文章编号：1674－9057(2018)03－0494－07 doi:10.3969/j.issn.1674－9057.2018.03.018

一种基于CUDA的改进Canny边缘检测算法

蓝贵文，吴昊铮，张　强，赵东方

（桂林理工大学 a. 测绘地理信息学院；b. 广西空间信息与测绘重点实验室，广西 桂林　541004）

摘　要：针对标准Canny算法在滤波器和梯度幅值计算上的缺陷，使用非均值滤波算法替换了传统Canny算法中高斯滤波算法，并对梯度幅值的计算过程进行了改进，设计了一种改进的Canny边缘检测算法，并基于CUDA平台对改进算法实现了并行化。实验结果表明，改进的Canny边缘检测算法具有较强的抗噪能力，边缘检测效果较好；基于CUDA的改进Canny边缘检测并行算法相较于串行算法效率提升明显，获得了较高的加速比。

关键词：边缘检测；Canny算法；非局部均值滤波；CUDA

中图分类号：P208　　**文献标志码：**A

0　引　言

图像的边缘检测是图像处理领域中最重要的内容之一。图像的边缘是指一幅图像中灰度值发生剧烈变化的区域，主要存在于目标与目标之间、目标与背景之间、区域与区域之间，是图像中最基本的特征。图像边缘所包含的信息往往能够直接反映图像中目标的形状结构以及目标之间的拓扑关系等，这些信息对目标的特征提取、描述、识别和解译等应用方面有着极其重要的作用。1986年Canny在文献［1］中提出了评价边缘检测算法性能的三大准则：1)定位精确准则：检测出的边缘点应落在实际图像边缘的中心；2)信噪比准则：误检和漏检的概率应尽可能的小，使输出的信噪比最大；3)单边缘响应准则：尽可能减少虚假边缘的响应和多个响应的发生，并且能够尽量抑制虚假边缘的响应。同时，Canny依据三大准则推导出了一种新的边缘检测算法，即Canny边缘检测算法。Canny边缘检测算法解决了传统边缘检测算法在实际图像边缘检测中效果不理想的问题，因此被广泛应用于数字图像处理、图像分析、图像识别和机器视觉技术等领域[2]。

一直以来，图像处理领域的学者都在对Canny边缘检测算法进行优化，文献［3－6］针对传统Canny边缘检测算法的计算过程进行了不同方面的改进，并由此提出了多种改进的Canny边缘检测算法。这些改进算法在提升了Canny边缘检测结果精度的同时，也增加了Canny边缘检测算法的计算复杂度；另一方面，这些改进后的Canny算法依旧是基于CPU计算的串行算法，随着图像分辨率的提升，算法的迭代次数随之增加，导致算法的运算时间加长，效率降低明显。基于图形处理器GPU的高性能计算在图像处理领域受到越来越多的重视，不少学者将传统的Canny边缘检测算法的改进重点放在了设计其并行化的算法上，文献［7－11］都在GPU平台上实现了Canny边缘检测的高性能并行算法，并且依靠GPU强大的计算能力极大地提升了Canny算法的计算效率。尽管他们在对Canny算法并行化的过程中作出了一些改进，使得算法更加适应GPU平台的高性能计算特性，但对Canny边缘检测算法的最终边缘检测结果精度影响甚微。

收稿日期：2016－11－19

基金项目：国家高技术研究发展计划项目(SS2013AA122801)；广西区研究生创新计划项目(002504216008)

作者简介：蓝贵文（1977—），男，博士，副教授，研究方向：地理信息系统，gwlan_770728@163.com。

引文格式：蓝贵文，吴昊铮，张强，等. 一种基于CUDA的改进Canny边缘检测算法［J］. 桂林理工大学学报，2018，38(3)：494－500.

本文使用非均值滤波算法替换了传统 Canny 算法中的高斯滤波算法，并对梯度幅值计算过程进行了改进，提出了一种改进的 Canny 边缘检测算法。由于非局部均值滤波算法计算量极大，传统的串行算法无法体现其性能优势[12]，因此本文在改进的 Canny 边缘检测算法基础上，基于 CUDA 编程模型在 GPU 平台上设计并实现了对改进算法的并行化。实验结果表明，基于 CUDA 的改进 Canny 边缘检测并行算法在边缘检测结果精度方面相较于传统 Canny 边缘检测算法拥有很好的抗噪能力，且边缘检测的效果更好；在算法的运行时间方面，由于对传统 Canny 算法的改进引入了更为复杂的计算操作，使得改进算法的串行版本在完成一幅图像的边缘检测所消耗的时间相较于传统的 Canny 边缘检测算法有所增加，而基于 CUDA 的改进 Canny 边缘检测并行算法依靠 GPU 强大的计算能力在计算效率方面相较改进算法的串行版本提升明显，获得了最大 24 倍的加速比。

1　Canny 边缘检测算法原理及缺陷

传统的 Canny 边缘检测算法的流程包括如图 1 所示的几个步骤：

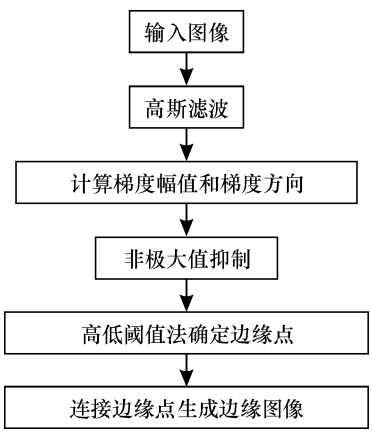


图 1　Canny 边缘检测算法流程

Fig. 1　Procedures of Canny edge detection algorithm

① 使用高斯滤波器平滑原始输入图像，用于去除噪声对最终边缘检测效果的影响。高斯滤波器可表示为

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-(x^2+y^2)/2\sigma^2}, \tag{1}$$

式中：σ 是高斯函数的标准差，用来控制滤波器的平滑程度。如果用 $F(x,y)$ 表示原始图像，那么经过高斯滤波器平滑后的图像 $I(x,y) = F(x,y) \otimes G(x,y,\sigma)$。

② 采用 2×2 邻域一阶偏导的有限差分计算平滑图像 $I(x,y)$ 在 (i,j) 处的梯度幅值 $G(i,j) = \sqrt{G_x^2(i,j) + G_y^2(i,j)}$ 和梯度方向 $\theta(i,j) = \arctan(G_x(i,j)/G_y(i,j))$，其中 $G_x(i,j)$ 和 $G_y(i,j)$ 分别为平滑图像在 (i,j) 处水平方向和垂直方向的偏导数[3]：

$$\begin{cases} G_x(i,j) = \dfrac{[I(i,j+1) - I(i,j) + I(i+1,j+1) - I(i+1,j)]}{2}, \\ G_y(i,j) = \dfrac{[I(i,j) - I(i+1,j) + I(i,j+1) - I(i+1,j+1)]}{2}。 \end{cases} \tag{2}$$

③ 根据计算出的梯度方向数组 $\theta(x,y)$ 对梯度图像数组 $G(x,y)$ 进行非极大值抑制：在梯度图像 $G(x,y)$ 中以点 (i,j) 为中心的 3×3 邻域内沿梯度方向 $\theta(i,j)$ 进行插值处理，若点 (i,j) 处的梯度幅值 $G(i,j)$ 大于梯度方向 $\theta(i,j)$ 上与其相邻的两个插值点，则将点 (i,j) 标记为边缘候选点，保持其梯度幅值不变；反之则标记为非边缘点，将其值设置为零。

④ 采用高低阈值法从经过非极大值抑制后的候选边缘图像中检测和连接出最终的边缘图像。高低阈值法首先选取高阈值 T_h 和低阈值 T_l，然后遍历图像。若某点 (i,j) 处的梯度幅值 $G(i,j)$ 大于高阈值 T_h，则认为该点一定是边缘点；若点 (i,j) 处的梯度幅值 $G(i,j)$ 小于低阈值 T_l，则该点一定不是边缘点。对于梯度赋值处于高低阈值之间的像素点，则视为疑似像素点。对于疑似像素点需要对其邻域像素进行再次判断，若其邻域像素点中存在边缘点，则该疑似边缘点同样视为边缘点；反之，则视为非边缘点。最后将所有边缘点进行连接，输出边缘图像。

对含噪声图像使用 Canny 算法进行边缘检测时，由于算法中高斯滤波器参数的不确定性和采用一阶偏导的有限差分计算梯度幅值的局限性，使得传统 Canny 算法最终的边缘提取效果较差，具体体现在：① 在高斯滤波器中，参数 σ 的取值直接影响了高斯滤波器的平滑效果：当 σ 取值较小时，高斯滤波器的边缘定位精度高，但信噪比低；当 σ 取值较大时，高斯滤波器的边缘定位低，但信噪比高。而参数 σ 的取值主要依靠人为确定，对不同类型图像的边缘检测结果受主观性影响较

大。② 计算梯度幅值和梯度方向的方法采用的是 2×2 邻域一阶偏导的有限差分，这种计算方法对噪声极其敏感，对噪声较强的图像进行边缘检测时容易对斜边上的边缘点造成误检和漏检，直接影响了最终边缘检测结果的正确性。

2 Canny 边缘检测算法改进及基于 CUDA 的并行化实现

2.1 对传统 Canny 边缘检测算法的改进

通过前文的分析，本文针对传统 Canny 边缘检测算法的改进主要集中在以下两个方面：

① 滤波器的改进。由于传统 Canny 边缘检测算法选用的高斯滤波器对图像平滑的结果受人为因素的影响较大，本文使用 Buades 等提出的非局部均值滤波算法[13]（NLM）对图像进行平滑处理。NLM 算法利用自然图像中存在大量冗余信息的特点，在对图像进行滤波的过程中不仅考虑目标像素点与其邻域像素的相关性，同时也考虑了目标像素点与整幅图像的相关性，最终目标像素的像素值由图像中所有与其结构相似的像素值加权得到。文献[14]已经论证了 NLM 算法的去噪效果明显优于其他算法，并且拥有较高的 PSNR 值。

设含噪声的原始图像为 v，对于图像中任一像素点 i 经过 NLM 算法滤波后的结果可表示为

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j), \tag{3}$$

式中：$w(i,j)$ 表示权值，其结果取决于分别以像素点 i 和像素点 j 为中心的图像块之间的相似度，并满足 $0 \leqslant w(i,j) \leqslant 1$ 且 $\sum_j w(i,j) = 1$ 两个条件。权值 $w(i,j)$ 的计算公式为

$$w(i,j) = \frac{1}{\sum_j \mathrm{e}^{-(\|v(N_i)-v(N_j)\|_{2,a}^2)/h^2}} \mathrm{e}^{-(\|v(N_i)-v(N_j)\|_{2,a}^2)/h^2}, \tag{4}$$

式中：$\|v(N_i) - v(N_j)\|_{2,a}^2$ 为加权的欧氏距离，其值越小权值越大，反之权值越小；h 为滤波系数。

② 梯度幅值和梯度向量计算的改进。文献[15]通过引入两个对角模板应用在传统 Canny 边缘检测算法梯度幅值提取的计算中，并且获得了较好的效果。本文采取相同的改进策略，在水平方向和垂直方向上使用 3×3 的模板替换传统 Canny 边缘检测算法中采用的 2×2 垂直方向模板和水平方向模板；同时在 45°和 135°方向引入两个 3×3 模板计算用于对原始图像斜边上的梯度幅值进行提取。最终，比较采用水平、垂直方向模板计算出的梯度幅值 G_1 和采用对角模板计算出的梯度幅值 G_2，选择其中较大者作为梯度幅值计算结果，存储到梯度幅值数组进入非极大值抑制步骤。

梯度幅值计算过程选取的模板为

$$S_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix};$$

$$S_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix};$$

$$S_3 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix};$$

$$S_4 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}。$$

其中：S_1、S_2 分别为水平方向和垂直方向模板，S_3、S_4 分别为45°和135°方向模板。设平滑后的图像为 $f(x, y)$，则对于任一像素点 (i, j) 的梯度幅值可表示为

$$G_1 = \sqrt{\left[\frac{1}{2}f(i,j) \otimes S_1\right]^2 + \left[\frac{1}{2}f(i,j) \otimes S_2\right]^2}; \tag{5}$$

$$G_2 = \sqrt{[f(i,j) \otimes S_3]^2 + [f(i,j) \otimes S_4]^2}; \tag{6}$$

$$G = \max\{G_1, G_2\}。 \tag{7}$$

梯度方向可表示为

$$\theta(i,j) = \arctan \frac{\frac{1}{2}f(i,j) \otimes S_2}{\frac{1}{2}f(i,j) \otimes S_1}。 \tag{8}$$

2.2 改进算法的 CUDA 实现

CUDA（compute unified device architecture）是 NVIDIA 公司为高性能并行计算设计的软硬件设备架构平台，它能够借助 NVIDIA 公司 GPU 上的大量计算单元实现使用 CPU 计算时间复杂度较高的问题。此外，CUDA 的编程语言以 C 语言为基础，并且提供了诸多 GPU 硬件调用函数，大大降低了利用 GPU 进行高性能计算的门槛，因而越来越受欢迎[16]。CUDA 的编程模型是典型的异构模型，CUDA 程序的代码分为两部分：在主机端（host）运

行的串行代码和在设备端(device)运行的并行代码。主机端由 CPU、主机内存以及相关的配套硬件构成；设备端通常是支持 CUDA 的 GPU[17]。在 CUDA 程序中，并行代码通常被封装为核函数(kernel)，核函数在主机端被调用，核函数代码则在设备端 GPU 上执行；串行代码则在主机端调用和执行。CUDA 核函数的执行模式遵从单指令多线程(SIMT)的执行模式：一旦 CUDA 程序中的核函数被调用，设备端将会创建成千上万的线程(thread)同时执行同样的核函数。CUDA 以三级的层次结构组织在设备端启用的线程，即多个线程组合在一起构成一个线程块(block)，而多个线程块则组合在一起构成一个线程格网(grid)，在调用核函数之前程序员必须对线程格网、线程块和线程的维度及数量进行配置。CUDA 还提供了多种接口函数访问 GPU 上不同的存储区域，程序员可以根据不同存储器的特性优化 CUDA 核函数对存储区域的访问效率，提升 CUDA 程序的性能。

对改进的 Canny 边缘检测算法的处理过程进行分析发现，非局部均值滤波、梯度幅值的计算、非极大值抑制这 3 个处理步骤的计算复杂性较高，并且其计算过程是以像素点为单位进行的迭代操作，不同像素点之间没有数据依赖和数据交互，具有很强的并行潜力，非常适合利用 GPU 进行高性能的并行计算，因此对于改进算法的并行化实现主要集中在上述 3 个操作步骤中；其他处理过程涉及到对图像的遍历操作，使用 GPU 进行计算没有较大的优势，因此仍交由 CPU 进行处理。在 GPU 的内存结构中，全局内存容量最大，但读取速度最慢；共享内存和纹理内存读取速度较快，它们的容量有限，共享内存只能在线程块内共享，纹理内存读取方式最为直接，易于编程读取，但创建纹理内存需要一定代价，对于计算过程中有多次重复读取时，具有性能优势。核函数设计需要充分考虑每个计算任务的流程和特点，选择合适的内存存储方式，并为每个线程分配合适的任务，尽量避免其处于空闲状态。本文针对非局部均值滤波、梯度幅值的计算、非极大值抑制的处理过程特点，设计了 3 个核函数实现并行化处理。

非局部均值滤波核函数(核函数名称：NLM_GPU())，其参数有：原始图像数组、图像的宽高信息以及滤波后的图像数组。由于非均值滤波算法中，权值的计算需要比较目标像素点所在的图像块与搜索窗口中图像块的相似性，具有较为复杂的循环操作，因此相对于其他方法，选择使用纹理存储器能够有一定程度的性能优势。首先申请纹理存储器并设置相关参数，将原始图像数组绑定至申请的纹理存储器；利用纹理拾取函数 tex2D 配合循环语句计算权值，并进行归一化运算，将修正后的像素点值存储到滤波后的图像数组中。

梯度幅值计算核函数(核函数名称：Gradient_GPU())，用于计算滤波后图像的梯度幅值和梯度方向，其参数有：滤波后的图像数组、图像的宽高信息以及梯度幅值数组和梯度方向数组。梯度幅值的计算是典型的卷积计算，适于使用共享内存模式，其算法为：首先将计算梯度幅值所需要的 4 个方向模板存储至 GPU 的常量存储器中，以便所有线程进行计算时对其进行访问；在每个线程中，读取 1 ~2 个像素到共享内存中，确保将需要处理的目标区域和其周围一定区域的像素读取到线程块的共享内存，并利用线程同步函数_syncthreads()确保所有线程在进行梯度幅值计算之前完成数据拷贝；之后在每个线程中利用梯度幅值计算公式以及梯度方向计算公式计算梯度幅值和梯度方向，并保存计算结果。

非极大值抑制核函数(核函数名称：NonMaxima_GPU())，用于实现非极大值抑制。核函数的主要参数有：梯度幅值数组、梯度方向数组、图像的宽高信息以及经过非极大值抑制后的梯度幅值数组。非极大值抑制过程也是一种涉及到邻域像素的计算，一般根据目标像素点周围 3 ×3 邻域像素点的梯度幅值和梯度方向，确定局部最大值点，适于使用共享内存模式，采取与梯度幅值核函数相同的设计策略。

程序的整体流程概述如下：(1)读取输入图像的信息，为原始图像数组、滤波后的图像数组、梯度幅值数组、梯度方向数组以及非极大值抑制后的梯度幅值数组在设备端分配显存空间，并将原始图像数组从主机端拷贝至设备端；(2)依次调用核函数 NLM_GPU()、Gradient_GPU()、NonMaxima_GPU()进行非均值滤波、计算梯度幅值和梯度方向以及非极大值抑制操作；(3)将经过非极大值抑制后的梯度幅值数组从设备端拷贝至主机端，

并在主机端完成确定边缘点、连接并生成边缘图像等剩余操作。算法流程如图 2 所示。

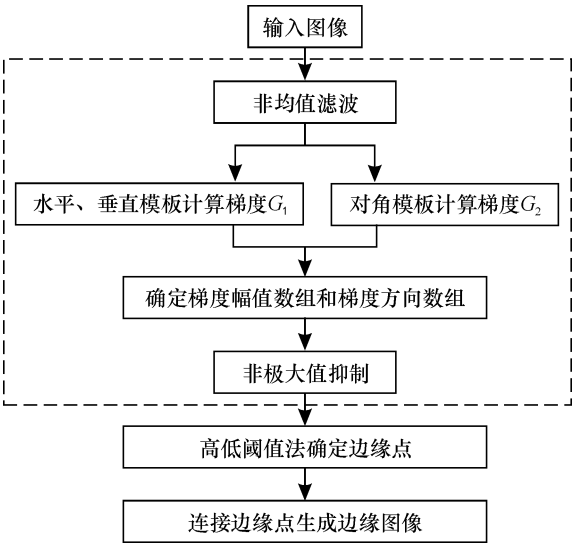


图 2 基于 CUDA 的改进 Canny 边缘检测并行算法流程

Fig. 2 Improved and parallel Canny edge detection algorithm based on CUDA

3 实验与分析

分别对基于 CUDA 的改进 Canny 边缘检测算法的边缘检测效果和加速效果进行了两次实验以验证其效果。实验平台参数：计算机的配置为 Intel Xeon CPU E5 - 2630 V2，64 GB 物理内存，Tesla K20C GPU；Windows 7 旗舰版操作系统；Microsoft Visual Studio 2010 和 CUDA Toolkit 7.0 开发平台。

3.1 改进的 Canny 边缘检测算法检测结果分析

为了验证改进算法的边缘检测效果，本文选取分辨率为 256 × 256 的添加 3% 椒盐噪声的 Lena 图像和添加标准差为 25 的高斯噪声的 Lena 图像作为实验图像，分别使用传统 Canny 边缘检测算法、改进算法对实验图像进行边缘检测，并对结果进行分析。实验图像和结果分别如图 3、图 4 所示。

图 4a—c 分别为使用传统 Canny 边缘检测算法、改进的 Canny 边缘检测串行算法和基于 CUDA 的改进 Canny 边缘检测并行算法对含 3% 椒盐噪声的 Lena 图像的边缘检测结果；图 4a′—c′分别为使用传统 Canny 边缘检测算法、改进的 Canny 边缘检测串行算法和基于CUDA的改进Canny边缘检测并行算法对含标准差为25 的高斯噪声的 Lena 图像的边缘检测结果。通过实验结果可以看出，传统



图 3 含噪声的 Lena 实验图像

Fig. 3 Tested Lena images with different noises

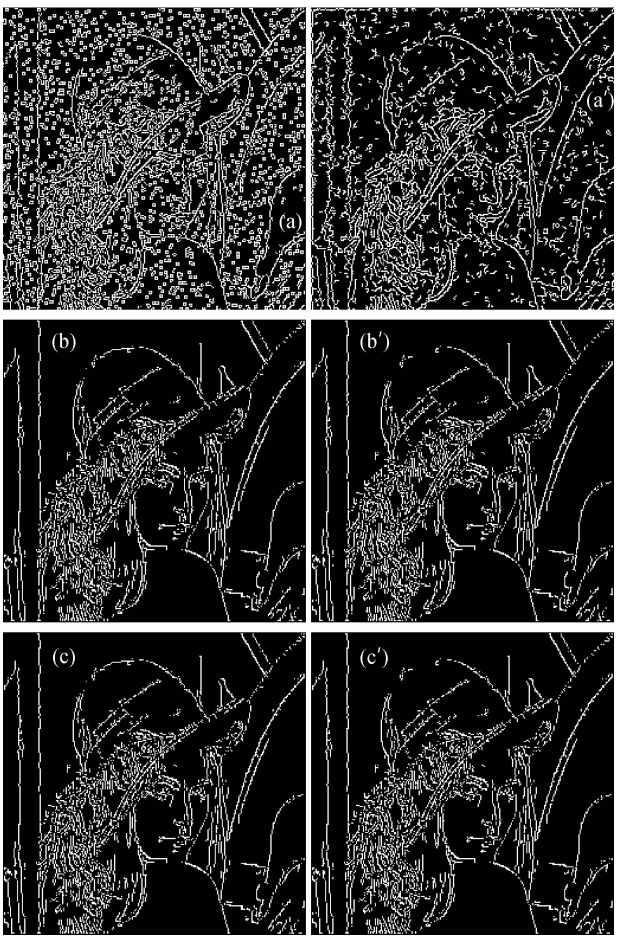


图 4 实验结果

Fig. 4 Experimental results

a—传统 Canny 边缘检测算法；b—改进 Canny 边缘检测串行算法；c—基于 CUDA 的改进 Canny 边缘检测并行算法；a′、b′、c′—对应 a、b、c 含标准差为 25 的高斯噪声 Lena 图像的边缘检测算法结果

的 Canny 边缘检测算法对噪声极其敏感，在对含噪声图像进行边缘检测时，噪声点大多都被作为边缘输出到了最终的结果中；而改进后的 Canny 边缘检测算法，具有很强的抗噪能力，图像边缘的最终检测结果较为理想。检测结果图 4b、4c 及

图 4b′、4c′是对应一致的，表明基于 CUDA 实现的并行算法正确。

3.2　改进的 Canny 边缘检测算法加速效果分析

在对改进的 Canny 边缘检测算法进行有效性验证的同时，还对基于 CUDA 的改进 Canny 边缘检测并行算法的加速效果进行分析。实验选择分辨率 128 ×128、256 ×256、512 ×512、1 024 ×1 024、2 048 ×2 048 的图像各一张，分别使用传统 Canny 边缘检测算法、改进的 Canny 边缘检测串行算法和基于 CUDA 的改进 Canny 边缘检测并行算法对上述图像进行边缘检测 20 次，统计改进 Canny 算法中非局部均值滤波计算过程的平均消耗时间，以及上述 3 种算法完成一次边缘检测所消耗的平均时间。另外，根据 Canny 边缘检测算法的串行算法和并行算法完成一次边缘检测所需要的平均时间，计算在 GPU 平台下非局部均值滤波及基于 CUDA 的改进 Canny 边缘检测并行算法获得的加速比。加速比计算公式为

$$SpeedUp = T_S / T_P, \tag{9}$$

式中：T_S、T_P 分别为使用串行算法和基于 CUDA 的并行算法完成一次计算所消耗的平均时间，*SpeedUp* 为并行算法所获得的加速比。

不同算法的实验结果（表 1）显示，尽管改进的 Canny 边缘检测算法在最终结果方面相较于传统 Canny 边缘检测算法有较大的提升，但改进算法的串行版本进行边缘检测由于非局部均值滤波的复杂性导致算法整体耗时显著增加；实验数据显示，在改进的 Canny 边缘检测算法中，非局部均值滤波的部分占用了大部分的计算时间，而基于 CUDA 平台实现的改进 Canny 检测并行算法中，非局部均值滤波获得了最大 28 倍的加速比，性能提升明显；同时，对于基于 CUDA 平台的改进 Canny 检测并行算法的整体计算时间相比其串行算法同样有明显提升，获得了最大 24 倍的加速比。在实验结果中可以注意到对于较低分辨率图像（如分辨率为 128 ×128 和 256 ×256 的图像），并行算法的优势并不明显，这主要是由 CUDA 并行编程模型自身的特点引起的。基于 CUDA 的并行程序在执行时包含数据拷贝和数据计算两个步骤，当输入数据量较小时，GPU 执行数据拷贝所消耗的时间远远超过执行数据计算所消耗的时间，并行算法在执行数据计算时的效率提升被掩盖，并行算法的优势没有得到体现；随着输入数据量的增大，数据拷贝消耗的时间基本不变，并行算法带来的效率提升逐渐明显。

4　结　论

针对传统 Canny 边缘检测算法中的滤波器和梯度幅值计算过程的不足，本文使用非局部均值滤波算法替换了标准 Canny 边缘检测算法中的高斯滤波器，并对梯度幅值的计算过程进行了改进，提出了一种改进的 Canny 边缘检测算法，并对改进算法在 CUDA 平台上实现了并行化。实验结果表明，改进的 Canny 边缘检测算法在对含噪声图像进行边缘检测时，其边缘检测结果都优于传统 Canny边缘检测算法；同时，基于CUDA的Canny边缘检测并行算法在进行大尺寸图像的边缘检测时，无论是非局部均值滤波算法的计算效率还是改进

表 1　不同算法的实验结果

Table 1　Experimental results of different algorithms

图像分辨率	算法类型	耗时/ms			加速比
		传统 Canny 算法	串行算法	基于 CUDA 的并行	
128 ×128	NLM	–	195	621	<1
	Canny 边缘检测	34	248	710	<1
256 ×256	NLM	–	874	609	1
	Canny 边缘检测	107	1 091	794	1
512 ×512	NLM	–	2 689	610	4
	Canny 边缘检测	351	3 976	810	5
1 024 ×1 024	NLM	–	11 041	809	13
	Canny 边缘检测	1 278	13 298	1 115	12
2 048 ×2 048	NLM	–	39 929	1 417	28
	Canny 边缘检测	3 921	44 750	1 822	24

Canny 边缘检测算法的计算效率，相较其各自的串行算法都获得了较高的加速比。

参考文献：

[1] Canny J. A computational approach to edge detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986 (6): 679 – 698.

[2] 孙保燕，张惊涛，程昂. 基于机器视觉的轮胎挤出部件外形尺寸在线测量方法 [J]. 桂林理工大学学报，2016，36 (4)：838 – 843.

[3] 吕哲，王福利，常玉清. 一种改进的 Canny 边缘检测算法 [J]. 东北大学学报 (自然科学版)，2007，28 (12)：1681 – 1684.

[4] 李旭，王正勇，吴晓红，等. 一种改进非极大值抑制的 Canny 边缘检测算法 [J]. 成都信息工程学院学报，2011，26 (5)：564 – 569.

[5] 许宏科，秦严严，陈会茹. 一种基于改进 Canny 的边缘检测算法 [J]. 红外技术，2014，36 (3)：210 – 214.

[6] 张帆，彭中伟，蒙水金. 基于自适应阈值的改进 Canny 边缘检测方法[J]. 计算机应用，2012，32(8):2296 – 2298.

[7] 唐斌，龙文. 基于 GPU + CPU 的 CANNY 算子快速实现 [J]. 液晶与显示，2016，31 (7)：714 – 720.

[8] Luo Y C, Duraiswami R. Canny edge detection on NVIDIA CUDA [C] //Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on. IEEE, 2008: 1 – 8.

[9] Ogawa K, Ito Y, Nakano K. Efficient Canny edge detection using a GPU [C]//Networking and Computing (ICNC), 2010 First International Conference on. IEEE, 2010: 279 – 280.

[10] Niu S X, Yang J J, Wang S, et al. Improvement and parallel implementation of Canny edge detection algorithm based on GPU [C] //ASIC (ASICON), 2011 IEEE 9th International Conference on. IEEE, 2011: 641 – 644.

[11] Cheikh T L B, Beltrame G, Nicolescu G, et al. Parallelization strategies of the Canny Edge Detector for multi-core CPUs and many-core GPUs [C] //New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International. IEEE, 2012: 49 – 52.

[12] Wang J, Guo Y W, Ying Y T, et al. Fast non-local algorithm for image denoising [C] //IEEE International Conference on Image Processing. IEEE, 2006: 1429 – 1432.

[13] Buades A, Coll B, Morel J-M. A non-local algorithm for image denoising [C] //2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVP' 05). IEEE, 2005, 2: 60 – 65.

[14] 许光宇. 非局部图像去噪方法及其应用研究 [D]. 合肥：合肥工业大学，2013.

[15] 苏连成，王东卫. 一种改进的 Canny 边缘检测算子 [J]. 燕山大学学报，2012，36 (5)：413 – 416.

[16] NVIDIA. CUDA C Programming Guide [EB/OL]. USA: NVIDIA, 2015 [2016]. http: //docs. nvidia. com/cuda/cuda-c-programming-guide/#axzz4F7Zkne2M.

[17] Tse J. Image processing with CUDA [D]. Las Vegas: University of Nevada, 2012.

Improved Canny edge detection algorithm based on CUDA

LAN Gui-wen, WU Hao-zheng, ZHANG Qiang, ZHAO Dong-fang

(a. College of Geomatics and Geoinformation; b. Guangxi Key Laboratory of Spatial Information and Geomatics, Guilin University of Technology, Guilin 541004, China)

Abstract: The Canny edge detection algorithm is one of the most widely used algorithms in the field of digital image processing. In this paper an improved Canny edge detection algorithm is proposed and implemented with CUDA based on analyzing the defects of the original Canny algorithm. Our modification to the Canny algorithm includes: (1) Utilizing Gaussian Filter with the non-local mean filtering in noise smoothing to improve the anti-noise capability, (2) Improving the calculation of the gradient magnitude to ensure the stability of edge detection. The experiment results show that the proposed method is more effective and has better anti-noise capability. Moreover, the parallel implementation of this method by using CUDA achieves quite a high speedcup, compared with its sequential counterpart.

Key words: edge detection; Canny algorithm; non-local mean filtering; CUDA