文章编号:1007-1423(2017)09-0054-04 DOI:10.3969/j.issn.1007-1423.2017.09.014

SIFT 算法 GPU 并行化研究

邓伯胜

(四川大学计算机学院,成都 610065)

摘要:

SIFT 是计算机视觉领域一个基础算法,从综合评估上看,SIFT 目前仍是最具使用价值的特征检测、特征描述算法。总结 SIFT 算法原理,针对算法耗时问题,参考多位学者研究者的并行计算解决方案,并利用 CUDA 具体实现实验,归纳出一种三级并行方案。实验表明,与 CPU 架构下 SIFT 实现方案相比,该方案大幅提升 SIFT 特征提取性能,加速比随着图片数目或特征点数目增加而增加。

关键词:

SIFT;CUDA;三级并行;加速

0 引言

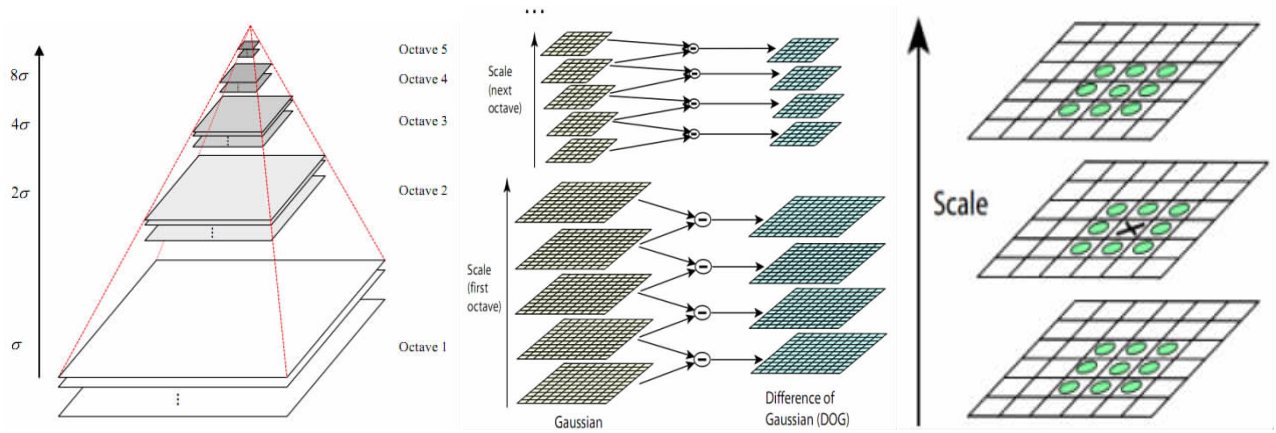
尺度不变特征变换(SIFT)是一种基于尺度空间理论的提取图像局部特征的优秀算法。SIFT 特征鲁棒性强,受旋转、缩放、平移、噪声、关照变化影响很小。但 SIFT 算法复杂度高,计算耗时,难以应用到实时性要求高的场合。随着可编程图像处理器 GPU 技术成熟,国内外很多学者针对 SIFT 计算效率问题做了很多研究,设计出很多并行加速方案。但就目前为止,这些方案的并行级别都不超过两级。例如 Sinha 等人基于 OpenGL 和 Gg 语言在 NVIDIA 7900GTX 显卡上实现 SIFT,仅仅实现了一级并行(像素级);田文等人提出一种基于 CUDA 的 SIFT 实现方案,性能达到 30-50 倍,但仅仅针对特定图片并且未考虑显卡和 CPU 端数据传输时间,并行级别只有像素级和高斯组(Octave)两级。本文考虑到 SIFT 主要应用于多张图片检测特征点,采用 CUDA 流(cudaStream)实现了首末数据传输和中间计算的并行,隐藏了多张处理图片上传图片数据和返回计算结果的数据传输时间,同时提高了计算并行度,达到了三级并行,在考虑到数据传输时间后,性能能达到 CPU 实现的 20 倍左右。

1 SIFT 算法原理

SIFT 检测图片特征点主要有两大步骤:提取特征点和描述特征点。提取特征点即计算出某个满足条件的图像像素点的坐标、尺度、方向值,描述特征点即统计出特征点及其领域点像素值变化的梯度值。包括的环节有:构建高斯金字塔,构建高斯差分金字塔,计算精选极值点,计算主方向和描述符。

1.1 建立高斯金字塔,即生成不同尺度(模糊程度)、不同大小的多组图片。

如图 1 左所示,图像高斯金字塔每一组 Octave 尺寸大小减半,一组中由下到上不同层图片模糊程度越来越大。其中第 n 组第 i 层的图片尺度(模糊程度)为:$\sigma_n(i)=\sigma_0*2^{n+\frac{i}{s}}$,(其中 σ_0 为经验值,David Lowe 在论文中推荐为 1.6,S 为差分高斯金字塔组 Octave 的实际探测极值点的层数,David Lowe 在论文中推荐为 3-5 层),为减少计算量,第 $k+1$ 张图片往往由第 k 张图片通过高斯模糊得到,即 $\sigma_n(k+1)=\sqrt{\sigma_n^2(k+1)-\sigma_n^2(k)}$,高斯卷积核的计算方法为:$G(x)=Ae^{\frac{-x^2}{2\sigma^2}}$($A$ 用于把各卷积系数累加为 1,σ 并不代表模糊后图像的尺度,而是模糊后图像尺度同模糊前的一种距离),同组第 $k+1$ 的图



左：高斯金字塔模型　　中：高斯差分金字塔　　右：极值点模型

图 1

片由第 k 张图片通过两次得到，即 $I(k+1)(x,y)=G(x)\otimes G(y)\otimes Ik(x,y)$。第 $n(n>0)$组图片的第一张图片由第上一组的第 S 张图片下采样得到。

1.2 建立高斯差分金字塔，高斯差分金字塔组数同高斯金字塔一样，每组的层数为 $S+2$(高斯金字塔每组层数为 $S+3$)，构建过程如图 1 中所 σ 示。

高斯差分金字塔第 i 组的第 k 层由高斯金字塔的第 i 组的第 k 层与第 $k+1$ 相减而得。

1.3 计算、精选极值点，如图 1 右所示，极值点为高斯差分金字塔中，极值点为其值为领域内最大或最小的点。领域包括 8+9+9=26 个点。

精选特征点包括校正特征点坐标值、尺度值，去除低对比度和边缘响应点。

高斯差分金字塔某组某层可视为一个二元离散函数 $D(x,y)$，加上尺度信息，离合出一个三元连续函数(1)，对函数(1)求导得到导函数，令导函数等于 0 得到校正后的极值点坐标尺度值(2)。

$$D(x)=D+\frac{\partial D^T}{\partial x}X+\frac{1}{2}X^T\frac{\partial^2 D^T}{\partial x^2}X,X=(x,y,\sigma)^T \quad (1)$$

$$\overline{X}=-\frac{\partial^2 D^T}{\partial x^2}^{-1}\frac{\partial D}{\partial x} \quad (2)$$

$$D(\overline{X})=D+\frac{1}{2}(\frac{\partial D^T}{\partial x})\overline{X} \quad (3)$$

去除低对比度点是除去 $D(x)$和 $D(X)$小于某个阈值的点。边缘响应点是满足条件(5)的点。

$$H=\begin{bmatrix} Dxx & Dxy \\ Dxy & Dyy \end{bmatrix} \quad (4)$$

$$\frac{Tr(H)}{|H|}<\frac{(r+1)^2}{r} \quad (5)$$

式(5)中 H 矩阵的迹 $Tr(H)=Dxx+Dyy$，行列式$|H|=Dxx*Dyy-Dxy*Dxy$.r 是 H 的较大特征值与较小特征值的比值，David Lowe 在论文中推荐为 10。

1.4 计算主方向和描述符。主方向和描述符均属于特征点及其领域点的灰度变化的梯度值的一个统计量，特征点及其领域点的梯度值和方向由(6)计算。

$$m(x,y)=\sqrt{(I(x+1,y)-I(x-1,y))^2+(I(x,y+1)-I(x,y-1))^2}$$

$$\theta(x,y)=\arctan\frac{I(x,y+1)-I(x,y-1)}{I(x+1,y)-I(x-1,y)} \quad (6)$$

区别在于，主方向的统计划分区间仅为方向，而描述符的统计划分区间除了方向还有子领域的纵横坐标；主方向的方向区间一般为 36 个(每区间 10 度)，而描述符的为 8 个(每区间 45 度)，描述符的方向需要进行旋转，旋转大小为主方向大小；主方向为最大梯度值对应的方向，描述符向量为统计的各梯度值。

2　基于 CUDA 的 SIFT 并行性分析设计

显卡端业务数据用 CUDA 数组 cudaArrayt 存储。由于在 cudaArrayt 上结合 cudaTextureObjectt 和 cudaSurfaceObjectt 有如下优势：a.数据读写性能远远高于直接把数据存储在显卡的全局内存；b.能在硬件上自动进行双线性插值；c.能在硬件上自动把超出边界的下标值校正为边界值；d.方便坐标值计算。

(1)构建高斯金字塔。计算卷积核及其半径由于计算量少在 CPU 段实现，高斯模糊分行列两次进行，每个线程块负责一行或一列的像素点模糊，线程和线程块均组织为线性结构。在图像进行降采样(图像长宽都缩小一半)时，每个线程负责算得一个降采样后的像素点，每个线程块负责处理连续的像素点，线程和线程块均组织为线性结构。

(2)构建高斯差分金字塔。每个线程负责一个差分点的计算，线程块组织成二维结构，一个线程块处理一组连续的点，第二个维度代表一个高斯差分金字塔一组中的一层，一个高斯差分组将一次得到。

(3)寻找极值点。线程组织结构与构建高斯差分金字塔类似，每个线程探测一个差分点是否为极值点，如果是极值点则采用原子操作将全局计数器累加。

(4)宏观流程。前期工作均是按照高斯组 Octave 进行，每组将产生大量极值点。按照 David Lowe 算法，前后期都分别对各组金字塔建立，极值点精选，计算主方向和描述符。这种方式对于串行执行有一些好处，在处理后一组时就可以释放前一组的大部分数据。但在并行算法中，这样并行度会大大降低，特别是在较后组图片大小较小时，每组检测出的极值点不多。最重要的是，每次计算极值点之后，精选极值点之后，计算主方向之后，都必须将数据从设备端(显存)将数据 copy 到 CPU 端，因为后期的 CUDE 核函数线程配置要用到前期的探测到的特征数(而 CPU 和 GPU 之间数据传输极度影响性能)。这样，主机和设备端的数据 I/O 时间复杂度为 O(log2(min(width,height)))。因而，本文重新设计计算流程，精选极值点时，前期的全部组的计算都应结束，这样一则增加后期运行的并行度，二则把 IO 时间复杂度降为 O(0)。由于数据在显存和主存间的传输带宽远远小于显存和显存之间的数据传输带宽，这将大大减少数据传输时间。

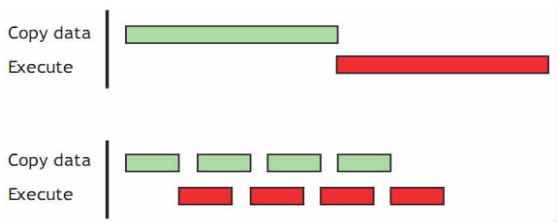


图 2 串并行方式比较

(5)执行和数据 I/O 的并行化。每张图片利用显卡进行 SIFT 特征检测时，必然前期要把数据上传到设备端，后期处理结束要将结果传输回 CPU 端。由于 SIFT 往往用于多张图片的特征点检测，因而利用 cudaStream 实现数据 I/O 和执行并行。如图 4 所示，使用此机制后，程序的运行时间由(执行时间+I/O 时间)降为(执行时间+I/O 时间/流的数量)，数据传输时间得到了有效的隐藏，在实际编程时，要向设备端上传数据前或从设备端返回数据后，各个流均进行同步，具体而言，包括在上传灰度值到显存之前，返回极值点个数到主机端之后，返回精选后极值点个数之后，返回计算主方向特征点个数之后，以及返回最终结果之后(开始使用检测到的 SIFT 特征点之前)。

(6)精选极值点。一个线程处理一个极值点，线程和线程块均设计为线性结构。计算过程中的向量矩阵临时变量使用共享内存(shared)。

(7)计算主方向描述符。线程组织结构同上一步。计算过程中重量级变量均使用共享内存(shared)，如描述符的 128 维向量，特征点方向的 36 个统计值等。

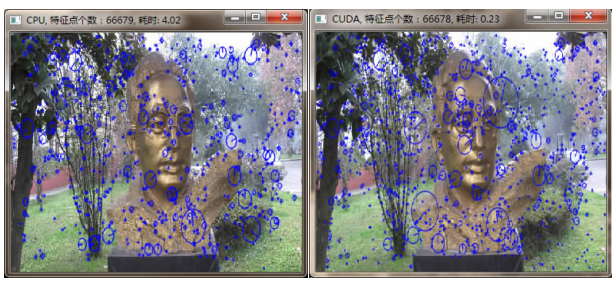


图 3 左:CPU 版本 SIFT；右:GPU 版本
实验图像分辨率:4416×2480 显示的特征点数占 0.88%

(8)三级并行说明。在 3.1,3.2 和 3.3 中每个线程处理一个像素点,3.6 和 3.7 中每个线程处理一个极值点或特征点,这属于第一级并行,这种并行是 GPU 加速方案的共性,限于 GPU 开始发展的硬件限制,最初的 GPU 加速方案大都仅做到了这一级并行度;3.4 将需多次进行的针对不同数据的极值点精选，特征点主方向计算和描述符计算缩减为一次针对一张图片的所有数据的,这不但减少了数据传输次数,也提高了计算并行度，这是本方案的二级并行;3.5 实现了对多张图片同时检测特征点的并行,不但可以提高计算的并行度,也能让数据传输和计算同时进行,隐藏了数据传输时间,

是第三级并行。

3 实验结果分析

如图 2 所示,CPU 版本与 GPU 版本的运行结果是一致的(特征点相差 1 是由于 GPU 的 float 类型精度比 CPU 版本稍低引起的，特征点显示不同是由于都只显示了很少部分),差别在于运行速率。

图表 1:纵轴表示 GPU 并行加速比,横轴表示单张图片的特征点数，单张图片特征点数量变化直接通过缩小图片分辨率得到。

从图表 1 可知:当图片分辨率小,特征点数目较少时,单张图片加速比很小,这时同时处理多张图片加速比会提高，但图片量增加到一定程度之后加速比保持不变,当图片分辨率较大时,同时处理多张图片和单张图片加速比几乎一致,主要是受 GPU 内存和计算资源限制。下一步的工作是提高显存使用率以及破除实现中的大量原子操作,以进一步提高加速比。

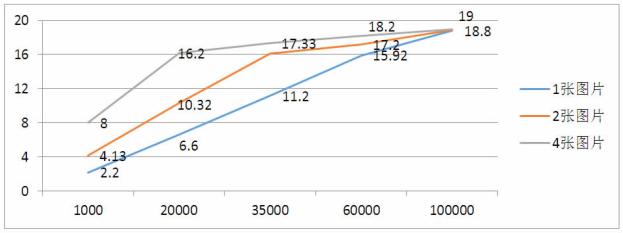


图5

参考文献:

[1]David G.Lowe. Distinctive Image Features from Scale-Invariant Keypoints[C]. Canada: Computer Science Department University of British Columbia Vancouver. 2004.pp.1-28

[2]Nabeel Khan, Brendan McCane, Steven Mills. Better than SIFT[C]. New Zealand: Machine Vision and Applications,2015.

[3]Zddhub. SIFT 算法详解. Published Blog: http://blog.csdn.net/zddblog/article/details/7521424,2012.

[4]Marten Bjorkman. A CUDA Implementation of SIFT for NVidia GPUs. https://github.com/Celebrandil/CudaSIFT.

[5]占正锋.基于 GPU 的 SIFT 立体匹配算法研究[D].黑龙江:哈尔滨工业大学机电工程学院,2014.7:23-42.

[6]NVIDIA. CUDA C Best Practice Guide, March 2015. pp.31-67: http://developer.nvidia.com/cuda-downloads.

作者简介:

邓伯胜(1989-),男,四川资中人,在读硕士研究生,研究方向为计算机图形学、数字图像处理、并行计算

收稿日期:2017-02-06　　修稿日期:2017-03-10

An Improved Algorithm of the Screen Space Ambient Occlusion

DENG Bo-sheng

(College of Computer Science, Sichuan University,Chengdu 610065)

Abstract:

SIFT is a fundamental algorithm in the field of Computer Vision. From the comprehensive evaluation, so far, SIFT is still the most valuable feature detection and description algorithm. Presents a three-level parallel solution to the time-consuming problem in SIFT feature extracting, refers to others parallel computation methods proposed by many scholars and researchers, with the implement and experiment on the platform of CUDA, summarizes its algorithm theory and experiment experience. Experimental results show that, compared with the CPU-based implement, this solution improves its performance radically, and the speedup ratio increases together with the addition of images or SIFT feature points.

Keywords:

SIFT; CUDA; Three-Level Parallel; Speedup