Computer Science and Application 计算机科学与应用, 2013, 3, 153-158
http://dx.doi.org/10.12677/csa.2013.33027 Published Online June 2013 (http://www.hanspub.org/journal/csa.html)

GPU-Based Parallel Prewitt Algorithm Implementation and Its Application on GPR*

Tuyou Peng[1], Qinghua Dong[2]

[1]Department of Electronics & Information Technology, Jiangmen Polytechnic, Jiangmen
[2]Department of Civil Engineering and Architecture, Wuyi University, Jiangmen
Email: typeng2001@yahoo.com, dqh0359@126.com

Received: Mar. 27th, 2013; revised: Apr. 12th, 2013; accepted: Apr. 21st, 2013

Abstract: The edge of image is one of the important features of the image. Edge detection is an important means to extract image features. GPU-CUDA parallel technology, as the most popular high-performance processing technology, is the best choice for parallel Prewitt algorithm implementation. Since conventional Prewitt algorithm based on CPU is computationally intensive and time-consuming, its application is very restricted. In order to improve the efficiency of Prewitt algorithm, GPU-based parallel Prewitt algorithm and fast imaging algorithm are applied to get higher speedup. Finally, the method is proposed by turning the GPR field data into gray-scale image data, then implementation of GPR field data processing with the Prewitt algorithm based on GPU. It is proved that the algorithm is not only of high efficiency, but also effective to improve target identification capability.

Keywords: Edge Detection; Prewitt Algorithm; GPU-CUDA; Ground Penetrating Radar; Imaging

基于 GPU 的 Prewitt 算法实现及其在探地雷达中的应用*

彭土有[1]，董清华[2]

[1]江门职业技术学院电子与信息技术系，江门
[2]五邑大学土木建筑系，江门
Email: typeng2001@yahoo.com, dqh0359@126.com

收稿日期：2013 年 3 月 27 日；修回日期：2013 年 4 月 12 日；录用日期：2013 年 4 月 21 日

摘　要：图像的边缘是图像的重要特征之一，边缘检测是提取图像特征的重要手段。GPU-CUDA 并行技术作为当前最热门的高性能处理技术，是并行 Prewitt 边缘检测算法实现的首选。由于常规的基于 CPU 的 Prewitt 算法计算量大、耗时，其应用受到很大限制。为了提高算法的效率，文中应用 GPU-CUDA 技术实现 Prewitt 算法及快速成像，获得了较高的加速比。最后，通过将实测探地雷达数据转换成灰度图像数据，并实施基于 GPU 的 Prewitt 算法处理的方法思路，对实测探地雷达数据进行处理。试验结果表明该算法不仅运行效率高，而且在突出有效异常，提高目标体的识别能力方面取得实效。

关键词：边缘检测；Prewitt 算法；GPU-CUDA；探地雷达；成像

1. 引言

边缘检测是图像处理和计算机视觉中的基本方法，构成了图像特征提取的重要研究领域。边缘检测的目的是标识数字图像中亮度变化明显的点。图像属性中的显著变化通常反映了属性的重要事件和变化。例如：深度上的不连续、表面方向上的不连续、物质

*基金项目：基于 Linux 地质雷达一体化处理解释技术及应用基金项目：2011 年省部产学研结合项目(2011B090400120)。

属性的变化和场景照明的变化等。因此，物质属性的变化必然在探地雷达 GPR (Ground Penetrating Radar)成像中有所反映[1]。本文探讨应用近年来新发展起来的 GPU-CUDA[2]技术实现 Prewitt 边缘检测算法，提出将实测探地雷达数据转换成灰度图像数据后，再实施基于 GPU 的 Prewitt 算法处理的方法思路，对实测探地雷达数据进行处理[3]和基于 GPU 的快速成像，以求取得好的应用效果。试验结果表明该算法不仅运行效率高，较好的解决了探地雷达一体化处理解释中的图形冻结问题，而且在突出有效异常，提高目标体的识别能力方面取得实效。

2. Prewitt 算法原理

多年以来，人们提出了许多以微分法[4]为基础的边缘检测算法，并以数据块与模板卷积的手段提取图像边缘。常用边缘检测算法包括 Roberts Cross 算法、Prewitt 算法、Sobel 算法、Canny 算法和 Laplace 算法。Prewitt 算法作为一阶微分算子的边缘检测算法[5]，它利用像素点左右、上下相邻点的灰度差在边缘处达到极值的重要特征来检测边缘和去掉部分伪边缘，对噪声具有平滑作用。其原理是在图像空间中，将左右、上下相邻点构成的 3 × 3 数据块(图 1)与水平、垂直两个方向模板进行卷积运算来完成。其中：水平模板(图 2)用于检测水平边缘，垂直模板(图 3)用于检测垂直边缘。

对于像素点 $p(i,j)$，Prewitt 算法定义如下：

水平方向梯度：

$$x = p(i-1,j-1)+p(i-1,j)+p(i-1,j+1) - p(i+1,j-1)-p(i+1,j)-p(i+1,j+1)$$

P(i − 1, j − 1)	P(i − 1, j)	P(i − 1, j + 1)
P(i, j − 1)	P(i, j)	P(i, j + 1)
P(i + 1, j − 1)	P(i + 1, j)	P(i + 1, j + 1)

Figure 1. *P*(*i*,*j*) pixel and its 3 × 3 pixel neighborhood
图 1. *P*(*i*,*j*)像素点及其 3 × 3 像素邻域

1	1	1
0	0	0
−1	−1	−1

Figure 2. Horizontal template
图 2. 水平模板

−1	0	1
−1	0	1
−1	0	1

Figure 3. Vertical template
图 3. 垂直模板

垂直方向梯度：

$$y = p(i-1,j-1)+p(i,j-1)+p(i+1,j-1) - p(i-1,j+1)-p(i,j+1)-p(i+1,j+1)$$

梯度：$g(i,j)=\left(x^2+y^2\right)^{1/2}$

其中：x 为像素点 $p(i,j)$邻域的 3 × 3 数据块与水平模板的卷积；y 为像素点 $p(i,j)$邻域的 3 × 3 数据块与垂直模板的卷积；计算出梯度 $g(i,j)$后，把 $g(i,j)$值大于 255 的点标记为边界点，其像素值设定为 255(即白点)，否则设定像素值为 $g(i,j)$的值。Prewitt 算法的串行实现简单，但是，由于需要做大量的卷积运算，当图像分辨率较高时，计算量很大，需要较长的运算时间，难以满足实际工作要求。

3. Prewitt 算法的 GPU-CUDA 实现

通过 Prewitt 算法分析可知：计算过程中不需要利用已经完成的像素处理结果参与，符合 GPU-CUDA (Compute Unified Device Architecture)数据并行处理要求。因此，可以将 Prewitt 算法中的所有单线程循环改为 GPU-CUDA 多线程并行处理，从而提高算法的效率。根据 CUDA 程序设计思想(即 CPU 进行数据准备、初始化 GPU 设备以及执行串行代码，GPU 进行并行计算并向 CPU 返回计算结果)，按如下算法过程完成 Prewitt 算法在 GPU-CUDA 的实现：

1) 由主机端程序完成数据的初始化；

2) 申请两块与图像数据相等容量的设备内存空间，其中一块用于存放由主机传送过来的图像数据，另一块用于存放设备中内核并行程序进行 Prewitt 算法处理后的结果；

3) 将主机端图像数据拷贝回至设备端；

4) 调用设备内核并行程序；

5) 将处理后的图像数据拷贝回主机端。

Prewitt 并行算法的关键是并行处理网格的构造以及内核并行程序实现，构造并行处理网格源代码如下[6]：

```
int lineByte = screenWidth*sizeof(float);//计算图
```

像数据大小

cudaMalloc((void **)&d_DataIn,lineByte*screenHigh);//申请设备内存空间

cudaMalloc((void **)&d_DataOut,lineByte*screenHigh);//申请设备内存空间

cudaMemcpy(d_DataIn,h_DataIn, lineByte*screenHigh,cudaMemcpyHostToDevice);//图像数据拷贝至设备端

dim3 dimBlock1(256,1);//构造并行处理网格

dim3 dimGrid1(((screenWidth-2)*(screenHigh-2) + 255)/256,1);//构造并行处理网格

prewittProcess<<<dimGrid1,dimBlock1>>>(d_DataIn,d_DataOut,screenWidth,screenHigh);//调用内核并行处理函数

cudaMemcpy(h_DataOut,d_DataOut,lineByte*screenHigh,cudaMemcpyDeviceToHost);//处理结果拷贝回主机端

内核并行程序实现源代码如下：

__global__ void prewittProcess(float *d_DataIn, float*d_DataOut,int screenWidth,int screenHigh)

{unsigned int id = 256*gridDim.x + threadIdx.x;//计算并行处理网格的索引号 id

if(id<(screenWidth-2)*(screenHigh-2))//控制图像处理范围

{int row=id/(screenWidth-2) + 1;//图像矩阵行编号 row

int col=id%(screenWidth-2) + 1; //图像矩阵列编号 col

int x,y,result;

x=*(d_DataIn+(row-1)*screenWidth+(col-1))+*(d_DataIn+(row-1)*screenWidth+(col))+*(d_DataIn+(row-1)*screenWidth + (col+1))-*(d_DataIn+(row+1)*screenWidth+(col-1))-*(d_DataIn+(row+1)*screenWidth+(col))-*(d_DataIn+(row+1)*screenWidth+(col+1));//计算水平方向梯度

y=*(d_DataIn+(row-1)*screenWidth+(col-1))+*(d_DataIn+row*screenWidth+(col-1))+*(d_DataIn + (row + 1)*screenWidth + (col-1))-*(d_DataIn + (row-1)*screenWidth+(col+1))-*(d_DataIn+row*screenWidth+(col+1))-*(d_DataIn + (row + 1)*screenWidth+(col+1));//计算垂直方向梯度

result=x*x+y*y;//计算梯度值 $g(i,j)$的平方值

*(d_DataOut+row*screenWidth+col)=result;//保存 Prewitt 并行处理结果}}

实验在 Acer ASPIRE 4736G(含 GEFORCE G105M CUDA 512M 图形卡)手提电脑上完成，试验计算结果如表 1 所示，获得了较好的加速比，满足了对探地雷达数据进行实时交互处理解释的实际需要，圆满地解决了操作界面冻结问题。

4. Prewitt 算法在探地雷达中的成功应用

4.1. 实测雷达数据转化为灰度图像数据

应用 Prewitt 边缘检测技术处理探地雷达数据的前提是需要把实测雷达数据转化为灰度图像数据。实测雷达数据是按道存储，必须在转化为灰度图像数据的同时进行行列变换。具体实现方法为[7]：遍历所有实测雷达数据求取最大、最小值，将实测雷达按数据最大值对应 255，最小值对应 0 的准则进行线性变换求取灰度图像数据。与此同时，完成行列变换后成为真正的灰度图像数据。主要代码如下：

for(int i=0;i<screenHigh;i++)//遍历所有实测雷达数据求取最大、最小值

for(int j=0;j<screenWidth;j++)

{if(min>*(h_DataIn+i*screenWidth+j)) min=*(h_DataIn+i*screenWidth+j);

if(max<*(h_DataIn+i*screenWidth+j)) max=*(h_DataIn+i*screenWidth+j);}

for(int i=0;i<screenHigh;i++)//每点数据进行行列变换，同时转换为灰度图像数据。

for(int j=0;j<screenWidth;j++)

{*(h_DataIn+i*screenWidth+j)=((255**(h_DataIn+i*screenWidth+j)/(max-min)-255*min/(max-min)));}

Table 1. Time comparison of CUP and GPU on different resolutions
表 1. 不同分辨率图像 CUP 与 GPU 处理时间对比

处理时间 单位：秒	640 × 480	1024 × 768	1280 × 768
CPU	2.245	4.018	5.323
GPU	0.458	0.461	0.551
加速比	4.90	8.72	9.66

以上预处理工作完成后，调用基于 GPU-CUDA 实现的 Prewitt 算法对灰度图像数据进行边缘检测。边缘检测结果再按如下实现的 GPU 快速彩色成像方法成像。

4.2. Prewitt 处理结果的 GPU 快速彩色成像

探地雷达 GPU 快速彩色成像算法的关键是在色标数据基础上，如何实现色标数据的转换及彩色成像的 GPU 多线程处理[8]？文中通过 GPU 多线程并行并读写 OpenGL 成图缓冲区方法直接成像，极大地提高了探地雷达成像速度，解决了当前探地雷达彩色成像速度慢造成程序操作界面冻结等问题。探地雷达 GPU 快速彩色成像算法流程图如图 4 所示[9]。

根据图 4 所示流程图，设计快速彩色成像算法内核并行程序源代码如下：

__global__void prepareMap(float4 *positions,float *seismic, float *tmp_color, int screenWidth, int screenHigh, float min, float max, int colorTableLength, float TranspSliderValue)//每个线程负责生成一个像素点数据

{unsigned int i=blockIdx.x*blockDim.x + threadIdx.x;//计算并行处理网格的索引号 i

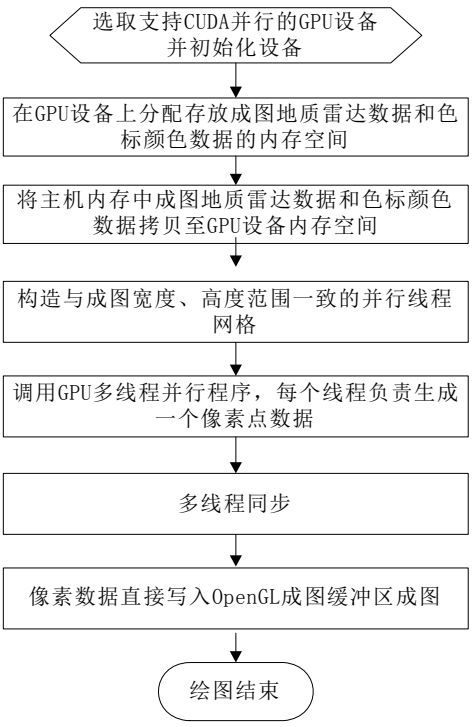

Figure 4. Flow chart of GPU fast imaging algorithm
图 4. GPU 快速彩色成像算法流程图

unsigned int j=blockIdx.y*blockDim.y+threadIdx.y;//计算并行处理网格的索引号 j

if(i<screenWidth && j<screenHigh)//成像数据范围：screenWidth*screenHigh

{int index=(int)(((colorTableLength-1)**(seismic+i*screenHigh+j)/(max-min)-(colorTableLength-1)*min/(max-min)));//根据定义的色标数据，计算成像数据

if(index<0)index=0; if(index>colorTableLength-1) index=colorTableLength-1; *(positions+i*screenHigh+j) =make_float4(*(tmp_color+index*3 + 0),*(tmp_color + index*3 + 1), *(tmp_color + index*3 + 2),TranspSlider Value);//生成彩色成像数据}}

4.3. 实验结果与分析

图 5 是实测探地雷达数据采用 GPU 快速彩色成像算法的直接成像，并包括探地雷达软件系统的总控界面；图 6 是实测探地雷达数据经过基于 GPU-CUDA 的 Prewitt 边缘检测处理后再采用 GPU 快速彩色成像算法的成像；由于 Prewitt 算法和彩色成像均采用 GPU 多线程并行处理，两者组合处理能达到近 10 倍的加速比，克服了常规 CPU 单进程处理速度过慢引起的窗口界面冻结问题。从图 6 看出，目标体周围的电磁波绕射干扰[10]受到明显压制，目标体的边界更加清晰。位于 A、B、C 三处的地下管线埋设情况及管线边界更加准确，其中，A、B 两处地下管是距地面埋藏深度只有 25 CM 的塑料管，C 处是人工砖砌排水管，埋藏深度 1.1 M(注：为了便于观察目标异常，已对上述成像在垂直方向进行了 1 M 的向下平移)。由此可见，应用 Prewitt 算法处理实测探地雷达数据对突出有效异常，提高目标体的识别能力效果明显，是一种提高探地雷达分辨率的有效方法。

5. 结语

在分析 Prewitt 边缘检测算法的基础上，首先设计并实现了基于 GPU-CUDA 的 Prewitt 快速算法。为了能够应用 Prewitt 快速算法于探地雷达的数据处理，文中提出了先将实测探地雷达数据转化为灰度图像数据，然后再调用 Prewitt 快速算法处理，最后应用 GPU 快速彩色成像算法成像的工作流程。实验结果表明：

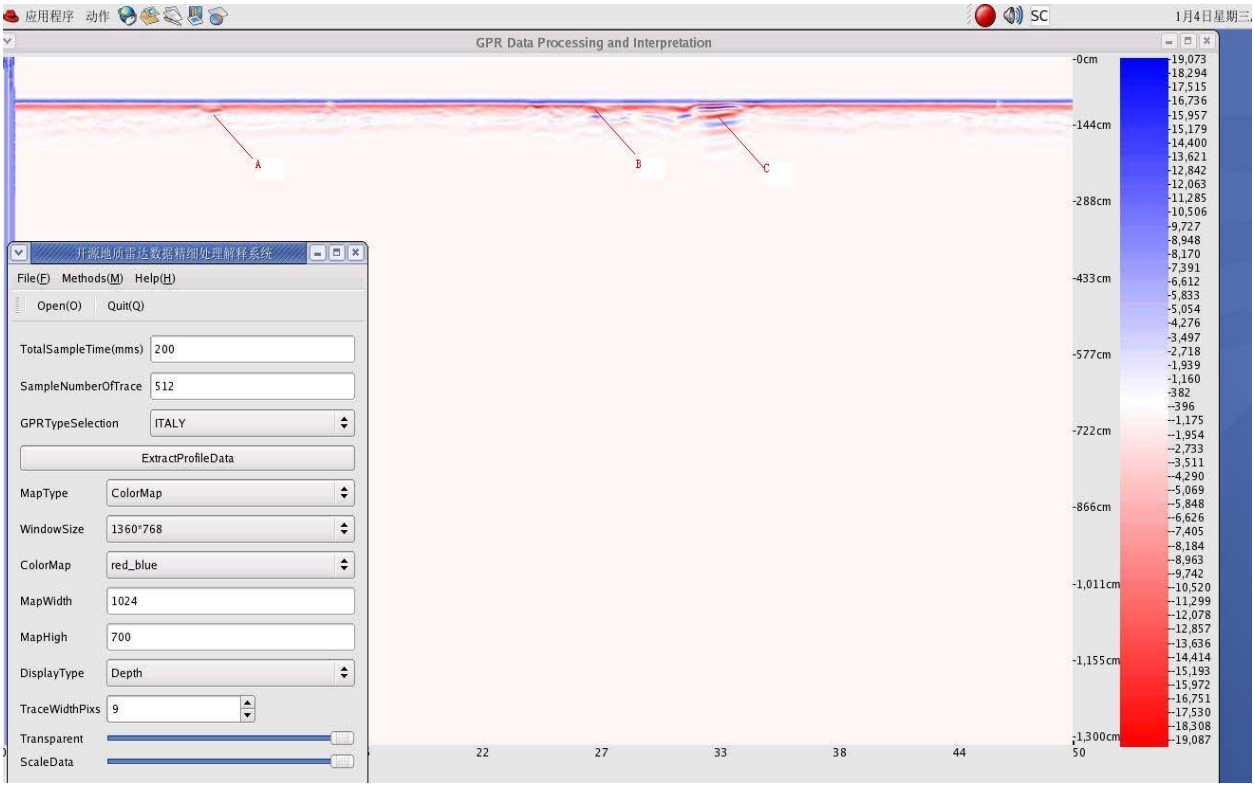

Figure 5. Imaging of GPR field data
图 5. 野外实测探地雷达成像

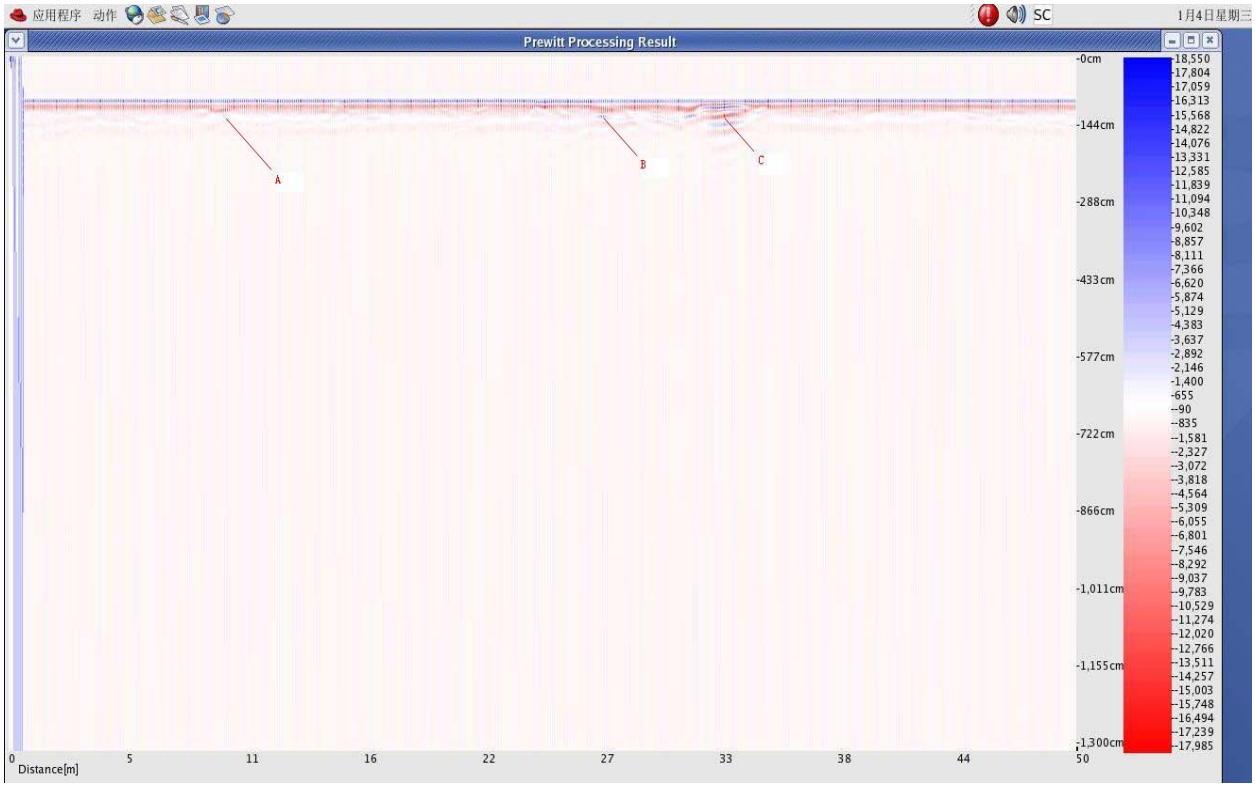

Figure 6. GPR data imaging after applying Prewitt algorithm
图 6. Prewitt 算法处理后的探地雷达成像

该工作流程及实现方法既有效解决了常规 CPU 算法的界面冻结问题，又在突出有效异常，提高目标体的识别能力方面取得了良好的应用效果。

参考文献 (References)

[1] 粟毅. 探地雷达理论与应用[M]. 北京: 科学出版社, 2006.
[2] NVIDIA. CUDA programming guide, 2012. http://www.nvidia.com/object/cuda_home.html
[3] 杨峰, 彭苏萍. 地质雷达探测原理与方法研究[M]. 北京: 科学出版社, 2010.
[4] 李荣华, 刘播. 微分方程数值解法(第四版)[M]. 北京: 高等教育出版社, 2010.
[5] 曾胜田, 刘羽等. 基于 CUDA 的 Prewitt 算子并行实现[J]. 微计算机应用, 2011, 11: 71-75.
[6] 蒋长锦. 科学计算与 C 程序集[M]. 北京: 中国水利水电出版社, 2010.
[7] J. E. Lucius, M. H. Powers. GPR data processing computer software for the PC. Reston: United States Geological Survey (USGS), 2002: 151-158.
[8] T. Y. Peng, J. X. Cao. The implementation and fast visualization application of Linux-based QT-GPU parallel architecture. American Journal of Engineering and Technology Research, 2011, 11 (12): 2063-2068.
[9] 彭土有, 叶亚平等. 意大利 RIS-2K 探地雷达数据解密及其快速可视化方法[J]. 计算机时代, 2012, 7: 17-20.
[10] 孟陆波, 李天斌等. 地质雷达超前预测不良地质体图像的智能识别[J]. 煤田地质与勘探, 2009, 2: 86-89.

