CPU-GPU 协同计算加速 ASIFT 算法

何婷婷[1,2] 芮建武[1] 温 腊[1,2]
(中国科学院软件研究所 北京 100190)[1] (中国科学院大学 北京 100190)[2]

摘 要 ASIFT(Affine-SIFT)是一种具有仿射不变性、尺度不变性的特征提取算法,其被用于图像匹配中,具有较好的匹配效果,但因计算复杂度高而难以运用到实时处理中。在分析ASIFT算法运行耗时分布的基础上,先对SIFT算法进行了GPU优化,通过使用共享内存、合并访存,提高了数据访问效率。之后对ASIFT计算中的其它部分进行GPU优化,形成GASIFT。整个GASIFT计算过程中使用显存池来减少对显存的申请和释放。最后分别在CPU/GPU协同工作的两种方式上进行了尝试。实验表明,CPU负责逻辑计算、GPU负责并行计算的模式最适合于GASIFT计算,在该模式下GASIFT有很好的加速效果,尤其针对大、中图片。对于2048*1536的大图片,GASIFT与标准ASIFT相比加速比可达16倍,与OpenMP优化过的ASIFT相比加速比可达7倍,极大地提高了ASIFT在实时计算中应用的可能性。

关键词 特征提取,ASIFT,SIFT,CPU/GPU协同工作
中图法分类号 TP311 **文献标识码** A

Accelerating ASIFT Based on CPU/GPU Synergetic Parallel Computing

HE Ting-ting[1,2] RUI Jian-wu[1] WEN La[1,2]
(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(University of Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract ASIFT(affine-SIFT) is a fully affine invariant, and scale invariant image local feature extraction algorithm. It has a good result in image matching. But because of its high computational complexity, it cannot be applied to real-time processing. Thus GPU is used to accelerate ASIFT. Based on the analysis of running time of ASIFT, firstly SIFT was adapted to GPU, and then the other parts of ASIFT. Memory pool was used in GASIFT to avoid frequently allocating and deleting memory during the runtime. Different ways of CPU/GPU synergetic parallel computing were studied to make GASIFT more efficient. Experiments show that the model in which CPU takes the logical calculation work and GPU makes parallel computing is the most suitable way. Based on this model, GASIFT has a good speed-up ratio over other methods. That's 16 times compared with traditional ASIFT, and 7 times compared with OpenMP optimized ASIFT.

Keywords Image feature extraction, ASIFT, SIFT, CPU/GPU synergetic parallel

1 引言

特征提取是图像处理中的一个重要部分,提取出来的特征被广泛用于图像匹配、遥感影像配准等多个领域。一个好的特征提取算法能够在很大程度上提高这些匹配的准确性。

1999年,lowe等人提出了具有尺度不变性的特征提取方法SIFT(Scale-Invariant Feature Transform)算法,并在2004年被加以完善[1]。这种算法基于图像的局部特征,对图像间存在的缩放、旋转、平移等几何变形及噪声、光照变化等图像变化因素有非常好的稳定性,因此得到广泛应用。但是SIFT算法计算时多次使用高斯卷积,复杂度较高,耗时长,不利于实时的计算。因而也有很多研究者对这种算法进行改进。2004年,提出的PCA-SIFT算法使用主元分析代替SIFT中的直方图法计算特征描述[2]。2006年,Herbet Bay等人提出了SURF算法,采用简化近似的思想对SIFT进行优化[3]。2007年S. Heymann等人实现了SIFTGPU[4],充分利用了GPU的纹理内存的机制进行加速,加速比可以达到8倍。2010年国防科技大学的王瑞等人,对基于GPU的SIFT特征提取算法进行了研究[5],研究中对除了特征描述计算之外的部分进行了GPU加速,结果表明对于大图像,其可以达到15

到稿日期:2013-07-08 返修日期:2013-10-20 本文受云计算操作系统及关键基础组件的研究与开发:面向云计算的大数据集并行处理平台研究与开发(KGCX2-YW-174),国家科技支撑计划项目:新型网络终端操作系统社区版本研究与开发,应用程序库汇总 meegobox(2011BAH14B02),2012年度"核高基"重大专项:开源操作系统内核分析和安全性评估(2012ZX01039-002),新闻出版重大科技工程项目-中华字库工程-第23包:应用平台研发(GAPP-ZDKJ-ZK/23)资助。
何婷婷(1989-),女,硕士生,主要研究方向为并行计算与分布式计算,E-mail:tingting2@nfs.iscas.ac.cn;芮建武(1972-),男,博士,高级工程师,主要研究方向为操作系统、并行分布式计算、中文信息处理;温 腊(1986-),女,硕士生,主要研究方向为并行分布式计算。

倍的加速比。同年，Herwig Lejsek 等人也对 SIFT 进行了 GPU 优化，对计算特征描述的部分进行了改进，使之可以使用 GPU 线程并行计算[6]。SIFT 虽然得到了广泛使用，但是它只是一个具有尺度不变性的算法，不具备仿射不变性。2009 年，Jean-Michel Morel 等人提出了具有完全仿射不变性的 ASIFT 算法，加强了对不同拍摄角度图片的识别[7]。

ASIFT 算法由于同时具有仿射不变性、尺度不变性的优势，被用于图像匹配中，得到了较好的效果。2011 年 Jonathon S. Hare 等人将其运用到图像检索中，得到了比其他特征更高的正确率[8]。2012 年，Chu Bin 等人将其应用到了自动全景拼接中，获得了较好效果[9]。

ASIFT 虽然有较高的识别能力，但是计算时间太长，复杂度是 SIFT 算法的数倍。根据 Chunxia Yin 等人的实验[10]，在几种局部特征提取算法中，ASIFT 对旋转、尺度变换、模糊变化有很好的鲁棒性，但它的时间消耗也是最大的，这极大地限制了它在实时计算中的应用。目前针对 ASIFT 的优化，主要是使用 OpenMP，利用多核 CPU 进行多线程并行计算从而达到加速目的，另外就是改进其中采用的特征提取算法，使用更快的算法替代 ASIFT 中使用的 SIFT。2012 年，Yanwei Pang 等人使用 SURF 算法加速 ASIFT[11]，获得了 3 倍左右的加速比。虽然这两种方法都取得了一定的成果，但是加速比不高，ASIFT 仍然难以运用到实时计算中。因此，本文尝试使用另一种方法对 ASIFT 进行加速。

GPGPU(General Purpose GPU)是一种利用图形处理器辅助计算的方式。由于图形处理器专注于并行计算，拥有更多的计算单元和 cache，它的浮点计算能力是 CPU 的几十倍甚至是几百倍，因此使用 GPU 加速图像处理技术也成为一个研究热点。根据上述内容，本文考虑使用 GPU 对 ASIFT 算法进行优化。

在如何设计 CPU/GPU 协同工作的模式这个问题上，根据国防科技大学卢风顺等人的研究[12]，CPU/GPU 协同工作包括两种层次，一种是 CPU 端承担逻辑计算，GPU 端承担并行计算任务，第二种是 CPU 端也承担一部分的计算。本文在分析 ASIFT 算法的基础上，先在第一层次上实现 CPU/GPU 协同加速 ASIFT，再尝试在第二层次上实现并行加速。

本文第 2 节简单介绍 ASIFT 算法的计算步骤，并分析 ASIFT 算法运行耗时分布；第 3 节是对 ASIFT 算法 GPU 加速的设计实现；第 4 节是性能测试；最后是结论。

2 ASIFT 算法性能分析

2.1 ASIFT 算法

2009 年 Jean-Michel Morel 在尺度不变性算法 SIFT 的基础上提出了完全仿射不变性 ASIFT 算法。ASIFT 对图片进行视角变化采样，并对采样结果分别使用 SIFT 提取特征，这样可以获取不同视角的图像特征进行比对，从而满足仿射不变性。

ASIFT 在变换中加入了摄像机的经度和纬度这两个参数来模仿摄像机在不同位置的视角。如图 1 所示，φ 表示摄像机的旋转角，ϕ 和 θ 表示相机的视角，分别称为经度和维度。

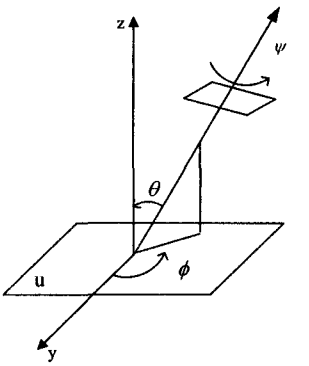


图 1 摄像机拍摄角度

任何一个正定矩阵 $A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 可分解为：

$$A = H_\lambda R_1(\varphi) T_t R_2(\phi)$$

$$= \lambda \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \quad (1)$$

其中，$\lambda>0$，$\lambda \times t$ 是 A 的行列式，R 代表旋转，T 为倾斜度，t 称为倾斜参数，$\theta=\arccos\frac{1}{t}$。

对上述的倾斜参数 t 和经度角进行采样，使相关倾斜遵循几何级数，$1, a, a^2, \cdots, a^n$，一般 $a=\sqrt{2}$，$n=5$。ϕ 采样值与倾斜程度相关，基本如下：$\phi=0, \frac{b}{t}, \frac{2b}{t}, \frac{3b}{t}, \cdots, \frac{mb}{t}$，其中 $b=72°$，m 的值为使 $\frac{mb}{t}<\pi$ 的最大值。

根据以上的 t 和 ϕ 对图像进行视角变化，见式(2)，I 为输入图像，I' 为输出图像。

$$I'(\phi,t)=\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} I \quad (2)$$

ASIFT 算法尽可能地模拟了摄像机光轴取向变化引起的所有扭曲变形，再使用 SIFT 算法计算关键点，这样就获得了图像在各种变换下的特征点。经验证，ASIFT 算法是一种具有完全仿射不变性的特征提取算法。

由于 ASIFT 算法在对图像进行特征提取时使用到了 SIFT，因此 SIFT 也是一个很重要的部分。下一节具体介绍 SIFT 算法。

2.2 SIFT 算法

SIFT 算法通过构建高斯金字塔获得多尺度空间，在这些多尺度空间中提取关键点，并产生关键点的描述信息，一般包含以下步骤[13]：

(1)构建高斯金子塔及高斯差分金字塔。通过对输入图像做 σ 不同的高斯连续滤波和降采样，一幅图像可以产生几组 octave 图像，每组中又包含若干 interval 图像。

$$G(x_i, y_i, \sigma)=\frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x-x_i)^2+(y-y_i)^2}{2\sigma^2}\right) \quad (3)$$

$$L(x,y,\sigma)=G(x,y,\sigma) * I(x,y) \quad (4)$$

获得高斯金字塔后，通过相邻层相减，获得高斯差分金字塔(DoG)，描绘出目标的轮廓。

(2)检测极值点并精确定位。在高斯差分金字塔的一个尺度上，每个点与和它同尺度的 8 个相邻点以及上下相邻尺

度对应的9×2个点，共26个点进行比较，以确保在尺度空间和2维图像空间都检测到极值点。为了避免噪声和边缘敏感，得到的极值点需要进行一次三维二次函数拟合以确定关键点，同时利用Hessian矩阵，除去边缘响应。

(3)关键点方向分配。计算每个像素点的梯度幅值和梯度方向：

$$grad(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \tag{5}$$

$$ori(x,y)=\tan^{-1}\left[\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right] \tag{6}$$

(4)利用直方图统计关键点周围的像素的梯度方向，直方图的峰值则表示该关键点的主方向，为每个关键点指定方向参数，使其具有旋转不变性。继续检测直方图，计算关键点辅方向。一般辅方向采用主方向的80%作为阈值。

(5)关键点描述。将坐标轴旋转为特征点方向，以确保旋转不变性。以特征点为中心，取16×16像素大小的窗口。将这个窗口分成4×4部分，每部分包括4×4的像素。计算这4×4像素块内在8个方向上的直方图，由此计算得到4×4×8=128维的特征描述向量。

2.3 ASIFT算法运行时间分析

根据ASIFT算法的原理，整个计算过程可分为对图像做采样变换和计算SIFT特征值两个部分。采样变换中又包括了模仿相机经度和纬度两个部分，即对图像做旋转变换、高斯平滑之后，再做仿射变换。使用gprof对CPU端的ASIFT程序进行分析，得到图2所示的运行时间比，从图中可以看到，在整个ASIFT中，占用时间最长的是使用SIFT计算图像特征的部分，占有55%的计算时间，倾斜度变换占了20%，旋转占有17%。

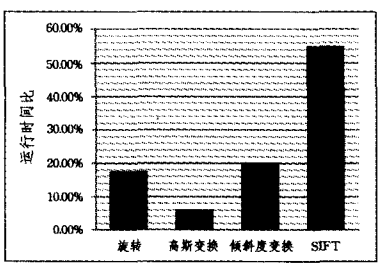


图2 ASIFT算法运行分析

3 GASIFT并行加速设计

本节尝试使用GPU对ASIFT进行并行加速，设计并实现了GASIFT计算模型。根据2.3节对ASIFT运行时间的分析，整个ASIFT算法中占用时间最长的是SIFT部分，为了实现GPU优化的ASIFT算法，本文先研究了对SIFT算法的GPU优化。

3.1 GPU优化SIFT部分设计

目前已有很多对SIFT的GPU优化版本，但它们都没有用到显存池的概念，这就导致在程序中会有大量的cudaMalloc和cudaFree释放显存空间。ASIFT会频繁调用SIFT，这样就会浪费掉许多时间在显存空间的申请和释放上。为了避免这一情况，本文对SIFT算法的优化使用了显存池的概念。在一开始申请一片显存，每次计算一个octave中的特征点，计算中所用到的显存空间都从显存池中获取。

SIFT的运算过程见2.2节。将图像灰度化处理后传输到GPU端进行大部分计算。第一步是构建高斯金字塔。接着是第二部，检测极值点。由于第三步计算关键点主方向和关键点描述都需要用到关键点附近某一范围内的点的梯度幅值和梯度方向，而这个计算点的梯度幅值和梯度方向可以与第二步检测极值点并行计算，因此可以使用CUDA流的机制来并发管理检测极值点与方向梯度的计算，并将各点的方向梯度保存在纹理内存中，方便之后使用。第二步检测到极值点之后，确定极值点的位置，用海森矩阵除去不稳定的边缘响应点，将最后的结果保存在位图中，传回到CPU端串行计算特征点个数。求得特征点个数后动态申请设备端内存，并且传回GPU端。

第四步，GPU端根据之前计算的特征点位图和各点方向梯度图，统计关键点周围像素的梯度方向直方图。每个线程计算一个关键点的主方向，将关键点周围像素的梯度方向放进36个柱中，每10°为一个柱。根据得到的方向直方图，进行一次平滑。以直方图中值最大的作为特征点的主方向。

第五步是计算关键点描述符，这是SIFT算法中最耗时的一部分。按照SIFT算法，需要计算关键点周围4 * 4个区域的方向直方图，每个区域又包含若干像素(根据图像所在尺度决定)。如果这一部分并行粒度太粗，例如一个线程计算一个关键点的特征，则会导致大量的访存冲突和访存延迟，并且能够并发的线程数少，GPU的资源被浪费。因此需要更细粒度的划分。在Lowe的SIFT算法中，先根据式(7)确定采样图像区域半径R，之后在原图中采样一个半径范围内的点(x,y)，根据式(8)旋转角度，获得它在以关键点主方向为坐标系中的位置$P(x',y')$，然后根据新的坐标，计算4 * 4个区域的8方向梯度直方图。

$$radius=\frac{3\sigma\times\sqrt{2}\times(d+1)+1}{2} \tag{7}$$

$$\begin{vmatrix}x'\\y'\end{vmatrix}=\begin{bmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{bmatrix}\begin{vmatrix}x\\y\end{vmatrix} \tag{8}$$

这样的设计如图3中的(a)关键点描述采样，使用GPU优化时，如果一个线程执行B区域所有点的计算，则在这个线程中可能要访问区域A'、B'、C'、D'区域的梯度直方图。这就要求在shared momery中存储每个区域的直方图，这样会引入3个问题：1)访存延迟；2)访存冲突；3)访问共享内存时还需要解决原子操作的问题，这些都会产生许多附加的时间开销，因此并不适合于并行计算。

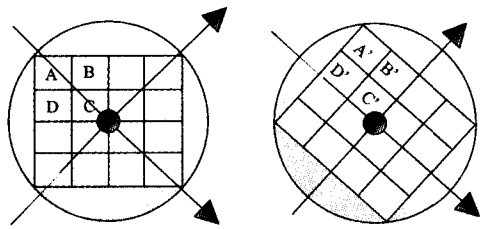
根据这样的情况，本文对关键点周围的采样进行了改进，图3(b)为改进后采样。

先找到A'、B'、C'、D'区域的中心点，例如(x_a',y_a')，根据式(9)，计算出该点在原图中的位置(x_a,y_a)。

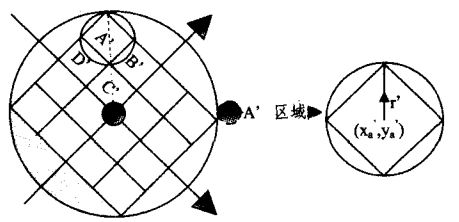
$$\begin{vmatrix}x\\y\end{vmatrix}=\begin{bmatrix}\cos\phi & \sin\phi\\ -\sin\phi & \cos\phi\end{bmatrix}\begin{vmatrix}x'\\y'\end{vmatrix} \tag{9}$$

根据区域中心点，采样$r'=R/4$范围内的点乘以高斯权重来计算该区域的梯度直方图。这样可以确保一个GPU线程中计算的点一定是属于某一区域中的点，并且距离中心点越远，对直方图影响越小。每个线程在自己的local memory中保存一个大小为8的数组，计算某区域的8方向直方图，最后再把结果写入到全局内存中，这种做法减少了访问共享内存或者全局内存带来的访存延迟，一定程度减少了访存的冲

突，并且不需要解决原子操作的问题。为了充分利用 GPU 的计算单元，可以采用更加细粒度的划分，把一个区域分成 4 部分，让 4 个线程并行计算这个区域中的 8 方向梯度，这样可增加线程数，从而增加 GPU 中流处理器的占用率。



（a）关键点描述采样



（b）改进后采样方式

图 3　SIFT 关键点描述采样

3.2　GASIFT 协同计算模式

第 1 节提到了 CPU/GPU 协同工作的两种层次，这两个层次中的主要不同在于 CPU 是只负责逻辑计算还是参与部分并行计算。由于 ASIFT 算法是对图像不同视角进行采样、计算特征点，不同视角之间的特征提取互不影响，因此它可以对计算的每一步进行细粒度的并行，CPU 只负责逻辑计算，GPU 负责并行计算；也可以实现同时计算多个视角的特征，这样粗粒度的并行，例如使用 OpenMP 对 ASIFT 的加速，让多个 CPU 核同时对图像进行视角变换来计算特征点。所以对 ASIFT 算法在 CPU/GPU 协同工作的两种层次上并行加速都是可行的。

由于 CPU 负责逻辑计算、GPU 负责并行计算，是采用 GPU 加速的一般方式，因此，本文先实现了 CPU 负责逻辑计算这一种协同模式。

3.2.1　CPU 负责逻辑计算

GASIFT 在 CPU 负责逻辑计算、GPU 负责并行计算的模式下的具体实现过程如图 4 所示。

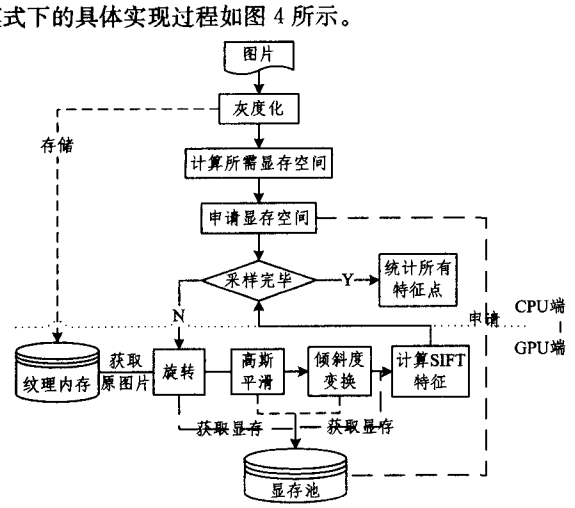


图 4　GASIFT 算法流程图

根据 3.1 节设计的使用显存池的 SIFT 算法，ASIFT 获取图像后，根据图像的大小申请一片全局内存作为显存池，之后的图像都保存在这里，包括 SIFT 计算时需要的高斯金字塔、方向、梯度等。由于原始图像可能会经常用到，将原始图像放入纹理内存中，可以加速读取。

获得输入图像后，进行灰度化，除去颜色信息干扰。将灰度化的图像保存在纹理内存中。从显存池中申请一块显存，从纹理内存拷贝一个图像副本到显存中，第一步，计算采样的倾斜度 t 和纬度 φ。t、φ 的采样根据 2.1 节描述的过程，$\phi=0$，$\frac{b}{t},\frac{2b}{t},\frac{3b}{t},\cdots,\frac{mb}{t}$，$t$ 取 $1,a,a^2,\cdots,a^n$，$a=\sqrt{2}$，$b=72°$。第二步，在 GPU 端对图像进行旋转计算，模仿相机在纬度 ϕ 上的视角。对图像旋转计算见式(10)。

$$I'(\phi,t)=\begin{bmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{bmatrix}I \tag{10}$$

GPU 端每个线程计算一个点的位置变换。对旋转后的图像进行一次高斯平滑，高斯平滑所需的临时显存空间从显存池中获取。第三步，继续进行倾斜度 t 的变换。根据采样的倾斜度 t，在 CPU 端计算出图像变换后的长、宽，在 GPU 端采用三次样条插值将图像变换到指定的长宽中。同样，每个 GPU 线程计算一个像素点。这样就得到了相机在倾斜度 t、纬度 ϕ 处的视角，把这个结果保存在显存池中。第四步，提取特征点，计算特征点描述符。使用 3.1 节实现的 GPU 优化的 SIFT 对显存池中变换过的图像进行特征提取，所需要的辅助显存空间都从显存池中获取。之后检查采样是否完毕，如果没有，则将计算结果保存至 CPU 端，并从纹理中获取原始图像，重复第一步采样倾斜度 t 和纬度 ϕ，获得下一个采样的 t 和 ϕ，继续对图片做变换，计算 SIFT 特征值；如果采样已经结束，则统计所有计算出的特征点，产生最后结果。

3.2.2　CPU 参与并行计算

由于 GPU 只能由一个线程控制，在目前多 CPU 核条件下，3.2.1 节在设计 GPU 对 ASIFT 的优化过程中，GPU 进行计算时，CPU 核是空闲的。因此考虑将这些 CPU 核也利用起来，进行并行计算，实现 CPU/GPU 协同工作的第二个层次。

设计思路：使用多线程进行并行计算，每个线程都选取一个倾斜度 t 和纬度 φ 的采样，计算这个 t 和 φ 下图像的视角变换，并对变换后的图像计算特征点，最后把结果合并在一起。所有线程中，有一个线程负责控制 GPU 端任务，其它线程则并行计算图像特征。

4　实验与分析

本文的实验平台采用 Ubuntu 10.04 操作系统，CPU 为 Intel(R) Core(TM) i5，主频为 3.33GHz，4 核，GPU 为 GeForce GTX 480，显存为 2G。

由于本文分别在 CPU/GPU 协同工作的两个层次上实现了对 ASIFT 的加速，因此关于这部分的性能测试也在这两个层次上展开。

4.1　CPU 负责逻辑计算模式

对于 CPU 负责逻辑计算、GPU 负责并行计算的模式，本文采用了不同像素大小的 5 组图像进行测试。将其与标准 ASIFT[14]，以及 OpenMP 加速过的 ASIFT 进行对比，测试结

果如表1所列。

表1 ASIFT运行时间测试对比

图像大小	标准ASIFT算法		ASIFT OpenMP	GASIFT(CPU负责逻辑计算)	
	特征点数	耗时(s)	耗时(s)	特征点数	耗时(s)
300*211	1373	1.8	0.73	1464	0.76
640*480	6598	8.78	3.52	6289	1.61
800*640	10731	14.95	5.76	10033	2.00
1600*1200	33526	55.93	20.76	32271	4.26
2048*1536	49221	97.26	35.88	47800	5.97

因为使用OpenMP优化后的程序提取特征点与标准的一样,所以表中并没有列出这一项。ASIFT算法对图像进行了仿射变换,模仿了相机在不同角度观察的图像,因此总的特征点数比SIFT算法多,并且运算时间也比SIFT算法长。实验中OpenMP优化的ASIFT用了4个CPU,4个线程。总的看来GASIFT与OpenMP优化的ASIFT和标准ASIFT相比,有较好效果。对于300*211的图像,产生1300个关键点,GPU优化效果不明显,只有1倍的加速比,与OpenMP优化类似。但是对于2048*1536的大图像,产生近50000个关键点,GPU优化比标准ASIFT有16倍的加速比,比OpenMP优化也有7倍的加速比。对于大小为800*640的中等图片,产生10000个关键点,加速比也有7倍。因为申请显存池需要额外花费1s的时间,所以总的计算优化对于很小的图片来说不是特别明显,但是对于中、大图片还是有较好的结果。

4.2 CPU参与并行计算模式

在CPU也参与并行计算的这种模式中,根据实验环境,CPU有4核,因此程序中设定了4个线程,其中一个负责控制GPU端计算,另外3个在CPU端并行计算图像特征。

由于CPU参与并行计算时,涉及到CPU/GPU端任务的划分,因此,针对不同大小的图片,进行不同CPU/GPU任务的划分。本节对GASIFT运行时间进行了一个测试,测试结果如表2所列。第一行表示CPU/GPU任务比,0/41表示CPU只负责逻辑计算。

表2 GASIFT在CPU/GPU不同任务划分下的比较(s)

图像大小 \ CPU/GPU任务比	20/21	10/31	2/39	1/40	0/41
300*211	0.710	0.946	0.980	0.991	0.8
640*480	2.970	1.525	1.602	1.639	1.640
800*640	4.945	2.151	1.962	2.010	2.02
1600*1200	17.596	6.911	3.737	3.820	3.800
2048*1536	29.568	10.877	6.153	5.121	5.060

从测试结果看,运算时间跟任务划分有关:CPU负责的任务越多,计算时间越长,总体性能不如CPU只负责逻辑计算的模式。其原因主要在于针对图像特征提取的应用,GPU的加速比很高,整个计算运行时间并不长。使用CPU并行计算的优化效果不明显,还会引入多线程的开销。因此,在ASIFT提取特征点算法中,只适合使用CPU负责逻辑计算、GPU负责并行计算的方式,并不适合使用CPU、GPU都负责并行计算任务的方式。

4.3 匹配能力比较

本节对GASIFT与ASIFT、SIFT的匹配能力进行了一个比较实验。采用大小为640*480像素的图片,对图片进行了平移、旋转、视点、模糊变换,最后识别结果如表3所列,表中的数字是匹配点个数。

表3 GASIFT、ASIFT、SIFT匹配能力比较

	GASIFT	ASIFT	SIFT
平移	985	1354	147
旋转	749	829	139
视点变换	418	233	42
模糊变换	1030	1167	103

从表3可以看出,ASIFT算法无论是计算平移、旋转、视点变换、模糊变换都比SIFT算法获得的匹配点多。GASIFT与ASIFT算法在匹配点的个数上基本一致。GASIFT对平移计算得到的匹配点比ASIFT少,但是对视点变换计算得到的匹配点比ASIFT多。这样的差异主要是由3.1节所描述的计算关键点描述符时的不同做法导致的。总体来说,GASIFT的匹配能力较好,优于SIFT算法。

结束语 本文详细描述了如何利用GPU实现对ASIFT算法的并行优化,先利用GPU对ASIFT计算中各个可以并行加速的部分进行了优化,之后讨论了在CPU/GPU协同工作的两种模式上对ASIFT的优化。实验表明对于ASIFT算法的GPU加速,使用CPU负责逻辑计算、GPU负责并行计算的模式,能获得很好的加速效果,与CPU端ASIFT相比可达16倍加速比。而使用CPU/GPU协同工作的第二层模式,即CPU也参与并行计算的模式,加速效果跟CPU、GPU端任务分配有关,并且引入了多余线程开销,因此不适合应用在图像特征提取上。

本文讨论的GPU加速都是在单节点单GPU上进行。对于2048*1536大图片,计算ASIFT特征点仍然需要5s的时间,如果想要进一步优化,还可以考虑在单节点多GPU及多节点多GPU上并行计算。

GASIFT可以被用于涉及到图像匹配的各个领域中,我们下一步的研究重点在于将GASIFT运用到实时的图像检索中,对图像检索的性能进行提升。

参考文献

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

[2] Yan Ke. PCA-SIFT: A more distinctive representation for local image descriptors[C]//CVPR. Washington. DC, USA, 2004: 66-75

[3] Bay H, Tuytelaars T, Van Gool L. SURF: Speeded up robust features, 2006 [C]// Proc. European Conference on Computer Vision. 2006: 404-417

[4] Heymann S, Maller K, Smolic A, et al. SIFT implementation and optimization for general-purpose GPU, 2007 [C]// Proc. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. 2007: 1-8

[5] 王瑞,梁华,蔡宣平,基于GPU的SIFT特征提取算法研究 [J]. 现代电子技术,2010,33(15):41-46

[6] Dağason K, Lejsek H, Ársæll T, et al. GPU acceleration of Eff2 descriptors using CUDA, 2010 [C]//Proceedings of the International Conference on Multimedia. Firenze, Italy, 2010: 1167-1170

[7] Morel J M, Yu G. ASIFT: A New Framework for Fully Affine Invariant Image Comparison [J]. SIAM Journal on Imaging Sciences, 2009, 2(2): 1597-1600

[8] Hare J S, Samangooei S, Dupplaw D P. OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images, 2011 [C]// Proceedings of the 19th ACM international conference on Multimedia. Scottsdale, Arizona, USA, 2011: 691-694

[9] Chu Bin, Jiang Da-lin. Panoramic Image Stitching Using ASIFT, 2012 [C]// Fourth International Conference on Multimedia Information Networking and Security. 2010: 216-219

[10] Yin Chun-xia, Li Cheng-rong, Liu Hong-lin, et al. Experimental Contrast of Several Typical Algorithms for Local Features Detection, 2012 [C]// International Conference on Mechanical Engineering and Automation Advances in Biomedical Engineering. 2012: 65-71

[11] Panga Yan-wei, Lia Wei, Yuan Yuan, et al. Fully affine invariant SURF for image matching[J]. Neurocomputing, 2012, 85: 6-10

[12] 卢风顺，宋君强，银福康，等. CPU/GPU 协同并行计算研究综述[J]. 计算机科学，2011，38(3)：5-9

[13] 王永明，王贵锦. 图像局部不变性特征与描述[M]. 北京：国防工业出版社，2010：79-87

[14] Morel J-M, Yu Guo-shen. ASIFT: online demo[OL]. http://www. cmap. polytechnique. fr/～ yu/research/ASIFT/demo. html

(上接第 13 页)

[14] Asterjadhi A, Zorzi M. JENNA: a jamming evasive network-coding neighbor-discovery algorithm for cognitive radio networks [Dynamic Spectrum Management] [J]. Wireless Communications, IEEE, 2010, 17(4): 24-32

[15] Zhang Lu, Pei Qing-qi, Li Hong-ning. Anti-jamming Scheme Based on Zero Pre-shared Secret in Cognitive Radio Network [C]// Proceedings of Computational Intelligence and Security (CIS), 2012 Eighth International Conference on. Guangzhou, China, 2012: 670-673

[16] 聂晓文，卢显良，唐晖，等. 基于洗牌策略的 Sybil 攻击防御[J]. 电子学报，2008(11)：2144-2149

[17] 陈珊珊，杨庚，陈生寿. 基于 LEACH 协议的 Sybil 攻击入侵检测机制[J]. 通信学报，2011，32(8)：143-149

[18] Yu Bo, Xu Cheng-zhong, Xiao Bin. Detecting Sybil attacks in VANETs[J]. Journal of Parallel and Distributed Computing, 2013, 73(6): 746-756

[19] Kumar R N, Bapuji V, Govardhan A, et al. An Improvement to Trust Based Cross-Layer Security Protocol against Sybil Attacks (DAS) [J]. Computer Engineering and Intelligent Systems, 2012, 3(7): 62-70

[20] El Zoghby N, Cherfaoui V, Ducourthial B, et al. Distributed Data Fusion for Detecting Sybil Attacks in VANETs[M]// Belief Functions: Theory and Applications. Springer Berlin Heidelberg, 2012: 351-358

[21] Chen Ze-sheng, Todor C, Chen Chao, et al. Modeling primary user emulation attacks and defenses in cognitive radio networks [C]// Proceedings of 2009 IEEE 28th International Performance Computing and Communications Conference (IPCCC). Scottsdale, AZ, 2009: 208-215

[22] Jin Z, Anand S, Subbalakshmi K. Detecting primary user emulation attacks in dynamic spectrum access networks[C]// Proceedings of 2009 IEEE International Conference on Communications(ICC'09). Dresden, German, 2009: 1-5

[23] Jin Z, Anand S, Subbalakshmi K. Mitigating primary user emulation attacks in dynamic spectrum access networks using hypothesis testing[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2009, 13(2): 74-85

[24] Liu Y, Ning P, Dai H. Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures[C]// Proceedings of 2010 IEEE Symposium on Security and Privacy (SP). Oakland, CA, USA, 2010: 286-301

[25] Wang Wen-kai, Li Hu-sheng, Sun Y L, et al. Attack-proof collaborative spectrum sensing in cognitive radio networks[C]// Proceedings of the 43rd Annual Conference on Information Sciences and Systems(CISS 2009). Baltimore, MD, 2009: 130-134

[26] Zhu F, Seo S-W. Enhanced robust cooperative spectrum sensing in cognitive radio[J]. Journal of Communications and Networks, 2009, 11(2): 122-133

[27] Min A W, Shin K G, Hu Xin. Attack-tolerant distributed sensing for dynamic spectrum access networks[C]// Proceedings of 17th IEEE International Conference on Network Protocols (ICNP 2009). Princeton, NJ, 2009: 294-303

[28] Chen Rui-liang, Park Jung-min, Bian Kai-gui. Robust distributed spectrum sensing in cognitive radio networks[C]// Proceedings of the 27th IEEE Conference on Computer Communications(INFOCOM 2008). Phoenix, AZ, 2008: 1876-1884

[29] Chen R, Park J-M J, Bian K. Robustness against byzantine failures in distributed spectrum sensing[J]. Computer Communications, 2012, 35(17): 2115-2124

[30] Rawat A S, Anand P, Chen Hao, et al. Countering byzantine attacks in cognitive radio networks [C]// Proceedings of 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP). Dallas, TX, 2010: 3098-3101

[31] Nguyen-Thanh N, Koo I. An enhanced cooperative spectrum sensing scheme based on evidence theory and reliability source evaluation in cognitive radio context[J]. Communications Letters, IEEE, 2009, 13(7): 492-494

[32] Li Hu-sheng, Han Zhu. Catching attacker (s) for collaborative spectrum sensing in cognitive radio systems: An abnormality detection approach[C]// Proceedings of 2010 IEEE Symposium on New Frontiers in Dynamic Spectrum. Singapore, 2010: 1-12

[33] Tang H, Yu F R, Huang M, et al. Distributed consensus-based security mechanisms in cognitive radio mobile ad hoc networks [J]. IET Communications, 2012, 6(8): 974-983

[34] Zhang Yong-guang, Lee Wen-ke. Intrusion detection in wireless ad-hoc networks[C]// Proceedings of the 6th annual international conference on Mobile computing and networking. ACM. New York, NY, USA, 2000: 275-283

[35] Pei Q, Li H, Ma J, et al. Defense against objective function attacks in cognitive radio networks[J]. Chinese Journal of Electronics, 2011, 20(4): 138-142

[36] Rathnayake U, Petander H, Ott M, et al. EMUNE: architecture for mobile data transfer scheduling with network availability predictions[J]. Mob. Netw. Appl., 2012, 17(2): 216-233

[37] Hernandez-Serrano J, León O, Soriano M. Modeling the lion attack in cognitive radio networks[J]. EURASIP Journal on Wireless Communications and Networking, 2011, 2011: 1-10