技术之一,目前使用 CUDA 进行边缘检测的论文和学术著作已有很多,但大多数都是针对灰度图像实现。本文使用 Sobel 算法,对彩色图像进行边缘检测,对目前已有的实例,例如 CUDA Samples 里面提供的 SobelFilter 程序,做出了技术上的改进[3,4]。

Sobel 有各向同性算子(Isotropic Sobel)和普通算子 2 种,2 种算子都是由 2 个 $3*3$ 的矩阵组成。不失一般性,本文只讨论后者,2 个矩阵分别用来计算横向和纵向亮度差分近似值,矩阵计算如下所示[5]:

$$G_x=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix}*A$$

$$G_y=\begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}*A$$

式中,$A$ 代表原始图像,$G_x$ 及 $G_y$ 分别代表经横向及纵向边缘检测的图像。图像某一点的梯度和方向的计算如下式:

$$G=\sqrt{G_x^2+G_y^2}$$

$$\Theta=\arctan\left(\frac{G_y}{G_x}\right)$$

Sobel 算子简单有效,适合于快速检测图像边缘;缺点是未能模拟人的视觉特征,在处理复杂场景时,不能很好地把背景和轮廓有效区别出来。

## 2 CUDA 与 OpenGL 互操作

### 2.1 CUDA 与 OpenGL 互操作的基本原理

CUDA 与 OpenGL 的相互结合可以通过 2 种方式来实现,即像素缓冲区对象(Pixel Buffer Object,PBO)和顶点缓冲

李驰新(1978—),男,硕士,讲师,主要研究方向为并行计算和图像处理;兰聪花(1979—),女,硕士,讲师,主要研究方向为计算机网络。

区对象(Vertex Buffer Object,VBO),2 种方法的基本原理相同,都是将 OpenGL 的高速缓冲映射到 CUDA 的显存中。CUDA 处理数据之后,直接由 OpenGL 在屏幕上显示,不需要再将数据回送到 CPU,大大提高了整个程序的运行速度[6]。因此,CUDA 与 OpenGL 的结合是处理图像数据的最好选择。

PBO 处理像素数据,适合于图像的显示;VBO 处理顶点数据,适合于构建几何形状。PBO 速度优势的原理可以用图 1 和图 2 说明。图 1 表明,不使用 PBO,CPU 需要控制内存和显存之间的数据传递和显示,占用了 CPU 时间。图 2 表明,使用 PBO 时,CPU 只需将数据从内存传递到显存,其余任务由 GPU 在 OpenGL 控制下完成,不占用 CPU 时间,也就是说 GPU 以存储器直接访问(Direct Memory Access,DMA)方式工作。


图 1 像素加载不使用 PBO

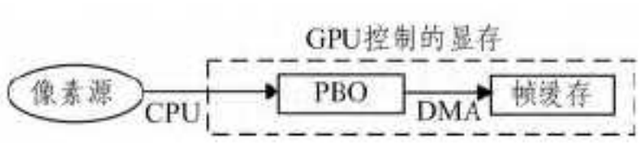


图 2 像素加载使用 PBO

### 2.2 实现 CUDA 与 OpenGL 互操作的基本步骤

在程序设计实现 CUDA 与 OpenGL 互操作时,需要经过以下几个步骤以及代码设定。

(1)创建 PBO

创建一个 PBO:

```
glGenBuffers(1,&bufferObj);
```

绑定 PBO:

```
glBindBuffer(GL_PIXEL_UNPACK_BUFFER_ARB,bufferObj);
```

为 PBO 分配数据:

```
glBufferData(GL_PIXEL_UNPACK_BUFFER_ARB, width * height * 3,NULL,GL_DYNAMIC_DRAW_ARB);
```

(2)注册 PBO

```
cudaGraphicsGLRegisterBuffer(&resource, bufferObj, cudaGraphicsMapFlagsNone);
```

(3)映射 PBO

把 PBO 缓冲映射到 CUDA 显存,函数中的 devPtr 是传递给 kernel 函数的指针

```
cudaGraphicsMapResources(1,&resource,NULL);
cudaGraphicsResourceGetMappedPointer((void**)&devPtr,&size,resource);
```

(4)使用 CUDA kernel 函数写入数据

执行 kernel 函数,launch_kernel 是 GPU 上执行的程序

```
launch_kernel(devPtr);
```

(5)解除映射

```
cudaGraphicsUnmapResources(1,&resource,NULL);
```

(6)解除注册

```
cudaGraphicsUnregisterResource(resource);
```

(7)删除 PBO

```
glBindBuffer(GL_PIXEL_UNPACK_BUFFER_ARB,0);
glDeleteBuffers(1,&bufferObj);
```

CUDA Samples 提供的 SobelFilter 实例,比以上步骤多了一步,即设置 OpenGL 纹理,把缓存里面的数据以纹理贴图的形式显示到屏幕上[7,8]。OpenGL 纹理的纹理设置比较复杂,不考虑这种方法,本文直接使用 OpenGL 中的 glDrawPixels语句,把显示缓存中的数据以写像素的形式显示到屏幕上,使程序简练,理论上显示速度也更快。

## 3 程序实现

为了验证 CUDA 与 OpenGL 互操作在图像处理中的优势,本文设计了 4 种方案来对比它们处理彩色图像的时间消耗。

方案 1 完全基于 CPU 进行边缘检测,将彩色图像的 RGB 通道分离,CPU 分别进行每个通道的 Sobel 边缘检测,最后将 3 个通道的处理结果合并[9-12]。

方案 2 由 CPU 完成 RGB 通道分离,依次将每一个通道的数据送往 GPU 处理,3 个通道的数据被 GPU 依次完成边缘检测之后,将处理结果送回 CPU,CPU 将 3 个通道的数据结果合并,由 CPU 显示最后结果。在这个方案之中,每一个通道的数据在 CPU 和 GPU 之间往返 2 次,因此一幅图像的处理需要 6 次数据往返。

方案 3 CPU 将所有图像数据送往 GPU,GPU 负责数据的 3 个通道分离和边缘检测,之后将最后的结果送回 CPU,CPU 只负责显示结果,与方案 2 相比,所有数据只需要 2 个数据往返。

方案 4 CPU 只负责将图像数据送往 GPU,GPU 完成通道分离、边缘检测、数据显示,与前 2 种方案相比,在 CPU 和 GPU 之间只有 1 次数据传递。

程序实现的软硬件环境如下:

硬件:CPU Intel 3770K,GPU Nvidia GTX760,内存 4G;

软件:Windows 7,Microsoft visual studio 2010,CUDA 4.2,Freeglut 2.8,OpenCV 2.4。

针对一幅彩色图像 Sobel 边缘检测的结果如图 3 所示,同时,选择了 5 种不同分辨率的图像,来验证每一种方案花费的时间,边缘检测所花费的时间如表 1 所列。根据表 1 的数据,以方案 1 作为基准,制作了方案 2、3、4 的加速比折线图,如图 4 所示。

图 3 原始图像与边缘检测结果

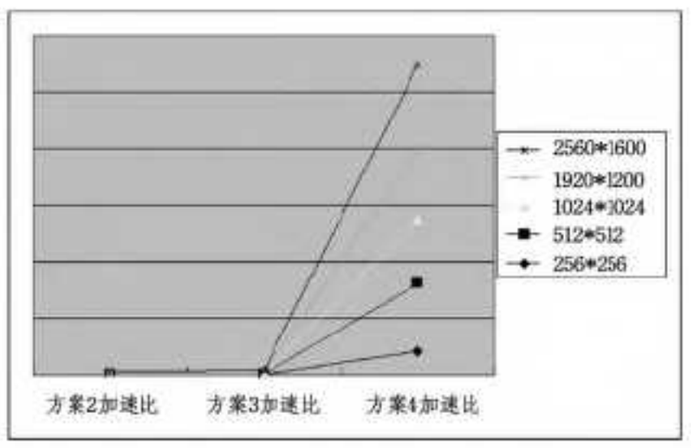


图 4 3 种方案加速比折线图

(下转第 230 页)

从表 1 可以看出，文献[9,10]与本文方法检测真实相似物体的准确率均在 95% 以上。但是文献[9]方法中，当 $QF$ 低于 50，$SNR$ 低于 15 时，准确率急剧下降；文献[10]对 JPEG 压缩有较好的抵抗力，但是对高斯白噪声表现较为敏感；本文算法对高斯白噪声和 JPEG 压缩攻击均有很好的鲁棒性，准确率不低于 90%，并且本文方法对背景复杂的图像亦能有效检测。

结束语　本文针对在复制篡改领域中真实相似物体易引起检测结果的假阳性，提出了真实相似物体检测算法。该算法具有以下特点：(1)利用双向匹配严格的匹配条件，对提取的 SIFT 特征点进行匹配，使匹配更为精确。(2)利用仿高斯影响函数评估点集之间影响程度，并对局部可达密度进行加权继而计算局部离群因子，使得到的特征数据更精确。(3)簇内紧凑度的估计基于 AP 聚类的基础上，降低无特征点区域对整体紧凑度的影响，减少运算次数，同时提高检测准确率。实验结果表明该方法对 JPEG 压缩和高斯白噪声具有良好的鲁棒性。最后，进一步降低算法复杂度是今后研究的重点。

## 参 考 文 献


[1] Farid H. Creating and detecting doctored and virtual images: implications to the child photography prevention act[R]. Technical Report, TR2004-518, Dartmouth College, Computer Science, 2004

[2] 骆伟祺，黄继武，丘国平. 鲁棒的区域复制图像篡改检测技术[J]. 计算机学报，2007，30(11)：1998-2007

[3] 秦娟，李峰，向凌云. 采用圆谐-傅里叶矩的图像区域复制粘贴篡改检测[J]. 中国图像图形学报，2013，18(8)：919-923

[4] 赵洁，郭继昌，武斌，等. 基于几何均值分解的图像区域复制篡改检测方法[J]. 小型微型计算机系统，2012，33(9)：2105-2108

[5] Tao W, Jin T, Bin L. Blind Detection of Region Duplication Forgery by Merging Blur and Affine Moment Invariants[C]// 2013 Seventh International Conference on Image and Graphics. 2013：258-264

[6] 左菊仙，刘本永. 伪造图像典型篡改操作的检测[J]. 中国图象图形学报，2012，17(11)：1367-1375

[7] Pan X Y, Lyu S W. Region duplication detection using image feature matching[J]. IEEE Transactions on Information Forensics and Security, 2010, 5(4)：857-867

[8] Jaberi M, Bebis G, Hussain M, et al. Accurate and robust localization of duplicated region in copy-move image forgery[J]. Machine Vision and Applications, 2014, 25(2)：451-475

[9] Muhammad G, Muhammad H, Khawaji K, et al. Blind copy move image forgery detection using dyadic undecimated wavelet transform[C]// 17th International Conference on Digital Signal Processing. 2011：1-6

[10] Muhammad G, Muhammad H, Bebis G. Passive copy move image forgery detection using undecimated dyadic wavelet transform[J]. Digital Investigation, 2012, 9(1)：49-57

[11] Shao H, Yu T S, Xu M J, et al. Image region duplication detection based on circular window expansion and phase correlation [J]. Forensic Science International, 2012, 222(1/3)：71-82

[12] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2)：91-110

[13] 刘笑楠，苑玮琦，张波. 一种虹膜色素块检测与分类方法[J]. 沈阳工业大学学报，2014，06：688-693


(上接第 222 页)

表 1　4 种方案的处理时间

| 分辨率 | 256×256 | 512×512 | 1024×1024 | 1920×1200 | 2560×1600 |
|---|---|---|---|---|---|
| 方案 1 时间(ms) | 5.50 | 17.36 | 70.65 | 147.52 | 256.08 |
| 方案 2 时间(ms) | 91.30 | 101.11 | 146.91 | 194.77 | 281.16 |
| 方案 3 时间(ms) | 101.21 | 90.92 | 91.47 | 97.39 | 190.04 |
| 方案 4 时间(ms) | 0.27 | 0.29 | 1.26 | 2.35 | 3.44 |

从表 1 可以看出，方案 4 获得了最好的加速比；方案 2 因为在 CPU 和 GPU 之间存在多次数据的往返，程序消耗了最多时间；方案 3 只有在图像分辨率较高的情况下获得一定的加速比，在图像分辨率较低、数据量较小的情况下，处理速度反而不如 CPU 直接处理快。由此可以看出，目前使用 GPU 进行通用计算的最大瓶颈在于 CPU 和 GPU 之间的数据传输带宽

结束语　GPU 通用计算的出现为许多传统算法提供了新的解决方案和途径，CUDA Samples 提供的 SobelFilter 实例虽然实现了灰度图像的边缘检测，但通过运行程序，我们发现 SobelFilter 只能对 pgm 格式的图像实现运算。本文设计了新的程序，通过引入 OpenCV，不但可以对 jpg、bmp 等多数主流格式的图像实现彩色边缘检测，而且进一步简化了互操作的设置步骤。通过多种方案的比较，验证了基于 CUDA 与 OpenGL 互操作方案在图像处理中拥有巨大的效率优势。

## 参 考 文 献


[1] NVIDIA. CUDA Compute Unified Device Architecture: Programming Guide(Version 4.2)[EB/OL]. [2011-11]. http://www.nvidia.com/object/cuda_home.html

[2] 邢军. 基于 Sobel 算子数字图像的边缘检测[J]. 微机发展，2009，30(14)：3360-3361

[3] Sonka M, Hlavac V. Image Processing, Analysis, and Machine Vision [M]. Tsinghua University Press, 2011

[4] NVIDIA. CUDA SDK-Graphics Interop [EB/OL]. http://www.nvidia.cn/content/cudazone/cuda_sdk/Graphics_Interop.html

[5] Podlozhnyuk V. Image Convolution with CUDA[EB/OL]. 2007-01-06. http://www.nvidia.com/object/cuda_home.html

[6] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture: Programming Guide (Version 4.2)[EB/OL]. http://www.nvidia.com/object/cuda_home.html

[7] Ahn S H. OpenGL Pixel Buffer Object[EB/OL]. http://www.songho.ca/opengl/gl_pbo.html

[8] CSDN. CUDA 与 OpenGL 交互开发[EB/OL]. http://blog.csdn.net/ruby97/article/details/8851403

[9] Zuo H R. Fast Sobel Edge Detection Algorithm Base on GPU [J]. Opto-Electronic Engineering, 2009(1)：8-12

[10] 谭立勋，刘缠牢，李春燕. 实时图像处理中 Sobel 算子的改进[J]. 弹箭与制导学报，2006，26(1)：291-293

[11] Kharlamov A, Podlozhnyuk V. Image Denoising [EB/OL]. 2007-05-16. http://www.nvidia.com/object/cuda_home.html

[12] 左颢睿，张启衡. 基于 GPU 的快速 Sobel 边缘检测算法[J]. 光电工程，2009，36(1)：10-12