第 29 卷 第 4 期
2014 年 8 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol. 29 No. 4
Aug. 2014

DOI:10.13364/j.issn.1672-6510.2014.04.011

面向 CPU + GPU 异构平台的模板匹配目标识别并行算法

马永军[1]，袁 赢[1]，李 灏[2]
（1. 天津科技大学计算机科学与信息工程学院，天津 300222；2. 天津瑞和天孚科技有限公司，天津 300384）

摘 要：针对大数据量导致模板匹配目标识别算法计算时间长，难以满足快速检测的实际需求问题，在采用最新 NVIDIA Tesla GPU 构建的 CPU + GPU 异构平台上，设计了一种模板匹配目标识别并行算法. 通过对模板图像数据常量化、输入图像数据极致流多处理器片上化和简化定位参数计算 3 方面优化了并行算法，并对算法进行性能测试. 实验表明，该算法在保证识别效果的同时实时性明显提高.

关键词：模板匹配；目标识别；并行计算；统一设备计算架构；图形处理器
中图分类号：TP311 **文献标志码**：A **文章编号**：1672-6510(2014)04-0048-05

Parallel Algorithm of CPU and GPU-oriented Heterogeneous Computation in Template Matching Target Recognition

MA Yongjun[1], YUAN Ying[1], LI Hao[2]
(1. College of Computer Science and Information Engineering, Tianjin University of Science & Technology, Tianjin 300222, China; 2. Tianjin Ruihe Tianfu Science & Technology Ltd.Co., Tianjin 300384, China)

Abstract: Moving object recognition algorithm with high-definition video data suffers from large computation complexities and slow speed. With NVIDIA Tesla K20 c GPU, a method of accelerating the template matching target tracking algorithm with the heterogeneous system integrated with CPU and GPU was proposed. The parallel algorithm was designed by three optimizing means: constant memory, the internal memory of SMX and the brief calculation of correlation coefficient. Finally, the program was coded on compute unified device architecture and tested. The results show that the designed algorithm can obviously improve the real-time performance of the algorithm and guarantee the recognition effect.

Key words: template matching; target recognition; parallel computing; compute unified device architecture (CUDA); graphic processing unit (GPU)

模板匹配运动目标识别已广泛应用到视频监控等领域，然而随着视频信息高清化、数字化发展，在视频处理领域中基于模板匹配的运动目标识别算法的计算代价高，实时性降低，因而迫切需要提高计算能力. 目前在此领域已取得了一些成果：于志华等[1]提出一种基于 FPGA 的并行超标量匹配处理架构加速算法，但仅局限于语音系统；李俊山等[2]提出基于 K 元 2–立方体网络结构的图像模板匹配并行算法，适用于 LS MPP(Li-Shan massive parallel processing)计算机. 另外，赵东明等[3]利用 DNA 计算的并行性实现模板匹配算法的并行化；马永军等[4]利用多核平台使用 OpenMP 对模板匹配进行加速. 这 2 种算法均基于 CPU 实现，而 CPU 并不擅长并行计算，无法解决在高清环境下视频数据量大、实时处理性能不佳的问题. Rajasekaran[5]通过 FFT 技术获得有效的模板匹配并行算法，但存在算法计算量巨大的问题；Anderson 等[6]通过 GPU 对模板匹配进行加速，但没有充分挖掘异构平台的优势，仍有提升空间. 因此，提高基于模板匹配目标识别算法性能具有现实意义.

NVIDIA 公司最新的 Tesla 系列图形处理单元(GPU)完全针对并行计算而设计，利用 NVDIA 公司统一设备计算架构(compute unified device architec-

收稿日期：2014–02–24；**修回日期**：2014–05–04
基金项目：天津市科技支撑计划重点资助项目(12ZCZDGX02400)
作者简介：马永军（1970—），男，吉林长春人，教授，yjma@tust.edu.cn.

ture，CUDA）的高可扩展性，完全可以满足高性能计算机的需要. 而且，由 CPU + GPU 组成的异构处理平台相比传统的同构多处理机系统，能够提供更好的计算性能和功耗效率，已经成为解决大数据量引发的计算复杂、代价高等问题的主流并行解决方案. 其中，如何高效地利用异构系统中 CPU 和 GPU 进行协同并行计算是目前研究的热点问题[7].

本文提出一种采用最新的 NVIDIA Tesla GPU 构建 CPU + GPU 异构平台，并进行 CUDA 并行编程的方法，利用 CPU + GPU 异构多核的高并行能力，使用最新的 NVIDIA Tesla K20c GPU 对模版匹配目标识别并行算法进行设计.

1　GPU 和 CUDA 编程介绍

Tesla GPU 凭借开普勒（Kepler）架构可以提供强大的处理功能以及优化，并具有提高 GPU 上并行执行工作负载的新方法. Kepler 架构 GPU 的核心是极致流多处理器（next generation streaming multiprocessor，SMX）[8]，与上一代 Fermi 架构使用的流多处理器（streaming multiprocessor，SM）相比，计算和功能单元的数量、安置方式有了很大变化. 这一新型的流式多处理器设计让应用到处理核心上的空间比例远高于应用到控制逻辑单元上的空间比例，从而可实现更高的处理性能和效率.

CUDA 核心包含线程组层次、共享存储器和栅栏同步 3 个重点抽象，用于引导程序员将问题划分为可以被多个块内线程独立并行处理的细粒度子问题；然后，每个子问题在 1 片 SMX 上被 1 个块内线程并行协作处理，以提高执行效率.

由于 CUDA 并未封装存储器系统的异构性，用户可以针对具体存储器的特点，有选择地使用[9]. CUDA 将优化程序的自主权交给了用户，可提高用户自主解决问题的灵活性.

2　模板匹配目标识别的 GPU 并行算法设计

2.1　滑动模板匹配目标识别算法分析

滑动模板匹配算法的匹配过程如图 1 所示. 模板匹配目标识别算法的基本原理是让包含目标的模板图像 T 在输入图像 I 中滑动，对模板图像和其覆盖的对应输入图像区域利用去均值归一化相关系数公式（式（1））[10]进行定位参数计算，选择相关系数最优值所在位置作为最佳目标匹配位置.

$$\gamma(s,t)=\frac{\sum_x\sum_y\left[I(x,y)-\bar{I}(x,y)\right]\left[T(x-s,y-t)-\bar{T}\right]}{\sqrt{\sum_x\sum_y\left[I(x,y)-\bar{I}(x,y)\right]^2\sum_x\sum_y\left[T(x-s,y-t)-\bar{T}\right]^2}} \tag{1}$$

由式（1）可以看出，在滑动模板匹配过程中，每次仅计算 1 个像素点位置的定位参数，按照匹配顺序依次计算每个像素点的定位参数，完成整幅输入图像匹配的过程需要大量简单、计算方式相同的重复计算. 因为在滑动匹配过程中，各个像素点的计算过程是相互独立、互不干扰的，所以适合通过大规模的线程并发执行来加速计算.

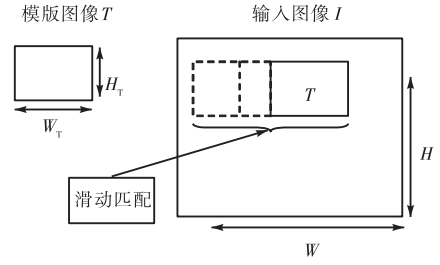


图 1　滑动模板匹配算法的匹配过程

Fig. 1　Matching procedure of sliding template matching algorithm

2.2　模板匹配并行算法总体设计

为了通过 GPU 执行大量并发线程，发挥高性能并行计算优势，实现减少模板匹配过程时间的目的，将图像区域模板匹配的并行算法按照“先整体区域分块，再局部并行计算”的策略分为 2 步实现（图 2）：第 1 步，在 GPU 中将图像区域对应的 CUDA 线程分块（Block），进行图像数据分配；第 2 步，在每个 Block 中对每个 CUDA 线程对应的像素点进行定位参数计算，存储局部统计结果，然后汇总局部结果，求整体极值得到整幅输入图像中的最佳匹配位置.

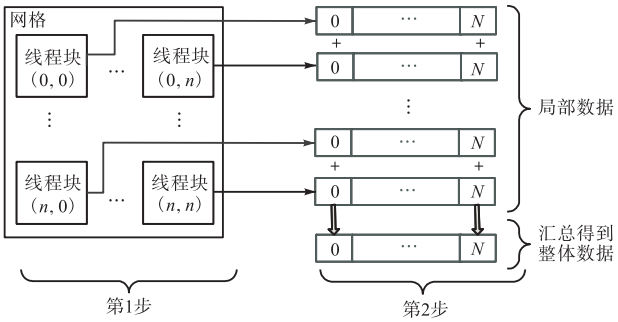


图 2　模板匹配并行算法总体设计

Fig. 2　Overall design of parallel sliding template matching algorithm

根据滑动匹配的过程，将每个像素点位置处的匹配过程映射到 CUDA 线程上，1 个线程处理与之对应的 1 个像素点的匹配过程. 其中，并行化过程中需要在满足线程映射关系的前提下进行线程关键参数计算，程序如下：

```
//网格宽度分配
gridDim.x=⌊(Width+blockDim.x-1)/blockDim.x⌋;
//网格高度分配
gridDim.y=⌊(Height+blockDim.y-1)/blockDim.y⌋;
//线程在网格中的横坐标
tid_in_grid_x=threadIdx.x+blockIdx.x*blockDim.x;
//线程在网格中的纵坐标
tid_in_grid_y=threadIdx.y+blockIdx.y*blockDim.y;
//线程在整个区域中的位置
tid_in_grid=tid_in_grid_x+tid_in_grid_y*Width;
//有效线程过滤
if(tid_in_x>Width||tid_in_grid_y>Height) return;
```

2.3　并行算法的优化

根据 GPU 硬件特性可知，GPU 存储系统中各存储单元为层次结构，其中包括：常量存储器、共享存储器、纹理存储器、全局存储器等，它们的作用域和访问效率各不相同[9]. 基于 GPU 的编程框架并未封装存储器系统的异构性，在总体设计的第 1 步中，针对模板图像数据在计算过程保持不变的特点，选用常量存储器存储模板图像数据，进行常数化参数存储优化；针对存储在纹理存储器中的输入图像数据访问延迟大的特点，选用 SMX 片上共享存储器存储局部数据，进行 SMX 片上数据存储优化，从而提高算法访问图像数据和存储计算结果的效率.

另外，在总体设计第 2 步中，可以通过简化定位参数的计算过程降低并行化过程中的计算量，进一步提高计算效率.

2.3.1　模板图像参数常数化

针对模板图像的参数在匹配操作的相关系数计算过程中始终保持不变的特点，算法中将与模板图像相关的参数提前计算出来存储于 GPU 的常量存储器中，从而实现模板图像参数常数化存储. 然后，在定位参数计算过程中可直接读取存储器中的参数. 虽然常量存储器仅有 64 KB[8]，但是访存速度要比设备存储器快，通常在 1 ~ 100 个时钟周期内就可以获取所需要的数据.

假设输入图像为 W 像素 × H 像素，在计算任意像素点定位参数时，模板图像的参数都会被计算 1 次，整个图像则会被重复计算 WH 次. 通过利用常量存储器，计算次数为原来的 $1/WH$，在模板图像参数获取上的性能提高了 WH 倍. 具体过程见图 3.

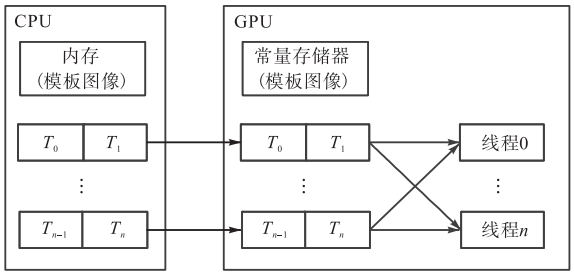


图 3　模板图像数据复用原理

Fig. 3　Reuse of template image data

2.3.2　输入图像数据 SMX 片上化

用于存储输入图像数据的 GPU 纹理存储器属于 SMX 片外存储器，通常有几百个时钟周期的访问延迟. 因此，如何缩短纹理存储器的访问时间是加速的关键.

存储器层次结构如图 4 所示. 其中 SMX 片上共享存储器的访问速度比纹理存储器快10倍左右[9]. 因此，在计算过程中把在本线程块中使用到的局部输入图像的数据存储在 SMX 片上共享存储器中，然后用 SMX 片上共享存储器中的局部原始输入图像与常量存储器中的模板图像数据进行定位参数计算. 通过充分利用 SMX 片上共享存储器解决纹理存储器访问效率低下的问题.

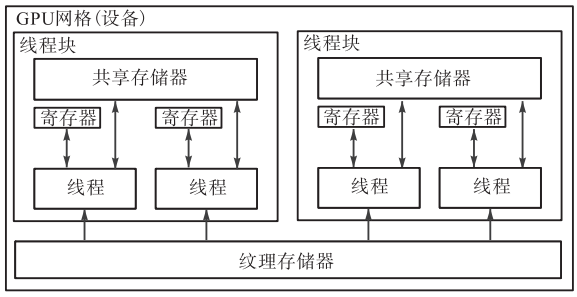


图 4　存储器层次结构图

Fig. 4　Memory hierarchy

在串行算法中，图像数据在主机内存中始终只有 1 份，而在 CPU + GPU 异构平台下，需要拷贝纹理存储器中的图像数据到 SMX 片上共享存储器中保留第 2 份副本. 这样做虽然使用了更多的 GPU 存储空间，但是减少了运算时间，可取得更好的算法性能.

2.3.3　简化定位参数计算

在计算定位参数时，如果模板图像包含 WH 个

像素点，则可以使用 $\frac{1}{WH}\sum_x\sum_y\left[I(x,y)\right]$ 来代替 $\bar{I}$，代入式(1)可得简化之后的计算公式为

$$\gamma(s,t)=\frac{WH\sum IT-\sum I\sum T}{\sqrt{\left(WH\sum I^2-\left(\sum I\right)^2\right)\left(WH\sum T^2-\left(\sum T\right)^2\right)}} \tag{2}$$

根据式(2)，在优化前对每个像素点定位参数计算过程中，需要计算 $\sum I$、$\sum T$、$\sum IT$、$\sum I^2$、$\sum T^2$ 共 5 个参数. 为了减少计算次数、节约时间，可以提取算法中的部分公共因子进行预计算并存储计算结果，需要时代入. 因为式(2)中与模板图像相关的参数 $\sum T$ 和 $\sum T^2$、表达式 $(WH)\sum T^2-\left(\sum T\right)^2$ 可以在匹配过程开始之前计算，所以每次计算定位参数时只需要计算 $\sum I$、$\sum I^2$ 和 $\sum IT$. 此优化使得定位参数计算过程中的计算量减少了 2/5.

而且，由于 NVIDIA Tesla K20c GPU 中的极致流多处理器(SMX)包含用于整数乘法和加法的硬件结构，因此能够在 GPU 上快速实现该函数的求值.

2.4　模板匹配目标识别并行算法总体流程

根据模板图像数据常数化、输入图像数据 SMX 片上化和简化定位参数计算 3 方面的优化，在 Tesla GPU 的核心上建立大量线程，将像素点与 CUDA 线程一一对应，并为每个像素的计算开辟空间，利用 GPU 将多个像素计算的时间缩短成和 1 个像素点计算的时间相似，从而提高算法执行效率. 基于 GPU 加速的模板匹配目标识别并行算法流程见图 5.

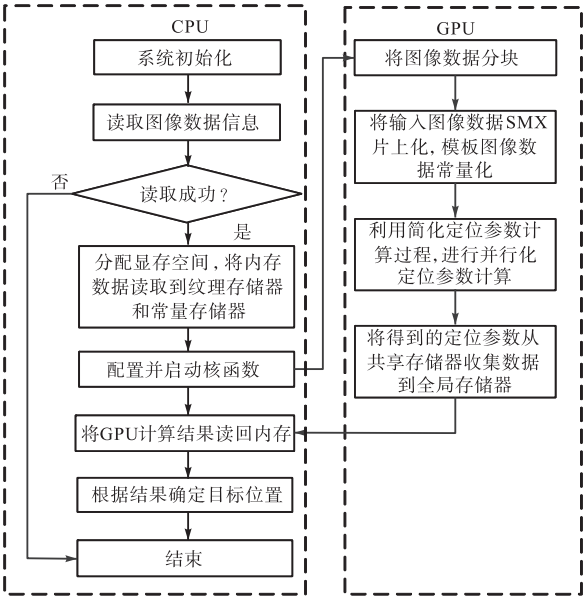


图 5　模板匹配目标识别并行算法流程

Fig. 5　Float chart of template matching target recognition parallel algorithm

3　实　验

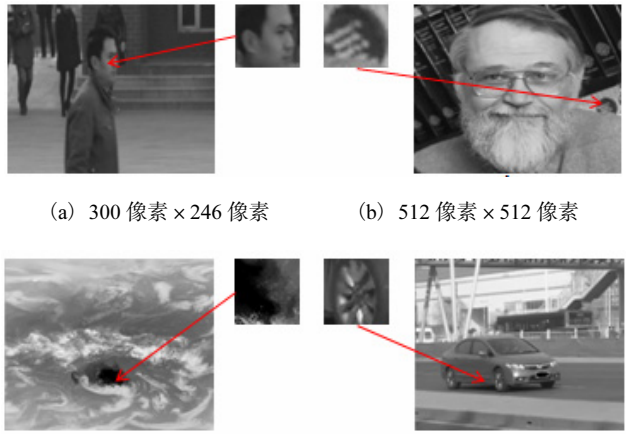
3.1　实验环境

硬件平台：超云服务器，GPU 为 Tesla K20c，含有 13 颗极致流多处理器，每个极致流多处理器上有 192 颗 CUDA 核心，其核心频率为 706 MHz，显存带宽为 320 bit，显存为 4 GB；CPU 为 Intel Xeon CPU 3.1 GHz；系统内存为 16 GB.

软件平台：操作系统为 Microsoft Windows 7 中文专业版；集成开发平台为 Visual Studio C++ 2010；并行开发环境包括 CUDA 5.0，CUDA Toolkit 5.0；计算机视觉库采用 OpenCV 2.4.3.

3.2　实验结果与分析

为了验证算法的并行加速效果，采用固定模板图像大小为 52 像素 × 52 像素、变化输入图像大小的 4 组数据(图 6)，对比基于 CPU 的串行模板匹配目标识别算法(方法 1)、基于 GPU 的未优化并行模板匹配目标识别算法(方法 2)、基于 GPU 的优化并行模板匹配目标识别算法(方法 3)的执行效果. 为了减小实验误差，算法分别运行 50 次，记录平均运行时间.



(a) 300 像素 × 246 像素　　(b) 512 像素 × 512 像素

(c) 560 像素 × 560 像素　　(d) 768 像素 × 576 像素

图 6　输入图像和模板图像

Fig. 6　Input image and template image

3 种方法的实验对比见表 1. 分析表 1 可知：(1)对同一幅图像进行特征点检测时，方法 3(基于 GPU 的优化并行模板匹配目标识别算法)与方法 1(基于 CPU 的串行模板匹配目标识别算法)和方法 2(基于 GPU 的未优化并行模板匹配目标识别算法)相比，极大地提高了算法的实时性，能够满足实际需求. (2)对于本实验，虽然将串行算法(方法 1)翻译成在 GPU 上运行的并行程序(方法 2)提高了算法效率，但是优

化后的算法(方法 3)能够取得更好的算法实时性. 可见,并行算法的设计和开发需要根据算法本身和所使用的 GPU 硬件特性,有针对性的优化之后才能更好地发挥 GPU 的性能,获得满意的加速效果.

表 1　串行算法、并行算法、优化并行算法的平均运行时间

Tab. 1　Average processing time of serial algorithm, parallel algorithm and optimal parallel algorithm

输入图像	图像大小/像素	平均运行时间/ms		
		方法 1	方法 2	方法 3
图 6(a)	300 × 246	3 027.0	372.3	54.0
图 6(b)	512 × 512	9 843.0	1 123.8	171.8
图 6(c)	560 × 560	11 934.0	1 345.9	213.8
图 6(d)	768 × 576	16 895.0	1 893.7	289.6

注:方法 1、方法 2、方法 3 分别为串行算法、并行算法、优化并行算法.

4　结　语

本文在构建的 CPU + GPU 异构平台基础上设计了模板匹配目标识别并行算法,给出了实现的步骤和性能优化的方法,并在 NVIDIA Tesla K20 c GPU 上进行性能测试. 结果表明,在 Kepler 架构下利用 CUDA 编程模型进行程序并行化处理和优化可明显提高算法的实时性,能够适应现在视频高清化、数字化、网络实时性等方面的要求.

参考文献:

[1] 于志华,张兴明,杨镇西,等. 一种高性能固定语音识别并行处理架构[J]. 计算机应用研究,2013,30(8):2419–2421.

[2] 李俊山,沈绪榜. K 元 2–立方体网络 SIMD 计算机图像模板匹配并行算法[J]. 计算机学报,2001,24(11):1296–1301.

[3] 赵东明,罗亮. 基于 DNA 计算的图像模板匹配算法[J]. 华中科技大学学报:自然科学版,2013,41(2):97–101.

[4] 马永军,吴文旭,何锵锵. 一种利用多核计算的目标检测算法[C]//Proceedings of 2010 International Conference on Services Science, Management and Engineering. Hongkong: IITA, 2010: 209–212.

[5] Rajasekaran S. Efficient parallel algorithms for template matching[J]. Parallel Processing Letters, 2002, 12(3/4): 359–364.

[6] Anderson R F, Kirtzic J S, Daescu O. Applying parallel design techniques to template matching with GPUs[M]. High Performance Computing for Computational Science: VECPAR 2010. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011: 456–468

[7] 张保,董小社,白秀秀,等. CPU-GPU 系统中基于剖分的全局性能优化方法[J]. 西安交通大学学报,2012,46(2):17–23.

[8] NVIDIA. NVIDIA CUDA C Programming Guide: Version 5.0[EB/OL].[2012–10–01]. http://www.nvidia.cn/object/maintenance-cudazone-cn.html.

[9] 仇德元. GPGPU 编程技术:从 GLSL, CUDA 到 OpenGL[M]. 北京:机械工业出版社,2011.

[10] Yoo J C, Han T H. Fast normalized cross-correlation [J]. Circuits, Systems and Signal Processing, 2009, 28(6): 819–843.

责任编辑: 常涛