平面单应性矩阵求解的 CUDA 并行实现*

袁红星，吴少群，朱仁祥，诸葛霞
（宁波工程学院 电子与信息工程学院，浙江 宁波 315016）

摘 要：针对平面单应性矩阵实时求解的需求，提出了 RANSAC 单应性算法在 GPU 上并行实现的方法。首先计算出理论上最优的迭代次数；然后根据迭代间数据的独立性将每次迭代看成一个独立的任务映射到 CUDA 线程上。实验表明，对于较高分辨率的图像，GT 240M 平台上平均加速比达到了8。

关键词：单应性；RANSAC；GPU；CUDA

中图分类号：TP391 **文献标识码**：A **文章编号**：1674-7720(2012)23-0038-04

CUDA-based parallel implementation of homography estimation

Yuan Hongxing, Wu Shaoqun, Zhu Renxiang, Zhuge Xia
(School of Electronics and Information Engineering, Ningbo University of Technology, Ningbo 315016, China)

Abstract: A massively parallel RANSAC algorithm implementation in GPU is proposed to meet the real-time requirements for homography estimation. At first, optimal iteration number is calculated. Then, each iteration task is assigned to a CUDA thread based on data independent between iterations. Experimental results on a GT 240M graphics card prove that the RANSAC homography algorithm could be used to deliver a speed-up of 8× with high resolution images on average.

Key words: homography; RANSAC; GPU; CUDA

平面单应性是一个射影线性变换，直线在该变换下仍映射为直线[1]。平面单应性矩阵在图像与计算机视觉处理方面有着重要的作用，单应性矩阵中包含着摄像机的内外参数信息。单应性矩阵已被广泛用于视觉测量[2-3]、图像拼接[4]、图像配准[5]、视觉伺服[6]和三维重构[7]等领域中。平面单应性在实际应用中面临着实时性的挑战，随着采集图像分辨率的不断提高，对计算速度不断提出更高的要求。算法的精度、速度、抗干扰性以及并行实现成为人们追求的目标。近年来，图像处理器(GPU)大规模并行计算的推广与普及为平面单应性的实时计算提供了硬件平台。NVIDIA 公司提出的统一计算设备架构 CUDA (Compute Unified Device Architecture)[8]编程模型显著降低了基于 GPU 的并行化难度和工作量。CUDA 编程模型对 C 语言进行了扩展，将 CPU 和 GPU 分别视为 Host 和 Device，两者协同工作。CPU 承担逻辑性强的事务和串行计算任务，而 GPU 负责数据密集型的并行计算任务。

已有的平面单应性算法主要包括基于特征点匹配的方法和基于像素匹配的方法。基于特征点的方法首先提取图像的特征点，计算描述符，然后进行匹配，最后利用匹配点估计单应性矩阵。基于像素匹配的方法利用光流或运动估计求解单应性矩阵[9]。

由于具有线性、简单性[10]和更广泛的应用场合[11]，基于特征点匹配的方法得到更多的关注。常用的特征点提取与匹配算法有 SIFT 算法[12]和 SURF 算法[13]。其中，SURF 是 SIFT 的简化算法，两者都具有旋转和尺度不变性。Zeng Hui 等人提出了基于归一化线匹配的平面单应性矩阵求解算法[10]。但在线接近或通过坐标原点时，这种匹配方法估计的单应性将会出错。Li Xiaowei 等人将基于 RANSAC 的平面单应性求解算法应用于增强现实中的图像配准[14]。SERRADELL E 等人将几何先验知识引入特征点的匹配中，以提高单应性求解的准确度[11]。现有研究要么关注单应性求解的准确性问题，要么强调单应性新的应用场合，很少考虑计算实时性的问题。随着采集设备分辨率的不断提高，这些算法的计算速度将难以满足高速计算机视觉处理的需要。本文以最基本和应用最广泛的 RANSAC 平面单应性求解算法为例，探讨了平面

*基金项目：国家自然科学基金（No. 61071173）；浙江省自然科学基金（No. LY12F01001）；宁波市自然科学基金（No. 2012A610043）

单应性矩阵求解在 GPU 上并行化的实现。

1 RANSAC 平面单应性求解算法

1.1 平面单应性定义

平面单应性是一个 3×3 非奇异矩阵 $\boldsymbol{H}$ 表示的射影变换。对于某个平面上的三维点，如果 $\boldsymbol{x}=[x\ \ y\ \ 1]^{\mathrm{T}}$ 和 $\boldsymbol{x}'=[x'\ \ y'\ \ 1]^{\mathrm{T}}$ 分别是其在两个图像平面上成像点的齐次坐标，则它们和该平面的单应性矩阵 $\boldsymbol{H}=[h_{11}\ \ h_{12}\ \ h_{13};h_{21}\ \ h_{22}\ \ h_{23};h_{31}\ \ h_{32}\ \ h_{33}]^{\mathrm{T}}$ 存在如下关系：

$$m\begin{pmatrix}x'\\y'\\1\end{pmatrix}=\begin{pmatrix}h_{11}&h_{12}&h_{13}\\h_{21}&h_{22}&h_{23}\\h_{31}&h_{32}&h_{33}\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}\tag{1}$$

其中，m 表示非零的任意大小尺度因子。因此，单应性矩阵 $\boldsymbol{H}$ 的自由度为 8。由式(1)和向量积的定义，可得到如下关系：

$$\boldsymbol{x}'\times\boldsymbol{H}\boldsymbol{x}=0\tag{2}$$

其中，“×”表示向量积。式(2)表明每对匹配点定义了两个独立线性方程：

$$\begin{cases}h_{11}x+h_{12}y+h_{13}-h_{31}xx'-h_{32}yx'-x'=0\\h_{21}x+h_{22}y+h_{23}-h_{31}xy'-h_{32}yy'-y'=0\end{cases}\tag{3}$$

理论上 4 对匹配点就可以达到求解 $\boldsymbol{H}$ 的目的。实际应用中，匹配点的数目远大于 4，称之为超定问题，常采用最小二乘法予以解决。假设有 n 对匹配点，且第 i 对匹配点的齐次坐标分别为 $\boldsymbol{x}=[x_i\ \ y_i\ \ 1]^{\mathrm{T}}$ 和 $\boldsymbol{x}'=[x_i'\ \ y_i'\ \ 1]^{\mathrm{T}}$ $(i=1,2,\cdots,n)$，则线性方程可写为 $\boldsymbol{A}\boldsymbol{h}=0$。其中，

$$\boldsymbol{A}=\begin{pmatrix}x_1&y_1&1&0&0&0&-x_1x_1'&-y_1x_1'&-x_1'\\0&0&0&x_1&y_1&1&-x_1y_1'&-y_1y_1'&-y_1'\\x_2&y_2&1&0&0&0&-x_2x_2'&-y_2x_2'&-x_2'\\0&0&0&x_2&y_2&1&-x_2y_2'&-y_2y_2'&-y_2'\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\x_n&y_n&1&0&0&0&-x_nx_n'&-y_nx_n'&-x_n'\\0&0&0&x_n&y_n&1&-x_ny_n'&-y_ny_n'&-y_n'\end{pmatrix}\tag{4}$$

$$\boldsymbol{h}=[h_{11}\ \ h_{12}\ \ h_{13}\ \ h_{21}\ \ h_{22}\ \ h_{23}\ \ h_{31}\ \ h_{32}\ \ h_{33}]^{\mathrm{T}}\tag{5}$$

对矩阵 $\boldsymbol{A}$ 进行奇异值分解，其最小特征值对应的特征向量即为单应性矩阵 $\boldsymbol{H}$ 的解[15]。

1.2 算法步骤

图 1 所示为本文 RANSAC 平面单应性矩阵求解的主要步骤。

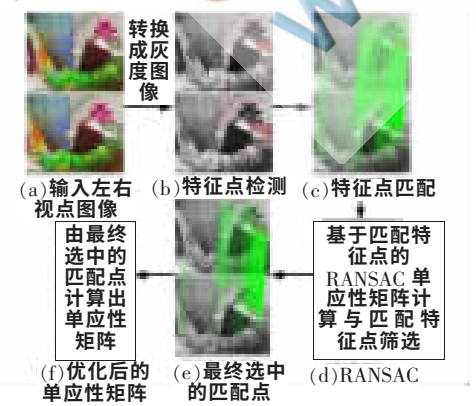

图 1 RANSAC 平面单应性算法流程示意图

算法步骤如下：

(1)将内点集合 I 设为空集，迭代索引 i 设为 0，确定理论上最优的迭代次数 N。

(2)左右视点图像上特征点提取与匹配。

(3)将当前内点集合 I_i 设为空集；从匹配的特征点集合中随机选择 n 对匹配点，根据式(2)计算平面单应性矩阵 $\boldsymbol{H}_i$。

(4)将满足 $\|\boldsymbol{H}_i\boldsymbol{x}-\boldsymbol{x}'\|<\varepsilon$ 的所有匹配点称为与 $\boldsymbol{H}_i$ 一致的匹配点，加入集合 I_i。

(5)如果 I_i 中元素个数大于 I，则令 $I=I_i$；否则，直接跳到步骤(6)。

(6)迭代索引 i 递增 1，如果 i 大于或等于 N，跳到步骤(7)；否则跳到步骤(3)。

(7)将集合 I 中匹配点代入式(2)重新计算平面单应性矩阵。

为了节省每次迭代所需的计算量，常将步骤(3)中的 n 设为 4，即求解式(2)所需的最少匹配点对。

2 算法的 CUDA 并行实现

上述算法的步骤(2)中特征点提取与匹配已有相关的 CUDA 并行化研究[16-17]。步骤(3)~(6)，从匹配点集合随机选取 n 对匹配点计算单应性矩阵 $\boldsymbol{H}_i$，并确定与其一致的匹配点集合 I_i 的方法相同，而处理的数据相互独立，这满足 GPU 并行化处理的条件。

在 CUDA 并行编程模型下，GPU 通过建立 k 个线程，将任务分解成 k 个子任务，每个线程独立执行一个子任务来实现任务的并行化处理。因而将步骤(3)~(6)用 CUDA 并行实现时，需要将其进行任务分解，并映射到 CUDA 线程中。RANSAC 平面单应性算法 CUDA 并行实现模型如图 2 所示。

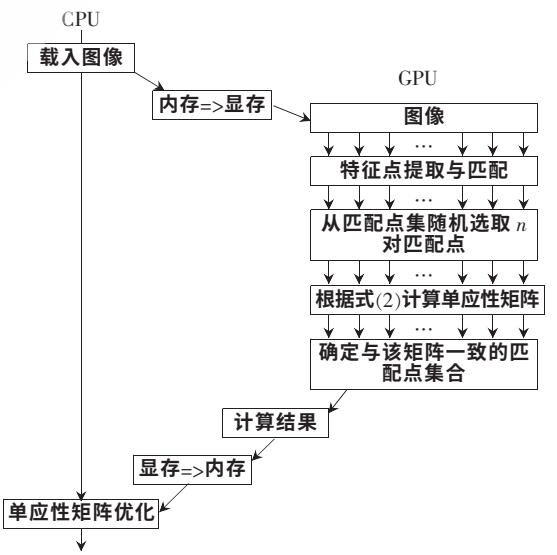

图 2 RANSAC 平面单应性算法 CUDA 计算模型

HARTLEY R 等人的研究成果表明[15]，步骤(3)~(6)理论上的最优迭代次数为：

$$K=\log(1-p)/\log(1-(1-\varepsilon)^s) \quad (6)$$

其中，p 表示从样本集合中随机抽取一个样本时至少有一个样本是内点的概率；ε 表示样本为外点的概率；s 是每次迭代中用于求解单应性矩阵的点对数目。在本文的 CUDA 实现中，将这 K 次迭代看成是 K 个独立的子任务。假设 CUDA 中每个线程块包含的线程数目为 M，则线程块的数目为：

$$L=\lceil K/M \rceil \quad (7)$$

算法 CUDA 实现步骤如下：

Host 端：

(1)根据式(6)和式(7)确定 GPU 线程块的数目；生成一个大小为 K 的随机数数组 indexH；生成大小为 K 的内点数组 inH；为 Device 端分配对应的随机数数组 indexD 和内点数组 inD。

(2)在 Device 中分配内存，并将 Host 端输入图像、随机数链表等相关数据拷贝到 Device 端。

Device 端：

(3)特征点提取与匹配。

(4)计算线程索引 i，根据 indexD[i]取出计算式(2)的 4 对匹配点，由奇异值分解得到单应性矩阵的估计值 H_i；将与 H_i 一致的匹配点对加入 inD[i]。

Host 端：

(5)将 Device 端的内点数组 inD 内容拷贝到 Host 端的 inH 中。

(6)遍历 inH，找到匹配点数目最多的元素索引号 j。

(7)将 inH[j]中所有匹配点对代入式(2)，重新计算平面单应性矩阵。

其中步骤(3)采用了参考文献[17]所提的 SURF 算法的 CUDA 并行化实现。步骤(4)中用于求解单应性矩阵的奇异值分解采用了参考文献[18]中的方法。

3 实验结果与分析

为了验证本文实现的有效性，实验环境配置如下：

(1)Intel Core2 Duo 2.20 GHz CPU，2 GB 内存，Microsoft Visual Studio 2008。

(2)GeForce GT 240M 1.21 GHz GPU，16 KB 共享内存，436 MB 全局内存，CUDA toolkit 和 SDK 4.0，NVIDIA Driver for Windows 7(275.33)。

以图 3 所示的两个视点图像在不同分辨率下，分别从计算精确度和加速比两个方面与 OpenCV2.3[19]中的单应性求解算法进行了对比。测试图像的分辨率列表如表 1 所示。

(a)左视点图像 (b)右视点图像

图 3 测试图像

表 1 测试图像标记

测试图像标记	A1	A2	A3	A4	A5
分辨率	200×134	400×268	800×536	1200×804	1600×1072

3.1 精确度比较实验

分别用 OpenCV2.3 RANSAC 单应性算法和本文 CUDA 实现的 RANSAC 单应性算法计算图 3 所示图像在不同分辨率下的单应性矩阵，以均方根误差 RMSE(Root Mean Square Error)衡量两者间的差异，其结果如表 2 所示。从表 2 可以看出，两者之间存在较小的差异，这是因为 OpenCV2.3 的单应性算法采用了双精度浮点运算，而本文实验所用的 GPU 只支持单精度浮点运算。

3.2 计算速度比较实验

图 4 给出了在不同分辨率下 OpenCV2.3 和本文 CUDA 实现平面单应性算法的计算时间。从图中可以看出，经过 GPU 并行实现后，加速比在 1.6~8.0 之间。图像分辨率较大时加速比也较大。

表 2 精确度比较

测试图像标记	OpenCV2.3			本文 CUDA 实现			RMSE
A1	1.634 007	0.132 208	−112.416 448	1.634 300	0.134 655	−112.481 815	0.021 988
	0.179 672	1.496 275	−29.289 939	0.174 678	1.490 942	−29.285 548	
	0.002 716	0.001 665	1.000 000	0.002 757	0.001 661	1.000 000	
A2	1.486 425	−0.030 185	−187.505 451	1.487 108	−0.033 197	−187.519 693	0.012 789
	0.167 357	1.309 013	−48.509 272	0.169 038	1.304 690	−48.544 460	
	0.001 232	−0.000 017	1.000 000	0.001 285	−0.000 013	1.000 000	
A3	1.467 914	−0.008 946	−373.188 465	1.454 481	−0.008 550	−373.113 220	0.038 424
	0.155 899	1.311 789	−93.497 948	0.158 395	1.312 089	−93.584 198	
	0.000 571	0.000 051	1.000 000	0.000 562	−0.000 050	1.000 000	
A4	1.495 063	−0.015 317	−570.680 169	1.491 955	−0.015 470	−570.444 916	0.083 661
	0.168 443	1.323 706	−148.464 502	0.161 551	1.323 169	−148.551 636	
	0.000417	0.000 006	1.000 000	0.000 441	0.000 006	1.000 000	
A5	1.575 588	−0.002446	−812.408629	1.575 919	−0.002 452	−812.200 928	0.069 812
	0.19 3895	1.397429	−228.497562	0.196 476	1.390 070	−228.523 323	
	0.00 0362	0.000 037	1.000 000	0.000 363	0.000 383	1.000 000	

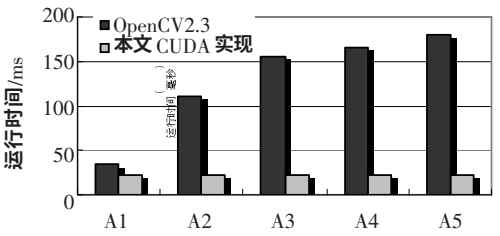

图 4 计算时间比较

本文针对常用的 RANSAC 单应性算法，根据算法迭代间数据的独立性，提出 GPU 上并行化的方法。对于 1 600×1 072 分辨率的图像，在 GT 240M 显卡上的计算时间仅需 23 ms，能够满足实时性的要求。下一步，将对 RANSAC 算法的并行实现进行优化，进一步提高计算速度。另外，将在支持双精度浮点运算的显卡上比较 CPU 与 GPU 实现的计算精度问题。

参考文献

[1] 孙凤梅，胡占义.平面单应矩阵对摄像机内参数约束的一些性质 [J]. 计算机辅助设计与图形学学报，2007，19(5)：647-650.

[2] 王亮，戴宪彪，居鹤华.一种基于单应的月球车车轮沉陷视觉测量方法[J].宇航学报，2011，32(8)：1701-1707.

[3] 吴斌，朱洪岩，肖心通等.视觉测量中基于单应性矩阵的平面靶标图像特征提取 [J]. 光电子激光，2011，22(8)：1211-1215.

[4] Wu Fuli, Fang Xianyong. An improved RANSAC homography algorithm for feature based image mosaic[C]. Proceedings of the 7th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision, Athens, Greece, 2007: 24-26.

[5] SALVI J, MATABOSCH C, FOFID, et al. A review of recent image registration methods with accuracy evaluation[J]. Image and Vision Computing, 2007, 25: 578-596.

[6] VARGAS M, MALIS E. Visual servoing based on an analytical homography decomposition[C]. Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain, 2005:12-15.

[7] 李晓明，秦茜茜.基于准稠密匹配的结构化场景三维重建 [J]. 计算机辅助设计与图形学学报，2011，23(5)：849-854.

[8] NVIDIA Corporation. CUDA programming guide 4.0 [OL]. [2011-05-26] http://www.developer.nvidia.com/.

[9] 张鸿燕，李托拓，耿征.彩色图像的单应矩阵估计算法[J]. 中国图象图形学报，2011，16(2)：287-292.

[10] Zeng Hui, Deng Xiaoming, Hu Zhanyi. A new normalized method on line-based homography estimation [J]. Pattern Recognition Letters 2008(29):1236-1244.

[11] SERRADELL E, ÖZUSAL M, LEPETIT V, et al. Combining geometric and appearance priors for robust homography estimation [C]. Proceedings of the European Conference on Computer Vision, 2010:58-72.

[12] LOWE D. Distinctive image features from scale-invariant keypoints [J]. International Journal of Computer Vision, 2004, 60(2): 91-110.

[13] BAY H, ESS A. SURF: speeded up robust features[J]. Computer Vision and Image Understanding, 2008, 110(3):346-359.

[14] Li Xiaowei, Liu Yue, Wang Yongtian, et al. Computing homography with RANSAC algorithm: a novel method of registration [C]. Proceedings of SPIE, Bellingham, USA, 2005: 109-112.

[15] HARTLEY R. Multiple view geometry in computer vision (2nd ed)[M]. Cambridge University Press, 2003.

[16] Wu Changchang. SiftGPU: a GPU implementation of scale invariant feature transform(SIFT)[OL]. [2011-09-03]. http://cs.unc.edu/~ccwu/siftgpu/.

[17] TERRIBERRY T, FRENCH L, HELMSEN J. GPU accelerating speeded-up robust features[C]. Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission, Atlanta, USA, 2008: 355-362.

[18] LAHABAR S, NARAYANAN P J. Singular value decomposition on GPU using cuda[C]. Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, Washington, DC, USA, 2009: 1-10.

[19] OpenCV 2.3. OpenCV [OL]. [2011-09-03]. http://www.opencv.org.cn/.

（收稿日期：2012-09-12）

作者简介：

袁红星，男，1980 年生，博士，高级工程师，主要研究方向：信号与信息处理、3D 视频信号获取与处理。