第43卷 第6A期
2016年6月

计 算 机 科 学
Computer Science

Vol. 43 No. 6A
June 2016

移动智能终端的SIFT特征检测并行算法

甘 威 张素文 雷 震 李怡凡
(武汉理工大学自动化学院 武汉 430070)

摘 要 特征的检测和匹配在计算机视觉应用中是一个重要的组成部分,如图像匹配、物体识别和视频跟踪等。SIFT算法以其尺度不变性和旋转不变性在图像配准领域得到了广泛应用。传统的SIFT算法效率低,因此提出一种在移动智能终端上实现的高效方法。在Android平台利用OpenCL框架实现了移动智能终端的SIFT算法,通过计算任务的重新分配,优化SIFT算法在移动GPU上的并行实现。实验结果表明,移动平台的SIFT算法充分利用了GPU并行计算能力,大大提高了SIFT算法的执行效率,实现了高效的特征检测。

关键词 SIFT,OpenCL,GPU,特征检测
中图法分类号 TP391 文献标识码 A

SIFT Feature Extraction Parallel Algorithm on Mobile Device

GAN Wei ZHANG Su-wen LEI Zhen LI Yi-fan
(College of Automation, Wuhan University of Technology, Wuhan 430070, China)

Abstract Feature extraction and matching is an important part in computervision applications, such as image matching, object recognition and video tracking. SIFT algorithm is widely used in the field of image registration because of its scale invariance and rotation invariance. To deal with the low efficiency of the traditional SIFT algorithm, we proposed an efficient method which is implemented on mobile platform. In this paper, we used the OpenCL to achieve the SIFT algorithm on mobile device, through redistributing the calculation of tasks and optimizing the SIFT algorithm in the mobile OpenCL parallel implementation. Experimental results show that our SIFT algorithm takes full advantage of the GPU parallel computing power, greatly improving the efficiency of the SIFT algorithm, achieving the efficient feature extraction.

Keywords SIFT, OpenCL, GPU, Feature extraction

1 引言

随着移动硬件的高速发展,移动应用程序可以完成更加强大的功能。特征检测在图像处理应用中相当重要,在移动领域越来越受到重视。尺度不变特征变换(SIFT)算法[1]被认为是当前应用最广泛的一种特征提取算法,对缩放和旋转保持不变性,对仿射变换、光照、噪声也保持局部不变性。但是,SIFT算法时间复杂度高,难以在实时环境下使用。我们需要改进SIFT的实现方式来提高其应用性。目前处理器的计算能力越来越强大,各种异构并行编程模型都得以实现,显著地提升了SIFT算法的效率。

Sinha等提出了基于OpenGL的SIFT特征提取算法[2],但是它只适合桌面应用,对于移动应用有局限性。Kayombya提出了在智能手机上使用OpenGL ES2.0的SIFT实现[3],但是内存传输开销过大。Blaine Riste等在其基础上进行了优化[4],能够有效提速,但它受限于OpenGL的计算性能。Weiyan Wang等提出了基于OpenCL的SIFT算法的实现[5],其应用在NVIDIA和AMD平台,需要拓展其通用性。Guohui Wang等提出了基于OpenCL的SIFT特征检测算法在智能手机上的初步实现[6],需要进一步优化。本文在以上工作的基础上,优化CPU-GPU之间的任务分配,平衡数据计算和数据传输的耗时,采用针对移动平台的优化策略,实现了在移动智能终端上基于OpenCL的SIFT特征检测算法。

2 SIFT特征匹配算法描述

SIFT特征检测是在尺度空间中进行的。首先生成图像尺度空间,然后检测尺度空间中的局部极值点,再通过剔除低对比度点和边缘响应点对局部极值点进行精确定位。在对特征进行描述时,先计算每个极值点的主方向,对以极值点为中心的区域进行直方图梯度方向统计,生成特征描述符,从而完成图像特征检测。

2.1 建立高斯差分尺度空间

Koenderink和Lindeberg等证明了高斯核是唯一的尺度空间核,为了得到更多检测点,将原始图像放大一倍作为基准图像[1],然后将不同尺度的高斯核函数与原始图像做卷积,得到一组不同尺度的模糊图像,将上一组第4幅图像尺寸减半,生成下一组模糊图像的原始图像,重复高斯滤波操作,即得到下一组图像,重复上面步骤,得到高斯金字塔。金字塔相邻图像相减来构造高斯差分尺度空间。

甘 威(1988—),男,硕士生,主要研究方向为计算机应用,E-mail:815571403@qq.com;张素文(1964—),女,教授,主要研究方向为模式识别、图像处理。

2.2 确定特征点

为了确定特征点，需要比较高斯差分尺度空间三维邻域的 26 个像素，如为极值点，则此点是图像在该尺度下的一个候选关键点。为了高效遍历，需要先上下层比较，再左右比较。检测到的所有极值点即为所有的候选关键点。通过插值法来确定关键点的位置和尺度，同时剔除对比度低的点和不稳定的边缘点，即可得到特征点。

2.3 计算梯度大小和方向

在以特征点为中心的邻域窗口，用直方图来统计该区域中所有点的梯度方向，其中方向直方图将 360°方向分成 36 份。每个梯度方向用高斯窗函数加权，直方图的峰值对应该特征点的梯度主方向。

2.4 生成特征描述符

在上述步骤已经得到了所有特征点，特征描述符即为这些特征点的邻域像素在尺度空间内梯度主方向的统计向量。特征点周围的 $16*16$ 像素窗口分成 16 份，即 $4*4$ 的子区域，计算每个子区域的 8 个方向的梯度方向直方图，得到 128 维特征向量，最后规范化该向量，得到特征描述符。

3 移动智能终端上的 SIFT 并行算法

本文使用算法在 ezsift 的基础上实现[7]，ezsift 是 C++写的一个不需要第三方包的开源 SIFT 算法实现，运行效率高。在安卓平台上使用 NDK 编程，针对移动智能终端的计算和存储能力都比计算机差等缺点，对原算法进行改进，并利用移动 GPU 的并行计算能力，改写算法中适合并行处理的步骤。

3.1 环境设置及平台优化

本文使用高通骁龙 Snapdragon 600 和 Adreno 320 硬件平台，系统在 Android 4.2.2 下进行测试，该平台支持 OpenCL1.1 标准。Guohui Wang 证明使用 ARM-v7a 编译 SIFT 算法程序的耗时只有 ARM-v5a 编译程序下耗时的 21%[6]。由于本文采用的平台支持硬件浮点运算，为了提高速度，也采用 ARM-v7a 编译。

为了在安卓上使用 SIFT 算法，需要将该算法的 C++程序实现编译成库，使用 jni 来完成本地调用。为了支持标准 C++的特性使用 stlport_static 运行库，相比 gnustl_static 运行库，其生成的文件体积小一些，算法运行效率较高。测试发现在 NDK 中直接构建共享库时相比编译为静态库然后并入共享库有 3 倍的加速比，因为静态库与共享库连接会增加应用程序的内存读取开销，对于移动平台的特定图像算法实现可以直接将其编译为共享库，减少应用程序处理图像算法的开销。本文测试算法的加速性能采用共享库方法。

3.2 算法实现优化

在算法描述中已经介绍了 SIFT 的主要步骤，Blaine Riste 等指出高斯滤波是 SIFT 算法中最耗时的步骤[4]，本文使用 ezsift 测试了两幅典型图像的 SIFT 特征检测，发现高斯金字塔耗时占据 SIFT 特征检测时间的 50%。因此减少此部分的时耗最能提高 SIFT 算法的实时性。在 CPU 端计算出不同尺寸的高斯滤波核，作为参数上载到 GPU 端，在 GPU 端来完成高斯滤波操作。

在图像预处理阶段，为了获取更多特征点，Lowe 建议将源图像放大一倍[1]。在不放大源图像的情况下检测时间大幅度减少，只有放大图像检测时间的 20%，可以大大提高移动设备的运行速度，但是会减少 64%的特征点。当匹配要求高、需要大量的特征点时则加倍图像；反之则直接使用源图像，可以根据具体情况取舍。

高斯金字塔的组数是由输入图像的分辨率得到的，因为要进行隔点降采样，所以在执行降采样生成高斯金字塔时，要一直执行到不能降采样为止，但图像太小又毫无意义，因此只需要有限次数的采样即可。高斯金字塔每组生成 6 幅图像，这里使用循环高斯滤波实现。只需要创建一次 OpenCL 程序对象及内核，改变每次送入缓存区的内存对象，读出输出结果即得到滤波后的图像。高斯差分金字塔是在高斯金字塔的基础上将相邻两幅图像相减得到的，每组生成 5 幅图像。其可以与高斯金字塔步骤合并为一个 kernel，减少 CPU-GPU 中间数据传输的耗时。

确定特征点时，需要大量的比较计算，会引入很多条件分支分歧，这对于移动 GPU 的高速计算是不适合的，此部分放在 CPU 上执行更有效。而且在这里计算可以保持一致的参数设置门限，获得与原 SIFT 算法一致的特征点结果，保证了特征点检测的质量。整个算法程序是分步骤执行的，每个步骤分别实现 OpenCL 的 GPU 计算，且只需要获取一次平台，创建一个上下文及一个命令队列，其余的分支程序(按照算法步骤划分)直接使用即可。该算法不需要每次都去重新创建，节约了硬件资源。本文实现算法的流程如图 1 所示。

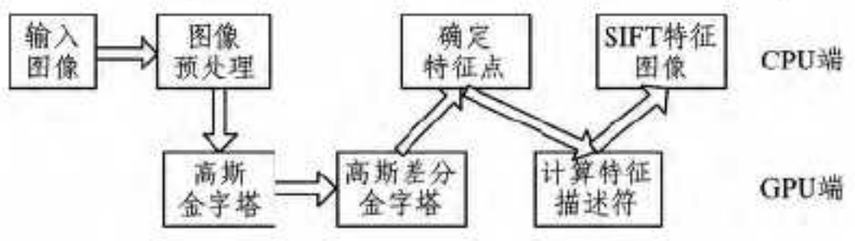

图 1 算法实现流程

3.3 内存与数据结构

在 OpenCL 中支持全局内存、私有内存、局部内存、常量内存、图像对象。使用图像对象可以自动处理越界和滤波模式，具有缓存功能、编码简洁等优点。使用全局内存可以更自由地处理各种长度的数据类型，可以更灵活地读写和处理数据。张樱等提出滤波窗口大于 $7*7$ 时，使用 buffer 和 LDS 可以取得更好的性能[8]。虽然利用图像对象处理在 OpenCL 中得到支持，但是在处理高斯金字塔时会有不同尺寸的滤波，为了更方便地处理数据，也采用 buffer 方法来处理内存对象，图像数据较大，超过了局部内存空间，所以使用全局内存来储存图像数据。由于 kernel 核函数都需要使用到图像数据的长宽等数据，将这些数据打包为一个数组 buffer 内存对象，这样参数在内存中是连续存储的，利于内存读取。针对 NDRange 的优化，对 $8*8$、$11*11$ 和 $16*16$ 线程组 3 种情况进行实验，结果发现 $16*16$ 组合最优。贾海鹏提出在每个 CU 上同时并发运行足够数目的 wave-front，可以隐藏访存延迟[9]，针对移动 GPU 此优化也有不错的效果。

4 实验结果及分析

图 2 示出本文算法特征点检测效果，可以看出所提算法有效地检测出了特征点。

在测试本文算法实现性能时，与 OpenCV 中经典 SIFT 算法的耗时做了对比，采用了不同大小的 5 幅图像，为了保持特征点数量一致，参数设置门限是一样的。测试效果见表 1。

图 2　graf(800 * 640)、lena(512 * 512)特征检测效果图

表 1　测试结果数据表

图像大小	特征点数目（个）	OpenCV 算法耗时(ms)	本文算法耗时(ms)	加速比
324 * 223	604	677	384	1.76
512 * 512	1109	1571	766	2.05
800 * 640	2907	3437	1391	2.47
1000 * 1348	4025	6616	2827	2.34
1600 * 1200	8413	10803	4676	2.31

由表 1 数据可看出本文算法实现了一定的加速，大尺寸图像比小尺寸图像取得了更好的效果，说明大尺寸图像更好地发挥了移动 GPU 的计算能力，掩盖了 CPU 与 GPU 之间数据传输的时间，但是图像尺寸继续增大后，加速效果不会进一步提升，这时数据传输开销过大，需要很多耗时。同时在移动平台上的实现较 PC 平台实现的接近 10 倍加速也有很大的差距[10]，这主要受限于移动终端的硬件性能。随着未来移动终端性能更加强大，可以取得更强大的加速能力。

结束语　本文实现了 SIFT 算法在移动平台的 GPU 加速，并针对安卓平台上的实现做了相应的优化，使得算法运行效率得到了提升。未来可以尝试更多的优化策略，进一步探索算法加速的可能性。同时可以尝试实现更多的图像处理加速算法，使移动 GPU 加速图像处理应用更加广泛。

参 考 文 献

[1] Lowe D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110

[2] Sinha S N, Frahm J M, Pollefeys M, et al. Feature tracking and matching in video using programmable graphics hardware[J]. Machine Vision & Applications, 2011, 22(1): 207-217

[3] Kayombya G R. SIFT feature extraction on a Smartphone GPU using OpenGL ES2.0[D]. Massachusetts: Massachusetts Institute of Technology, 2011

[4] Rister B, Wang G, Wu M, et al. A fast and efficient sift detector using the mobile GPU[C]// 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2013: 2674-2678

[5] Wang Wei-yan, Zhang Yun-quan, Long Guo-ping, et al. CLSIFT: An Optimization Study of the Scale Invariant Feature Transform on GPUs[C]// IEEE 15th International Conference on High Performance Computing and Communication. 2013

[6] Wang G, Rister B, Cavallaro J R. Workload Analysis and Efficient OpenCL-based Implementation of SIFT Algorithm on a Smartphone[C]// Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2013: 759-762

[7] Wang G. An easy-to use standalone SIFT library easy-to-use standalone SIFT library written in C/C++[CP/OL]. http://sourceforge.net/projects/ezsift

[8] 张樱，张云泉，龙国平. 基于 OpenCL 的图像模糊化算法优化研究[J]. 计算机科学，2012，39(3)：260-264

[9] 贾海鹏，张云泉，徐建良. 基于 OpenCL 的图像积分图算法优化研究[J]. 计算机科学，2013，40(2)：1-7

[10] 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)：41-43

(上接第 148 页)

重要性的非均匀简化策略。本文提出 3 种简化后的枝干重构方法，并对 3 种方法的优缺点进行分析对比。另外，还使用纹理模板进行树冠重构，并针对简化后植物模型的存储空间、绘制效率和视觉质量 3 个方面进行对比。实验数据表明，本文提出的面向网络应用的植物模型简化技术不但具有较高的模型简化率，且简化后的模型具有较高的视觉质量。因此，面向网络应用的树木模型简化方法能够大大提高网络应用中的三维树木模型的传输和绘制效率。

参 考 文 献

[1] Lee C H, Varshney A, Jacobs D W. Mesh saliency[J]. ACM Transactions on Graphics, 2005, 24(3): 659-666

[2] Marco T, Nico P, Paolo C, et al. Practical quad mesh simplification[J]. Computer Graphics Forum, Blackwell Publishing Ltd, 2010, 29(2): 407-418

[3] 金勇，吴庆标，刘利刚. 基于变分网格的曲面简化高效算法[J]. 软件学报，2011，22(5)：1097-1105

[4] 偶春生，张佑生. 基于特征保持的曲面网格简化[J]. 计算机应用研究，2013，30(10)：3162-3164

[5] Morigi S, Rucci M. Multilevel mesh simplification [J]. Visual Computer, 2014, 30(5): 479-492

[6] Remolar I, Chover M, Ribelles J, et al. Geometric Simplification of Foliage[C]// Proc of Eurographics. 2002

[7] Bao Guan-bo, Li Hong-jun, Zhang Xiao-peng, et al. Realistic real-time rendering for large-scale forest scenes[C]// 2011 IEEE International Symposium on VR Innovation (ISVRI). IEEE, 2011: 217-223

[8] Bao Guan-bo, Li Hong-jun, Zhang Xiao-peng, et al. Large-scale forest rendering: real-time, realistic, and progressive[J]. Computers & Graphics, 2012, 36(3): 140-151

[9] Lee J, Kuo C C J. Tree model simplification with hybrid polygon billboard approach and human-centered quality evaluation[C]// 2010 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2010: 932-937

[10] Gumbau J, Chover M, Remolar I, et al. View-dependent pruning for real-time rendering of trees[J]. Computers & Graphics, 2011, 35(2): 364-374

[11] Qin Xue-ying, Eihachiro N, Katsumi T, et al. Fast Photo Realistic Rendering of Trees in Daylight[J]. Computer Graphics Forum, Blackwell Publishing, 2003, 22(3): 243-252

[12] Zeng Ji-guo, Zhang Yan, Zhan Shou-yi. 3D tree models reconstruction from a single image[C]// Sixth International Conference on Intelligent Systems Design and Applications (ISDA). IEEE, 2006: 445-450

[13] Marcelo D G M, Walter M. A hybrid geometry and billboard-based model for trees[C]// 2011 Brazilian Symposium on Games and Digital Entertainment (SBGAMES). IEEE, 2011: 17-25

[14] Livny Y, Pirk S, Cheng Zhang-lin, et al. Texture-lobes for tree modelling[M]// ACM Transactions on Graphics (TOG). ACM, 2011: 76-79

[15] Itti L, Koch C. A saliency-based search mechanism for overt and covert shifts of visual attention[J]. Vision Research, 2000, 40(10-12): 1489-1506