JOURNAL OF IMAGE AND GRAPHICS

主办：中国科学院遥感与数字地球研究所
中国图象图形学学会
北京应用物理与计算数学研究所

中国图象图形学报

2013
12
VOL.18

ISSN1006-8961
CN11-3758/TB

图像分割 P1610

中国图象图形学报

刊名题字：宋健 | 月刊（1996年创刊）

第18卷第12期（总第212期）
2013年12月16日

中国精品科技期刊
中国国际影响力优秀学术期刊
中国科技核心期刊

主管单位 中国科学院
主办单位 中国科学院遥感与数字地球研究所
中国图象图形学学会
北京应用物理与计算数学研究所

主　　编 李小文
编辑出版 《中国图象图形学报》编辑出版委员会
邮政信箱 北京9718信箱
邮　　编 100101
电子信箱 jig@irsa.ac.cn
电　　话 010-64807995
网　　址 www.cjig.cn

广告经营许可证 京朝工商广字第0361号
总 发 行 北京报刊发行局
订　　购 全国各地邮局
海外发行 中国国际图书贸易集团有限公司
（邮政信箱：北京399信箱　邮编：100048）
印刷装订 北京科信印刷有限公司

Journal of Image and Graphics
Title inscription: Song Jian | Monthly, Started in 1996

Superintended by Chinese Academy of Sciences
Sponsored by Institute of Remote Sensing and Digital Earth, CAS
China Society of Image and Graphics
Institute of Applied Physics and Computational Mathematics

Editor-in-Chief LI Xiaowen
Editor, Publisher Editorial and Publishing Board of Journal of Image and Graphics
P.O.Box 9718, Beijing, P.R.China
Zip code 100101
E-mail jig@irsa.ac.cn
Telephone 010-64807995
Website www.cjig.cn

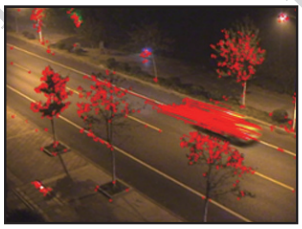
Distributed by Beijing Bureau for Distribution of Newspapers and Journals
Domestic All Local Post Offices in China
Overseas China International Book Trading Corporation
(P.O.Box 399, Beijing 100048,P.R.China))
Printed by Beijing Kexin Printing Co., Ltd.

CN 11-3758/TB
ISSN 1006-8961
CODEN ZTTXFZ

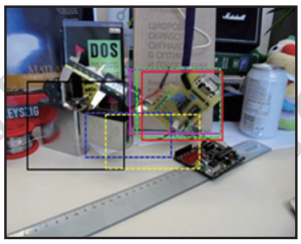
国外发行代号 M1406
国内邮发代号 82-831
国内定价 50.00元

2013年12月 第18卷 第12期（总第212期）
中国图象图形学报 Zhongguo Tuxiang Tuxing Xuebao

目次



稀疏光流快速计算的动态目标检测与跟踪(第1593页)

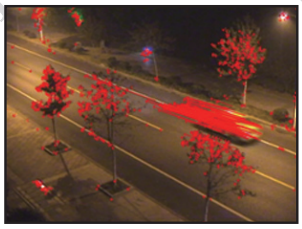


字典学习联合粒子滤波鲁棒跟踪(第1628页)

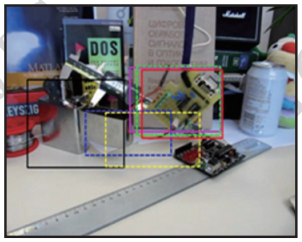
结合SURF特征的多功能彩色图像水印算法(第1694页)

Volume 18, Number 12
Published December 16, 2013

CONTENTS
JOURNAL OF IMAGE AND GRAPHICS



Dynamic target detection and tracking based on fast computation using sparse optical flow (P1593)



Object tracking based on learning a dictionary joint practical filter(P1628)

Multi-purpose watermark algorithm of color images based on SURF(P1694)

E-mail: jig@irsa.ac.cn
Website: www.cjig.cn
Tel: 010-64807995

中国图象图形学报
JOURNAL OF IMAGE AND GRAPHICS

中图法分类号：TP391　　**文献标识码**：A　　**文章编号**：1006-8961(2013)12-1610-10
论文引用格式：李宏益，吴素萍 . Mean Shift 图像分割算法的并行化[J]. 中国图象图形学报，2013，18(12)：1610-1619. [DOI：10.11834/jig.20131209]

Mean Shift 图像分割算法的并行化

李宏益，吴素萍
宁夏大学数学计算机学院，银川　750021

摘　要：图像分割作为高性能并行计算的一个主要应用领域，其算法本身的时间复杂度和实时性需求要求不断改进计算机硬件技术和并行处理的算法。Mean Shift 算法是图像分割领域一个比较经典的算法，在图像分割过程中，不需要任何先验知识，是一种无监督的分割过程，在图像分割的具体实现中应用广泛。利用 TBB(threading building block)工具和 CUDA(compute unified device architecture)对 Mean Shift 算法进行多核和 GPU(graphic processing unit)并行化改造。首先分析 Mean Shift 分割过程中最耗时的部分 Mean Shift 聚类，然后利用 TBB 和 CUDA 对 Mean Shift 聚类进行了并行化改造，并对两种并行方法进行了对比分析。实验结果表明，两种并行方法都取得了较好的加速效果，加速比都随着图像增大和带宽参数的增加而增大，基于 TBB 的加速比稳定趋于核数。
关键词：Mean Shift；并行计算；TBB；CUDA；图像分割

Parallelization of Mean Shift image segmentation algorithm

Li Hongyi，Wu Suping
School of Mathematics and Computer Science，Ningxia University，Yinchuan　750021，China

Abstract：Image segmentation is a main field of application in parallel computing. To achieve the high performance needed，the algorithm needs to make use of the improved computer hardware and parallel computing algorithms. Mean Shift algorithm is a relative classic algorithm in image segmentation field，which needs no prior knowledge and is an unsupervised segmentation process，attracting widespread attention for its good applicability. In this paper，we give two parallel improvement methods of Mean Shift using TBB (threading building block) and CUDA (compute unified device architecture) based on Multi-core and GPU (graphic processing unit) processing. First，the most time-consuming part the Mean Shift iteration in the process of Mean Shift image segmentation，is analyzed，then two parallel improvement methods of the Mean Shift iteration using TBB and CUDA are given，Two parallel methods are compared and analyzed. The experimental results show that，two kinds of parallel methods have achieved preferable acceleration effect，with the increase of the image and bandwidth parameter the speedup of two parallel methods is on the increases，and the speedup based on TBB tends to be equal to the number of CPUs.
Key words：Mean Shift；parallel computing；threading building block (TBB)；CUDA；image segmentation

收稿日期：2013-02-07；**修回日期**：2013-06-18
基金项目：国家自然科学基金项目(60963004)；国家星火计划项目(2011GA880001)
第一作者简介：李宏益(1985—)，男，宁夏大学计算机软件与理论专业硕士研究生，主要研究方向为图形图像处理、并行处理与高性能计算。
通讯作者：吴素萍，E-mail：wspg123@163.com

0 引　言

Mean Shift 算法是一种有效的迭代统计算法,由 Fukunaga 于 1975 年在一篇基于概率密度函数的梯度估计的文献中提出的[1],该文章发表 20 年来,一直没有引起人们的广泛关注,直到 1995 年 Cheng 发表了第 2 篇关于 Mean Shift 的重要文章[2]。Cheng 对 Mean Shift 算法进行了两方面的改进:一方面,引入了核函数,随着样本与被偏移点的距离不同,其偏移量对均值偏移向量的贡献也不同;另一方面,增加了权重函数,不同的样本点对均值向量的影响不一样,扩大了 Mean Shift 算法的使用范围。到 1997 年及以后 Comaniciu 等人将 Mean Shift 应用到了特征空间分析,在图像平滑和图像分割方面都取得了很好的结果[3-5]。

Mean Shift 分割具有良好的性能和无参化的特点,运用 Mean Shift 算法进行图像分割的研究主要分为 3 类:1)引入 Mean Shift 进行图像分割和在非速度上的改进,主要对图像颜色空间、带宽的选择以及 Mean Shift 分割后的处理进行研究[6-13];2)对 Mean Shift 在速度上和分割结果上所进行的改进[14-16];3)对 Mean Shift 算法进行并行化研究[17-19]。下面针对以上 3 类进行说明:

1)Mean Shift 算法在图像分割中的应用

(1)颜色空间转换　Mean Shift 进行图像分割时由于 RGB 颜色空间的色彩差异无法度量和 RGB 颜色空间的相关性,因此一般将图像由 RGB 空间转到 CIELUV 空间或者 HSV 空间[9-11]。其中 CIELUV 空间是国际照明委员会 1973 年提出的,L 表示亮度,U 和 V 表示色差;HSV 空间中 H 是色调,S 是饱和度,V 是亮度,这些空间中各分量之间相关性弱,易于计算,颜色差异易于表示。

(2)自适应带宽选择　在 Mean Shift 分割的过程中,采用自适应的带宽,而不是指定固定带宽[11-12]。文献[11]中带宽公式为

$$H_i = \frac{\boldsymbol{L}^{\mathrm{T}}\boldsymbol{I}_5}{lk_{r,i}}\left\{\mathrm{tr}\left[\sum (x_{i,j}-\bar{x}_i)(x_{i,j}-\bar{x}_i)^r\right]\right\}^{\frac{1}{2}} \tag{1}$$

式中,$\boldsymbol{L}=\{l_h,l_s,l_v,l_x,l_y\}$,$l$ 为 $\boldsymbol{L}$ 所有分量的平均值,$\boldsymbol{I}_5$ 为 5 阶单位阵,$k_{r,i}$为 r 半径内的样本总数,$x_{i,j}$是落入半径为 r 内的点的值,$\bar{x}_i$ 为

$$\bar{x}_i = \frac{1}{k_{r,i}}\sum_{j=1}^{k_{r,i}} x_{i,j} \tag{2}$$

文献[12]修正了最优带宽的判别条件,给出了自适应滤波器参数的设计方法。

(3)分级 Mean Shift　在初次迭代获得的聚类中心基础上采用不同的带宽矩阵进行多次聚类,从而获得不同级的聚类中心集合,并建立一个归属树结构,最终通过叶节点与根节点的归属关系进行归类从而完成图像分割[13]。

(4)融合多特征的 Mean Shift 算法　在 Mean Shift 分割的过程中,不仅考虑图像的颜色空间、坐标信息,还要考虑图像的纹理特征,抽取的纹理特征包括对比度、极性、各向异性[10]。

2)串行加速的 Mean Shift 算法

(1)共轭梯度法的加速,将共轭梯度法引入到 Mean Shift 聚类过程中,利用了共轭梯度法存储需求小,收敛速度快并具有全局收敛性的特点[14];

(2)动态改变样本点,在聚类的过程中,以每次聚类中心的值代替原来图像中的像素值进行计算,加快聚类的收敛速度[15];

(3)记录搜索路径,在进行 Mean Shift 聚类过程中,记录所有经过的像素点,如果在均值移动的过程中,遇到已经标记过的像素点则停止均值漂移,该像素点的收敛点与已标记过的像素的收敛点相同,直接将收敛值赋值给该点[15-16];

(4)与其他分割算法结合,在图像分割过程中,先使用 Mean Shift 算法进行聚类,确定分类的类别总数和各类别的类心,然后以类别数和类心为输入,使用 k-means 算法得到图像最终的分割结果。

3)并行加速的 Mean Shift 算法

(1)基于集群的并行,文献[17-18]在集群上实现了并行的 Mean Shift 算法对遥感图像的分割,采用了 VC++编程语言和 MPI 库。为了消除并行分块分割中普遍存在的分割线的问题,文献[17]采用了在图像分块处两边加冗余数据的办法。文献[18]给出了并行 Mean Shift 算法的计算硬时间指标,由于集群的各主机的 CPU 主频和核心数不一致,所以该文中没有给出统计加速比。

(2)基于 GPU(图形处理器)的并行文献[19]对遥感图像分割分别使用了 CPU(Intel Core2 2.4G)和 CUDA(计算统一设备架构)(Quadro FX 3700M)进行了加速。得到了大约 20 倍的加速比。

(3)基于多核的并行,也有文献使用多线程技

中国图象图形学报
JOURNAL OF IMAGE AND GRAPHICS
Vol. 18, No. 12, Dec. 2013

术在多核平台下对 Mean Shift 进行了加速，在处理器核数从 1 增加到 2 的时候加速比接近 2，但是再增加处理器核数直到 3、4 时效果都不明显。

使用 C++ 语言实现了 Mean Shift 串行算法，分析出了最耗时的部分 Mean Shift 聚类，然后分别应用 TBB（threading building block）和 CUDA 对 Mean Shift 聚类进行并行化加速，取得了较好的加速效果，最后对两种方法进行了比较分析。

1 Mean Shift 基本原理[1-3]

给定 d 维空间 $\mathbf{R}^d$ 中的 n 个样本点 $\boldsymbol{x}_i$，$i=1,2,\cdots,n$，在 x 点的 Mean Shift 向量的基本形式定义为

$$\boldsymbol{M}_k(\boldsymbol{x}) = \frac{1}{k}\sum_{\boldsymbol{x}_i\in S_h}(\boldsymbol{x}_i - \boldsymbol{x}) \tag{3}$$

$\boldsymbol{y}$ 点的集合

$$\boldsymbol{S}_h(\boldsymbol{x}) \equiv \{\boldsymbol{y} \mid (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{x}) \leqslant h^2\} \tag{4}$$

式中，$\boldsymbol{S}_h$ 是一个半径为 h 的高维球区域，k 表示在这 n 个样本点 $\boldsymbol{x}_i$ 中，有 k 个点落入 $\boldsymbol{S}_h$ 区域中。可以看出$(\boldsymbol{x}_i-\boldsymbol{x})$是样本点 $\boldsymbol{x}_i$ 相对于点 $\boldsymbol{x}$ 的偏移，式(3)定义的 Mean Shift 向量 $\boldsymbol{M}_k(x)$是对落入区域 $\boldsymbol{S}_h$ 中的 k 个样本点相对于点 $\boldsymbol{x}$ 的偏移向量求和之后再求平均。如果样本点 $\boldsymbol{x}_i$ 从一个概率密度函数 $f(x)$ 中采样得到，因为非零的概率密度梯度指向概率密度增加最大的方向，所以，$\boldsymbol{S}_h$ 区域内的样本点更多地落在沿着概率密度梯度的方向，对应的 Mean Shift 向量 $\boldsymbol{M}_h(\boldsymbol{x})$应该指向概率密度梯度的方向。

2 Mean Shift 串行实现

2.1 算法步骤

Mean Shift 图像分割可分为 3 个过程：1）图像预处理过程，包括图像滤波和颜色空间变换；2）Mean Shift 聚类过程即 Mean Shift 算法；3）区域合并过程。

本文图像滤波采用中值滤波，颜色空间变换将图像变换到 LUV（L 为亮度，U 和 V 为色差）空间，Mean Shift 算法的一般过程对每一个像素点$\{\boldsymbol{x}_i \mid 1\leqslant i\leqslant n\}$执行以下步骤：

1）初始化 $j=1$，$\boldsymbol{y}_{i,1}=\boldsymbol{x}_i$；

2）用 $\boldsymbol{y}_{i+1} = \dfrac{\sum_{i=1}^{n}\boldsymbol{x}_i g\left(\left\|\dfrac{\boldsymbol{x}_i-\boldsymbol{y}_j}{h}\right\|\right)}{\sum_{i=1}^{n} g\left(\left\|\dfrac{\boldsymbol{x}_i-\boldsymbol{y}_j}{h}\right\|\right)}$ 计算 $\boldsymbol{y}_{i,j+1}$ 直到均值漂移向量 $\|\boldsymbol{y}_{j+1}-\boldsymbol{y}_j\|$ 小于设定的 ε 值，将收敛的值记为 $\boldsymbol{y}=\boldsymbol{y}_{i,c}$；

3）$\boldsymbol{z}_i=(\boldsymbol{x}_i^s, \boldsymbol{y}_i^r)$。

步骤 3）将原来的坐标，收敛的颜色空间值重新组成 5 维数据 $\boldsymbol{z}_i$。

在 Mean Shift 聚类过程中用到两个带宽参数 hs 和 hr。hs 是空间带宽，在分割过程中用于决定 $\boldsymbol{S}_h$ 半径的大小，hs 越大，$\boldsymbol{S}_h$ 的范围越大，落入 $\boldsymbol{S}_h$ 的样本数越多。hr 是颜色带宽，用于决定落入 $\boldsymbol{S}_h$ 范围内的点是否可以参加式(3)的计算，如果落入 $\boldsymbol{S}_h$ 的点的颜色值与当前点的颜色值的差的平方根小于 hr，该点满足条件，hr 越大，满足条件的样本点越多。

使用 Mean Shift 算法进行聚类后的结果，会出现聚类到同一个收敛点的像素数目过少情况，需要将这些小区域进行合并。通过设定阈值 M 的大小来调整最后分割结果中容许最小的小区域的像素数目，本文取 70 和 100 两种情况。在 2.2 节的实验结果示例中可以看到不同的 M 值对分割结果的影响。

对一幅大小为 256 × 256 的 Lena 图像使用 Mean Shift 图像分割算法，分割参数为$(hs, hr, M)=(5,7,70)$时的结果如图 1 所示。

图 1 Mean Shift 分割结果

Fig. 1 Segmentation result of Mean Shift algorithm

2.2 实验结果

给出大小为 256 × 256 的两幅 Lena 图像（其中一幅为带噪声，另一幅为不带噪声），分别在分割参数(hs, hr, M)取$(5,7,70)$、$(5,7,100)$、$(7,12,70)$、$(7,12,100)$进行分割，结果如图 2 所示。从图 2 中可以看出，无噪声的图像分割效果较好，分块数少，因为不存在完全去除噪声的算法，噪声的存在影响了分割过程中的收敛点。带宽越大，分割之后的小区域数越少，每次迭代时参加计算的点增多，漂移到共同点的可能性大。

(a) (*hs*,*hr*,*M*)=(5,7,70)　(b) (*hs*,*hr*,*M*)=(5,7,100)　(c) (*hs*,*hr*,*M*)=(7,12,70)　(d) (*hs*,*hr*,*M*)=(7,12,100)

图 2　Mean Shift 分割结果

Fig. 2　Segmentation result of Mean Shift algorithm

表 1 给出一幅大小为 256 ×256 的 Lena 图像,在 Intel 酷睿双核 1.73 G 主频的 CPU 上,分割参数(*hs*,*hr*,*M*)分别取(5,7,70)、(5,7,100)、(7,12,70)、(7,12,100)时 Mean Shift 分割过程各部分所占时间比。

表 1　256 ×256 图像分割过程中各部分所占时间比

Table 1　256 ×256 time of the portion in image segmentation process

分割参数 (*hs*,*hv*,*M*)	预处理时间 /ms	预处理所占百分比 /%	Mean Shift 所占的时间/ms	Mean Shift 所占百分比/%	合并时间 /ms	合并所占百分比/%
(5,7,70)	47	1.635	2 781	96.73	47	1.635
(5,7,100)	47	1.653	2 766	97.26	31	1.09
(7,12,70)	46	0.769	5 922	98.964	16	0.267
(7,12,100)	47	0.785	5 922	98.947	16	0.267

3　基于 TBB 的 Mean Shift 的并行化

3.1　TBB 介绍

TBB 是由 Intel 针对多核平台开发的一组开源 C + +模板库,基于 GPLv2 的开源证书,支持可扩展的并行编程[20]。TBB 可以在 Windows、Linux 和 OSX 上运行,支持 Intel、Microsoft 和 GNU 工具,能够满足大多数用户的需求。

TBB 编程模式通常是使用模板作为并行迭代的模型,TBB 提供了丰富的 C + +模板。TTB 提升了程序的伸缩性,并且完全支持嵌套的并行编程。程序员可以根据需要创建自己的并行组件以及构建大型的并行程序[21]。

3.2　基于 TBB 并行化要点

1)并行方式的选择

基于 TBB 的并行化方式有很多种,可以是基于循环的并行化,基于流水线的并行化,基于任务的并行化。基于循环的并行化是算法中有 For 循环,For 循环里面的数据相互独立或者没有读写、写写冲突即可,此外基于循环的并行化方式是其余两种并行化方式的封装,在使用上更方便。基于流水线的并行化方式适用的情况是算法分为几个阶段,从第一阶段到最后一个阶段,不断有数据从第一个阶段流入,至最后一个阶段流出。基于任务的并行化适用的是算法可以分阶段,算法的每个阶段的操作都相

中国图象图形学报
JOURNAL OF IMAGE AND GRAPHICS
Vol. 18, No. 12, Dec. 2013

同,后面的阶段使用前面阶段的结果或者不使用。Mean Shift 图像分割算法是对图像中的每一个像素点进行迭代操作,将 2 维的图像点以 1 维的形式进行计算,可以使用 For 循环对每一个像素点进行循环迭代,本文算法选择基于循环的并行化方式。

2)选择迭代空间

TBB 迭代空间有 blocked_range < T > 、blocked_range2d,还可以根据需要构造适合自己需要的迭代空间形式。Mean Shift 图像分割算法将图像展开成了 1 维的形式,所以该算法实现选用 blocked_range <T > 。

3)设计 operator 循环体函数

operator 函数是基于 TBB 并行化中真正的执行体,Mean Shift 聚类过程在其中完成。该函数看似串行,其实编译之后在执行过程中根据环境变量的设置或者物理 CPU 的数目产生相同数量的执行实体。在该函数内区间的定位使用迭代器,所以需要自己控制好计算边界,不能产生读写冲突或写写冲突。

4)确定划分粒度

在 parallel_for(blocked_range < size_t > (start, end, gransize) 的调用过程中,需要指定 gransize 的大小。其中 start、end 表示整个空间的起始地址。gransize 的大小决定了每一个处理器核一次取的数据的长度,gransize 的值越大,即数据划分的粒度越大,gransize 的值越小,即数据划分的粒度越小。粒度大小不一样所产生的并行效率是不一样的。文献[22]的方法是首先将 gransize 的值初始化为 10 000,然后逐步减小 gransize 的值并记录程序的运行时间,将并行对串行的加速比绘制图像,取峰值点的加速比对应的 gransize 值为最适合的 gransize 值,该方式可以取得最好的加速比所需的粒度大小,但是该方法费时费力。TBB 提供了一个简单的方式,使用 auto_partitioner,粒度划分的任务由任务调度器处理。

3.3 基于 TBB 并行化实现

TBB 并行化是基于 2.1 节的 Mean Shift 图像分割的串行算法实现,该串行算法的步骤 1)中值滤波采用了线性时间选择算法,不考虑进行并行化改造,串行算法的步骤 3)小区域合并过程复杂,该过程在整个分割过程中所占时间比例很小,也不进行并行化改造。由于 Mean Shift 分割过程中步骤 2)Mean Shift 聚类在整个分割过程中所占比例最大,如表 1 所示,所占时间的百分比最高约近 99% 。因此只针对串行算法的步骤 2)Mean Shift 聚类进行基于 TBB 的并行化改造。

在 Mean Shift 图像分割的串行算法实现中,参与计算 Mean Shift 向量的原始点的颜色值存放在一个数组中,一个点迭代稳定之后的数据存放在另外一个数组中,对于图像中每一个点的迭代过程只需要在原始的颜色值数组中取值,之后将计算结果写回另一个结果数组,根据这一特性,图像中所有的像素点的聚类过程对原始数据数组和结果数据数组不存在写写和读写冲突,所以可以利用 TBB 的基于循环的并行模式进行并行化,将数据进行分块处理。

在串行程序的实现过程中,对图像中每一个像素点分别进行 Mean Shift 聚类,在并行实现时,需要将这些数据进行分组,让每一个 CPU 的处理器核心负责一段数据的迭代计算过程,如果有 80 000 个数据,在 4CPU 核心的计算机进行并行,可将数据分成块,其划分可以是 0,…,9 999;10 000,…,19 999;20 000,…,29 999;30 000,…,39 999;40 000,…,49 999;50 000,…,59 999;60 000,…,69 999;70 000,…,79 999 共 8 组。算法过程如下:

1)构建 For 循环体类

class MeanShiftTBB

2)包装 For 循环类

使用了 auto_partitioner( )划分器

void meanshiftFilterTBB (float sigmaS, float sigmaR, float* msRawData, int bandcount, int imagewidth, int imageheight, float* LUV_data)

3)调用包装函数

meanshiftFilterTBB(sigmaS, sigmaR, msRawData, N, width, height, data)

4 基于 CUDA 的 Mean Shift 并行实现

与 TBB 并行类似,只针对 2.1 节 Mean Shift 图像分割的串行算法实现中的步骤 2)Mean Shift 聚类进行基于 CUDA 的并行化改造。

4.1 CUDA 并行化要点

进行基于 CUDA 的并行化程序设计,需要考虑很多因素,主要有如下几个方面[22]:

1)使用页锁定内存

在基于 CUDA 的并行编程模式中,主机端内存分为两种:1)可分页内存,即一般所使用的内存模式,由操作系统的 API 进行分配和释放操作,malloc( )负

责申请内存,free( ) 负责释放内存;2) 页锁定内存,永远不会被移动到虚拟内存中,一致存在于物理内存中,可以通过 DMA 加速与设备端(GPU)进行通信,由 cudaHostAlloc( ) 和 cudaFreeHost( ) 控制申请和释放操作。

在某些设备上还可以不需要拷贝数据直接将数据映射到设备(GPU)的地址空间,使 GPU 直接访问内存,节省了主存和显存之间数据拷贝的时间。在申请内存时加上标志 cudaHostAllocPortable 可使多个 CPU 线程共享一块页锁定内存,以此提高 CPU 线程之间通信的效率。在申请内存时加上 cudaHostAllocWriteCombined 标志,CPU 不再处理该申请块内存的缓存一致性,这样数据在经过 PCI-e 总线传输期间不会被 CPU 监视中断,原则上可以加快数据传输的效率。虽然页锁定内存具有一些好处,但是不能大量使用,因为如果大量使用页锁定内存,这些内存只能提供给 GPU 使用,用于可分页内存的物理内存的数量将会显著减少,会降低系统的整体性能。

2) CPU 和 GPU 的异步执行模式

CPU 和 GPU 的异步执行模式是指 GPU 的内核操作或者数据在主存和显存之间交换的过程中,CPU 不一直等待这些操作的返回,而是继续执行自己代码,提高 CPU 的利用率。内存与显存之间的数据交换可以通过异步函数实现,CUDA 的异步函数一般以 Async 结尾。可以通过使用流来实现 CPU 的运算和 GPU 内核运算之间的异步执行,最后需要用一个等待异步执行完成的函数来检查这些异步操作是否已经完成。

3) 使用共享存储器

共享存储器是 GPU 片内的高速存储器,可以被一个 block 内的所有线程进行读写访问。其访问速度与寄存器的访问速度一样快,是实现线程间通信延迟的最好办法。

4) 优化指令运算

CUDA 的硬件特别适合使用单精度浮点运算,在精度要求不高的情况下应尽可能地使用单精度浮点数进行运算,此外尽可能使用 CUDA 提供的算术运算函数;一些控制指令如 For、While 等尽可能地考虑展开来写,或者通过使用#pragma unroll 来展开循环以提高速度;访问存储器时应该使每个线程访问的位数宽度与寄存器位数宽度一致,如果大于则进行拆分访问,如果小于则进行合并访问。

4.2　算法的 CUDA 并行化

从 Mean Shift 图像分割的串行算法的实现中可知,图像中的所有数据可以同时进行计算,可以利用 CUDA 数据并行的方式进行并行化改造。

在串行程序的实现过程中,对图像中每一个像素点分别进行 Mean Shift 迭代。使用 CUDA 进行并行化就是将每一个像素点的计算过程映射到 GPU 的每一个线程上进行计算,根据 GPU 硬件的特殊性,GPU 将 8 个物理线程分为一组(block),为了隐藏延迟,一个 block 上至少要运行 64 个以上的逻辑线程,但是最大不能超过 512 个,逻辑线程的数量要求是 32 的倍数。一个逻辑线程根据 GPU 的轻量级的特点实际上就是并行的基本单元,本文算法就是一个像素点的迭代过程。一幅大小为 1 024 × 1 024 的图像,每一个 block 的数量文中设定为 256,数据可分为 4 096 块。

Mean Shift 算法进行图像分割的计算过程是图像中每个点不断迭代计算直至收敛的过程。将每个点的计算过程分配给 CUDA 中的一个线程,让一个 CUDA 线程负责图像中的一个点的收敛计算过程,算法过程如下:

1) 为显卡开辟空间和传递数据到显存,该过程在 CPU 上完成

```
// 开辟空间
CUDA_SAFE_CALL( cudaMalloc( ( void ** ) &device_LUV
_data, sizeof( float ) * L* bandcount) );
// 传递数据到显存
CUDA_SAFE_CALL( cudaMemcpy( device_LUV_data,
LUV_data, sizeof( float ) * L* bandcount, cudaMemcpyHostTo-
Device) );
```

2) 设置线程数目和块数目

```
int threadnum = 256;
int blocknum = L/threadnum + ( L% threadnum? 1:0);
```

3) 调用内核函数 meanShiftCUDAKernel,其函数原型如下:

```
__global__ static void meanShiftCUDAKernel
( float * msRawData, float * LUV_data, float sigmaS, float
sigmaR, int imagewidth, int imageheight, float minEPSILON, int
maxLIMIT, float minDELTA)
```

4) 将显卡上的结果数据拷贝回内存

```
//将结果传递至内存
CUDA_SAFE_CALL( cudaMemcpy( msRawData, device_
msRawData, sizeof( float ) * L* bandcount, cudaMemcpyDevice-
ToHost) );
```

5)释放显存空间

```
//释放显存
CUDA_SAFE_CALL(cudaFree(device_LUV_data));
CUDA_SAFE_CALL(cudaFree(device_msRawData));
```

5 结果和分析

5.1 实验结果

Mean Shift 图像分割的并行化研究中使用的硬件环境为 CPU Q9400 2.66 GHZ(4 CPU),GPU NDVIDA Geforce GTX 260(24 个 Multi-Stream Processors,192 个 Stream Processors),内存 4 G;软件环境:Windows XP;编程环境:Microsoft Visual Studio 2008,tbb30_20100915oss,devdriver_3.2。测试用图为 Lena 256×256 的三波段无噪声彩色图像。在分割参数(hs,hr,M)分别取(5,7,70)、(5,7,100)、(7,12,70)、(7,12,100)时 TBB 和 CUDA 的分割结果如图 3 所示(串行的结果如图 2 所示。)

(a) (hs,hr,M) = (5,7,70) (b) (hs,hr,M) = (5,7,100) (c) (hs,hr,M) = (7,12,70) (d) (hs,hr,M) = (7,12,100)

图 3 基于 TBB 和 CUDA 的 Mean Shift 分割结果

Fig. 3 Segmentation result of Mean Shift based on TBB and CUDA

从图 3 的显示效果可以看出,增加 M 值图像的分块数会减少,这是因为 M 值越大,图像分割后每一块的最小值越大。基于 TBB 并行化将要分割的图像数据的每一个像素点同时进行计算,计算的并行度为处理器的数量,与串行计算的硬件平台相同,Mean Shift 算法的 TBB 并行化分割的结果与串行结果完全一致。基于 CUDA 的并行化分割结果与在 CPU 上分割结果相比有一定的误差,CUDA 的结果根据硬件和迭代阈值的不一样会略差,并且误差随着带宽参数的增加而增加。使用 JS(Jaccard similarity)方法对分割结果进行定量分析。JS 值在 0 与 1 之间,比值越接近 1,说明分割结果越接近标准结果,分割精度越高。基于 TBB 并行化分割结果的 JS 值为 1,基于 CUDA 的并行化分割结果在迭代次数小于 100,两次迭代差值小于 0.01 时的 JS 值为 1,迭代次数大于 100 时,JS 值会降低。基于 CUDA 的误差是由于 GPU 的计算精度低于 CPU,首先由于精度低,计算的结果精度也低,此外 GPU 的精度低导致在给定的迭代次数内会有更多的像素点(与 CPU 相比)没有漂移到稳定点。

算法加速比如表 2 所示。

在分割参数为(5,7,100)时,大小分别为 64×64、256×256、1 024×1 024 的图像在 TBB 和 CUDA 的环境下的加速比如图 4 所示。

图 4 中 TBB 加速比显示不明显,图 5 给出了 4 核系统上 TBB 加速比在不同分割参数下随图像大小变化的曲线。

实验结论:1)Mean Shift 算法的执行时间(并行和串行)与 M 值的大小无关;2)Mean Shift 算法的执行时间(并行和串行)随着 hs、hr 的增加而增加;

表 2　图像的加速比

Table 2　Images speedup

图像大小	分割参数	串行时间/ms	TBB 时间/ms	TBB 加速比	CUDA 时间/ms	CUDA 加速比
64 ×64	(5,7,70)	282	110	2. 56	109	2. 59
	(5,7,100)	282	94	3. 00	94	3. 00
	(7,12,70)	656	235	2. 80	125	5. 25
	(7,12,100)	672	188	3. 57	125	5. 38
256 ×256	(5,7,70)	3 562	937	3. 80	187	19. 06
	(5,7,100)	3 563	937	3. 80	187	19. 06
	(7,12,70)	9 031	2 281	3. 96	250	36. 12
	(7,12,100)	9 032	2 343	3. 85	234	38. 60
1 024 ×1 024	(5,7,70)	55 969	12 984	4. 31	1516	36. 92
	(5,7,100)	55 984	13 812	4. 05	1578	35. 48
	(7,12,70)	141 484	34 069	4. 15	2562	55. 22
	(7,12,100)	141 500	34 625	4. 09	2562	55. 23

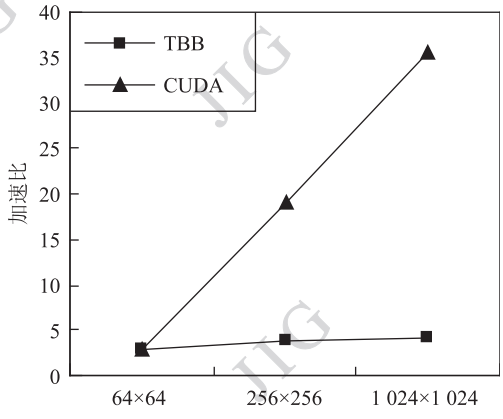


图 4　TBB 和 CUDA 的加速比

Fig. 4　Speedup of TBB and CUDA

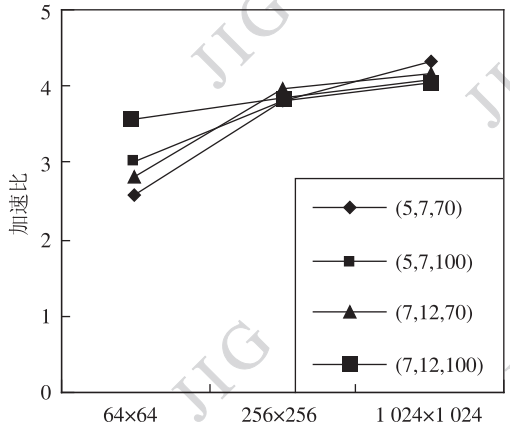


图 5　TBB 下图像的并行加速比

Fig. 5　Speedup of TBB

3)Mean Shift 算法的执行时间(并行和串行)随着图像增大而增加;4)Mean Shift 算法基于 CUDA 并行化加速比随着图像增大而增加;5)Mean Shift 算法基于 TBB 并行化加速比随着图像增大而增加;6)Mean Shift 算法基于 CUDA 并行化的加速比随着带宽参数的增加而大幅度增加;7)Mean Shift 算法基于 TBB 并行化的加速比随着带宽参数的增加而缓慢增加,稳定趋于核数;8)一般情况下,CUDA 比 TBB 的(分别基于同价格的 GPU 和 CPU)加速比高。

5. 2　结果分析

产生 5. 1 节中实验结论 1)和 2)的原因是 Mean Shift 图像分割算法的迭代过程与 M 无关,只与 hs、hr 有关,hs 决定了迭代窗口的大小,hr 决定了在窗口内参与计算 Mean Shift 向量的点像素的个数,参加计算的点越多,计算时间越长;产生实验结论 3)的原因是对图像中的每一个点都要进行迭代,迭代的点增加(图像增大)了,时间相应增加;产生实验结论 4)的原因是在图像较小的时候,不能满足让所有 GPU 的物理线程都处于运算状态,随着图像的增大,参与运算的线程增加,所以加速比增大,但是当增加到一定程度时,加速比不会继续增加;产生实验结论 5)的原因是由于 4 核 CPU 的计算机在图像较小时,相当于细粒度的并行方式,粒度太细,每一个 CPU 运算的时间过短,不能有效地隐藏延迟;产生实验结论 6)的原因是随着带宽增加,迭代内运算的

中国图象图形学报
JOURNAL OF IMAGE AND GRAPHICS
Vol. 18, No. 12, Dec. 2013

总次数增加,由于 CUDA 的高性能,其加速比会增加,另外在带宽增加时,每次迭代时参与迭代的像素点更多,但会有精度的损失;产生实验结论 7)的原因是随着带宽增加而每次迭代时参与计算的像素点增多,则并行时粒度增大,所以加速比开始增加,但是增加的幅度与 CUDA 相比较是较缓慢,一是由于 CPU 的核心数较少,二是由于 TBB 方式在一开始就达到了一定的并行粒度,加速比随图像增加到一定程度时,不再增加;产生实验结论 8)的原因是 GPU 专注于运算,更多的芯片用于数据运算处理,而 CPU 还要负责其他的控制、逻辑操作。

此外,从表 2 的 1 024 ×1 024 图像大小中可以看到基于 TBB 的方式的加速比略大于处理器的核心数量,该结果可能由 CPU 计时的误差以及并行划分之后每次计算的计算规模减少带来的加速效果。

6 结 论

分别用 TBB 和 CUDA 实现了对 Mean Shift 聚类算法的并行化改造,并分析了其加速比和分割的效果。在 4 核 CPU 的计算机上取得了 4 倍的加速比,基于 CUDA 最高可得到 55 倍的加速比,选择使用 TBB 和 CUDA 进行并行化是因为它们可以充分利用现有的硬件资源,使用 CUDA 可以取得很高的加速比,但会有一定的误差使用 TBB 由于 CPU 核心数量的限制加速比不是很高,但其分割效果好,与串行的结果一致,所以在选择使用时,要根据实际情况在加速比和分割效果上寻求平衡,为了能够既保证有很高的加速比又有很好的分割效果,需要进一步研究将两种方法进行混合,保证算法的效率和精度。

参考文献(References)

[1] Fukunaga K, Hostetler L D. The estimation of the gradient of a density function with applications in pattern recognition[J]. IEEE Trans. on Information Theory, 1975, 21(1): 32-40.

[2] Cheng Y Z. Mean Shift, mode seeking, and clustering[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1995, 17(8): 790-799.

[3] Comaniciu D, Meer P. Mean Shift: a robust approach toward feature space analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 24(5): 603-619.

[4] Comaniciu D, Meer P. Mean shift analysis and applications[C]//Proceedings of IEEE 7th International Conference on Computer Vision. Washington: IEEE Press, 1999, 2(2): 1197-1203.

[5] Comaniciu D, Meer P. Robust analysis of feature spaces: color image segmentation[C]//Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington: IEEE Press, 2002, 1: 750-755.

[6] Liao J Y, Guo S Y, Huang X X. A maximum entropy segmentation method based on mean shift clustering[J]. Computer Simulation, 2009, 26(9): 187-190. [廖建勇, 郭斯羽, 黄梓效. 基于 Mean Shift 聚类的最大熵图像分割方法[J]. 计算机仿真, 2009, 26(9): 187-190.]

[7] Ben Z W, Zhao X J. Meaningful region segmentation based on Mean Shift algorithm[J]. Laser & Infrared, 2009, 39(9): 1004-1008. [贲志伟, 赵勋杰. 基于 Mean Shift 算法提取彩色图像有意义区域[J]. 激光与红外, 2009, 39(9): 1004-1008]

[8] Yi H F, Huang X W. Color image segmentation algorithm based on mean shift[J]. Computer Applications, 2006, 26(7): 1605-1607. [伊怀锋, 黄贤武. 基于均值偏移的彩色图像分割算法[J]. 计算机应用. 2006, 26(7): 1605-1607.]

[9] Guo Q C, Cai Q, Guo S Y. Research of image segmentation based on mean-shift[J]. Computer Simulation, 2010, 27(2): 231-235. [郭庆昌, 蔡蒨, 郭盛雨. 基于均值移动算法的图像分割的研究[J]. 计算机仿真, 2010, 27(2): 231-235.]

[10] Li H, Zhang M X, Zhen J L. Color image segmentation based on mean shift and multi feature fusion[J]. Journal of Computer Applications, 2009, 29(8): 2074-2076. [李华, 张明新, 郑金龙. 融合多特征的均值漂移彩色图像分割方法[J]. 计算机应用, 2009, 29(8): 2074-2076.]

[11] Pan H Y. Color image segmentation method combining mean shift and region merging[J]. Computer Engineering and Applications, 2009, 45(22): 156-158. [潘红艳. 融合均值漂移和区域合并的彩色图像分割方法[J]. 计算机工程与应用, 2009, 45(22): 156-159.]

[12] Zuo J Y, Liang Y, Zhao C H, et al. Researches on scale adaptation strategy in mean shift tracking algorithm[J]. Journal of Image and Graphics, 2008, 13(9): 1750-1757. [左军毅, 梁彦, 赵春晖, 等. Mean Shift 跟踪算法中尺度自适应策略的研究[J]. 中国图象图形学报, 2008, 13(9): 1750-1757.]

[13] Tang Y, Pan Z G, Tang M, et al. Image segmentation with hierarchical mean shift[J]. Journal of Computer Research and Development, 2009, 46(9): 1424-1431. [汤杨, 潘志庚, 汤敏, 等. 基于分级 mean shift 的图像分割算法[J]. 计算机研究与发展, 2009, 46(9): 1424-1431.]

[14] Li Y L, Shen Y. Fast mean shift for image segmentation based on conjugate gradient[J]. Opto-Electronic Engineering, 2009, 36(8): 94-99. [李艳灵, 沈轶. 基于共轭梯度法的快速 Mean Shift 图像分割[J]. 光电工程, 2009, 36(8): 94-99.]

[15] Li G, Li Y L. Research on fast mean shift algorithm for image segmentation[J]. Mathematics In Practice And Theory, 2009, 39(8): 173-177. [李刚, 李艳灵. 快速均值漂移图像分割算法研究[J]. 数学的实践与认识, 2009, 39(8): 173-177.]

[16] Sun X W, Li Y J, Chen Y. Fast mean shift algorithm in image

segmentation[J]. Measurement and Control Technology, 2008, 27(7):23-26. [孙小炜,李言俊,陈义. Mean Shift 图像分割的快速算法[J]. 测控技术,2008,27(7):23-26.]

[17] Shen Z F, Luo J C, Wu W, et al. Implementation of parallelization of mean-shift algorithm for multi-scale segmentation of remote sensing images[J]. Journal of Harbin Institute of Technology, 2010,42(5):811-815. [沈占锋,骆剑承,吴炜,等. 遥感影像均值漂移分割算法的并行化实现[J]. 哈尔滨工业大学学报,2010,42(5):811-815.]

[18] Wu W, Shen Z F, Luo J C, et al. Implementation of Multi-scales segmentation for high resolution RS images based on cluster[J]. Computer Engineering and Applications, 2009, 45(34): 7-10. [吴炜,沈占锋;骆剑承,等. 均值漂移高分辨率遥感影像多尺度分割的集群实现[J]. 计算机工程与应用,2009,45(34):7-10.]

[19] Sun X G, Li M C, Liu Y X, et al. Accelerated segmentation approach with CUDA for high spatial resolution remotely sensed imagery based on improved mean shift[C]//Proceedings of Urban Remote Sensing Joint Event. Shanghai: IEEE, 2009, 65-70.

[20] Hu B, Yuan D H. TBB multi-core programming and research on its hybrid paradigms[J]. Computer Technology and Development, 2009, 19(2):98-102. [胡斌,袁道华. TBB 多核编程及其混合编程模型的研究[J]. 计算机技术与发展,2009,19(2):98-102.]

[21] Janes reinders, Nie X J translation. Programming Guide Of Intel Threading Building Blocks[M]. Beijing: Machinery Industry Press, 2009. [Janes reinders 著, 聂雪军译, Intel Threading Building Blocks 编程指南[M]. 北京:机械工业出版社,2009.]

[22] Zhang S, Chu Y L. GPU high-performance computing by CUDA [M]. Beijing: China Water Power Press, 2009. [张舒,褚艳利. GPU 高性能运算之 CUDA[M]. 北京:中国水利水电出版社,2009.]