中图法分类号：TP391　　**文献标识码**：A　　**文章编号**：1006-8961(2012) - -
论文引用格式:

非正交二值子空间的模板表示并行生成算法

杨培，武港山，杨扬，任桐炜

南京大学 计算机软件新技术国家重点实验室，江苏 南京，210046

摘　要：非正交二值子空间（Non-orthogonal binary subspace）是一种重要的图像模板表示方法，能够高效地进行模板匹配。针对其生成过程耗时多难以实时应用的问题，提出了一种基于统一计算设备架构（CUDA）的非正交二值子空间模板表示的快速并行生成算法。该算法将模板生成过程划分为三个步骤，并根据每个步骤数据处理的特点，进行并行任务分解，极大地缩短了生成时间。批量处理实验表明，相比基于CPU生成相同的模板表示，算法的速度提升了60~120倍。在应用于模板匹配时，该并行的模板表示生成算法能够很好的改进匹配效率。

关键词 ：非正交二值子空间；哈尔基；匹配追踪；模板表示；模板匹配；统一计算设备架构

A parallel algorithm for generating template representation based on Non-orthogonal Binary Subspace

Yang Pei, Wu Gangshan, Yang Yang, Ren Tongwei

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing Jiangsu 210046, China

Abstract: NBS (Non-orthogonal Binary Subspace) is an essential image representation method, whose major advantage is to support high-efficiency image matching. However, generating NBS is itself time-consuming which makes the method difficult to be applied in real-time applications. In this paper, a parallel template-generating algorithm based on CUDA (Compute Unified Device Architecture) is proposed. The generating procedures are divided into 3 phases in our approach. For each phase, paralleled task distribution is used to fully utilize the capability of GPU. Experimental results demonstrate that our algorithm achieves 60-120 times speed-up, compared to the same template generating task on CPU. Additionally, we apply our algorithm on the template matching problem, and observe significant improvement in efficiency.

Key words: non-orthogonal binary subspace; haar base; matching pursuit; template representation; template matching; compute unified device architecture (CUDA)

0 引　言

模板匹配[1-3]是计算机视觉领域的关键问题之一，它已广泛应用于跟踪、识别和许多其他应用。非正交二值子空间(Non-orthogonal binary subspace，以下简称NBS)模板表示方法[3-7]是一种有别于传统方法并且能较快地进行图像多尺度匹配的模板表示方法。它主要思路是将模板表示成由较少数量的哈尔(Haar)二值特征基组成的一个线

基金项目:国家自然科学基金项目(61021062)；南京大学计算机软件新技术国家重点实验室自主课题(ZZKT2011B04)；国家高技术研究发展计划(863)项目(2011AA01A202)。
收稿日期:　　；　**改回日期**:
第一作者简介：杨培(1986~),男,南京大学计算机科学与技术系硕士研究生。主要研究方向为视频图像处理。E-mail: yypp_918@163.com。
通讯作者:武港山，E-mail: gswu@nju.edu.cn。

性组合。在进行模板匹配时，用这个哈尔基的线性组合替代模板来进行处理。这种模板表示方法用于图像模板比对时，借助预先计算的积分图像[8]，可用几次简单的加法运算替代像素点之间的大部分浮点乘法，从而大大提高了比对速度。NBS 模板表示方法比较简单并且适应性强，因此它在图像领域有很好的应用前景。但是，在结合 NBS 模板表示进行匹配前，必须先找到模板对应的哈尔基的线性组合，由于模板对应的候选哈尔基的数目过于庞大，找到合适的哈尔基的线性组合是很耗时的，这一方面使得 NBS 模板表示的生成不能进行实时处理，另一方面在进行批量处理的时候，它的效率也是极低的。在文献[5]中，作者提出了用匹配追踪算法(Matching Pursuit，以下简称 MP[9])替代优化的正交匹配追踪算法 (Optimized Orthogonal Matching Pursuit，以下简称 OOMP[10])，以提高生成速度达到实时处理的目的，但是在大批量处理图像数据时，却还是会耗费大量时间。

针对 NBS 模板表示生成耗时的缺点，提出一种基于 NVIDIA 的统一计算设备架构(Compute Unified Device Architecture, 以下简称 CUDA)的快速的 NBS 模板表示并行生成算法。计算设备选择图形处理单元(Graphic Processing Unit，以下简称 GPU)，一方面是能够减少 CPU 的负担，让 CPU 能进行其余更多的操作，提高工作效率，另外一方面也是由于 GPU 本身特别适合处理图像，在图像处理方面有很大的优势。分析了 NBS 模板表示生成的可并行性，并进一步分解该算法提出了快速的 NBS 模板表示并行生成算法。实验表明，该算法能够大幅度提高 NBS 模板表示的生成效率，平均耗时至少能缩减到三十分之一左右。实验中与基于 CPU 的 NBS 模板表示生成算法进行了对比，并对算法的实现细节和实验结果进行了分析。

文章其余部分组织如下：第二部分介绍了 NBS 模板表示的生成算法及其应用背景，第三部分讨论了 NBS 模板表示的生成可并行性，第四部分阐述了基于 CUDA 平台的 NBS 模板表示的并行生成架构，第五部分，给出了实验结果以及算法的进一步分析，第六部分总结。

1 NBS 图像模板表示

1.1 图像模板的哈尔基表示

最近，哈尔基特征[11-12]渐渐成为一种比较流行的图像特征。如公式(1)是哈尔基α形式化的定义 。

$$b(u,v)=\begin{cases}1, u_0 \le u \le u_0+w-1, v_0 \le v \le v_0+h-1\\ 0, \text{其他}\end{cases} \tag{1}$$

式中 u_0，v_0 是矩形框的左上角坐标，w，h 分别为宽和高，$b(u,v)$即为哈尔基对应的二值函数。

对于一个模板图块 I，可以用哈尔基线性表示为：$I = c_1\alpha_1 + c_2\alpha_2 + c_3\alpha_3 + c_4\alpha_4 + \cdots$，式中$\alpha_i$表示一个哈尔基，$c_i$是其对应的系数，如图 1 所示。

图1 模板的二值哈尔基的线性表示

Fig.1 Linear representation of template by binary Haar-bases

1.2 NBS 图像模板表示的生成

生成 NBS 图像模板表示就是要提取出一组合适的哈尔基(假设选出的哈尔基数量为 K)及其对应系数来描述一个给定的模板。NBS 模板表示的具体生成过程如下：首先输入模板作为初始图像，然后进行 K 次循环，每次循环找出一个最合适的基(以下简称最优基)及其对应的系数，再用当前的图像减去这个基及其系数重构的部分，相减得到的部分作为新的图像(称之为剩余图像)，接着继续下次循环，直到 K 次循环执行结束，就选出了 K 个哈尔基和它们对应的系数。图 2 所示是 NBS 模板表示的生成流程图。

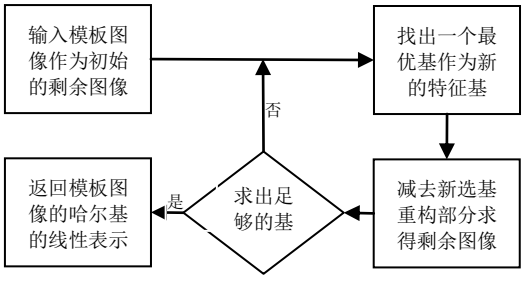


图2 NBS模板表示的生成算法流程

Fig.2 Flow of generating NBS template representation

由于图像模板对应的候选哈尔基数量非常多，要找出这些合适的哈尔基，使得它们构成的线性组合与模板相似度尽量高，这就需要一种策略来寻找这些基，MP[9]算法就是其中一种算法。

MP 算法最初是一种信号处理算法，它把一个信号分解为一个线性的基的组合，这些基来源于一个很大的基字典[13]。而 MP 算法就是用来从这个基字典中循环地找出若干个比较合适的基，使得这些基的线性表示尽量准确地重构原始信号。MP 算法的具体定义如下：

定义信号分解如下：$D=[b_1, b_2, \cdots b_N]$为二值特征基字典，对于模板 Y，经过 K 次循环筛选，选出了 K 个特征基为 $B_k=[b_1, b_2, \cdots b_k]$，这样可以用这 K 个基来重构 Y 为 $R_{B_k}(Y)$ 即：$R_{B_k}(Y)=\sum_{i=1}^{k} c_i b_i$。另外定义每次循环后信号的剩余部分为 $Y-R_{B_i}(Y)$（式中 i 表示第 i 次循环，初始剩余部分即为 Y）。

MP 算法使用的是常规的贪心机制[14]，它通过 K 次循环，每次循环找出一个特征基，并且这个特征基满足条件：信号剩余部分在这个选出来的基上的投射能量最大，即满足：

$$b_i = \arg\max_{b\in D} |< Y - R_{B_{i-1}}(Y), b >| \qquad (2)$$

并且 b_i 对应的系数 $c_i = < Y - R_{B_{i-1}}(Y), b >$。

这样，基于 MP 的 NBS 模板表示的生成就是进行 K 次循环，每次循环找出与当前剩余部分点积绝对值最大的基，并把它作为新的特征基然后继续下次循环。此时图 2 中第二个步骤找最优基就变为求点积找绝对值最大的基，如图 3 所示。

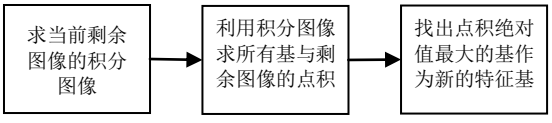


图3 基于MP方法求最大点积找最优基

Fig.3 Seek optimal base by figuring out maximum dot product based on MP

1.3 NBS 模板表示用于匹配的优越性

模板匹配主要用来研究某个特定图案位于整个检测图像中的位置，它根据相似度来确定该特定图案是否存在以及确切位置[3]。模板匹配过程中，在匹配区域（一幅大图像）上逐次选取大小相同的小图块，计算模板与小图块的距离，并根据距离值度量相似性，最终以最相似位置为匹配点。在计算距离时，很多常见的方法(比如 NCC，SSD 等[4-5])都是基于模板与图块的卷积。这些基于卷积算相似性的方法简单有效，但是卷积的计算量很大，比如对大小为 $h\times w$ 的模板，计算一次卷积就需要 $h\times w$ 次乘法。而引入 NBS 图像模板表示，将模板用少量哈尔基线性表示出来，再利用预先计算出的原图像的积分图像[4, 12, 15](定义如公式(3)所示)，就可以将卷积的大部分乘法运算转化为若干次的加/减操作。具体示例如下：

定义 $f(i, j)$ 为原图像坐标 (i, j) 处的像素，则定义原图像 $f(i, j)$ 的积分图像 $f_{\text{int}}(i, j)$ 如下：

$$f_{\text{int}}(i, j) = \sum_{m=1}^{i}\sum_{n=1}^{j} f(m, n) \qquad (3)$$

这样一个基与原图像的点积就由 $h\times w$ 次乘法计算转化为三次加/减操作，其过程如下：

$$\sum_{i=t}^{b}\sum_{j=l}^{r} f(i, j) = f_{\text{int}}(b, r) - f_{\text{int}}(b, l-1) - f_{\text{int}}(t-1, r) + f_{\text{int}}(t-1, l-1) \qquad (4)$$

式中 t，b，l，r 分别对应图 1 中哈尔基的白色矩形区域的上下左右四个坐标。

因此对于若干个哈尔基线性表示的图像模板，计算它与目标区域的点积就可以转化为计算这几个基与图像的点积，然后乘以系数求和。这样求点积的计算量能大大减少，进而在匹配时将大大提高计算速度。

1.4 NBS 模板表示的缺陷

由 1.3 节可以看出 NBS 模板表示用于匹配时，速度是非常快的，然而对一个图像模板，找出其对应的特征基的线性组合却是很耗时间的，这主要原因就是模板对应的基字典是非常庞大的，例如，对 $w\times h$ 的小图块，其基字典包含的候选基数目为 $w\times(w+1)\times h\times(h+1)/4$。如果在寻找特征基时采用 OOMP[10]算法，更是极为耗时。在[5]中作者提出使用 MP 替代 OOMP，能提高了速度，但是在很多实际应用中，是需要对大量的图块生成 NBS 模板表示(比如视频的逐帧匹配)，这就需要进行批量处理，此时 MP 方法仍然显得力不从心，它仍然会耗费大量时间。

2 NBS 模板表示生成算法可并行性分析

如图 2 和图 3 综合所示，输入一个模板图像，基于 MP 的 NBS 模板表示的生成，采用求最大点积的方法寻找出 K 个最优基(如图 3 所示)，然后输出该模板的哈尔基的线性组合。 它的提取过程主要是进行 K 次循环，每个循环可以概括为三步：利用积分图像求所有基与当前剩余图像的点积，然后找出点积绝对值最大的基，最后找到新的基后更新剩余图像对应的积分图像。对这三个步骤进行并行处理改造,是提升 NBS 模板表示生成算法效率的关键。

1)求点积

该过程中主要耗时的地方在于，每次循环时所有的基都要进行求点积计算。虽然每个基的求点积已经可以利用积分图像将计算复杂度降低到 3 次加/减运算，但是由于候选基数目庞大，时间开销还是很大。

考虑到每个基利用积分图像求点积，都是相同的操作（即 3 次加/减运算），所以可以最大限度地并行化处理，例如共有 N 个候选基，在完全并行的理想情况下，可以将 $N\times t$（t 是一个基求点积的时间开销）的时间开销通过并行缩为 t（通常 N 的数目至少是 10^4 以上）。在实际应用时，由于硬件资源的限制，虽然无法做到完全并行，但是可以通过任务并行分解方法，充分利用设备的并行能力。

2)找点积绝对值最大的基

在并行求出所有基的点积后，要找出绝对值最大的，此时可以简化为在数组中找最大元素。为了提高效率，在这一步也可以使用并行方法来提高速度。

3)更新积分图像

更新积分图像就是更新积分图像中所有的像素值，而每个像素的新值取决于更新前像素的值和选取的特征基及其系数，所以像素之间是没有任何影响的。因此对于一个像素个数为 M 的图像，更新积分图像可变为并行地更新 M 个像素的值。

3 基于 CUDA 的 NBS 模板表示并行生成算法设计

3.1 算法总体架构

基于 CUDA 进行算法设计时，需要充分考虑该平台本身的特点，对于一个 CUDA 程序，它的执行模型是将 CPU 作为主机端(Host)，GPU 作为设备端（Device）。执行时，主机将相关的数据传给设备的全局存储器，设备的执行模块从全局存储器中直接读取数据，或者将数据拷贝进该模块的共享存储器中以备用。因此设计 CUDA 程序时，就需要设计好传输的数据。

基于 CUDA 设计 NBS 模板表示的并行生成算法时，主要需解决三个难点问题：

1)负载平衡。负载平衡是并行设计的一个大难题，算法的关键就是要把基字典中数量众多的候选基均匀划分开来，分别进行处理。

2)减少主机与设备的数据传输。大量的数据传输会导致非常大的时间开销，将基字典预生成传给设备，然后通过下标均匀划分，这样做虽然适合均衡负载，但是却会导致过多的数据传输。

3)减少全局存储器的访问。全局存储器的访问也会导致很大的开销，不管是将基字典传给设备，还是在全局存储器中先预生成基字典，都会造成频繁的全局访问。

在充分考虑上述几个问题的基础上，提出基于 MP 的 NBS 模板表示并行生成算法的框架，如图 4 所示。

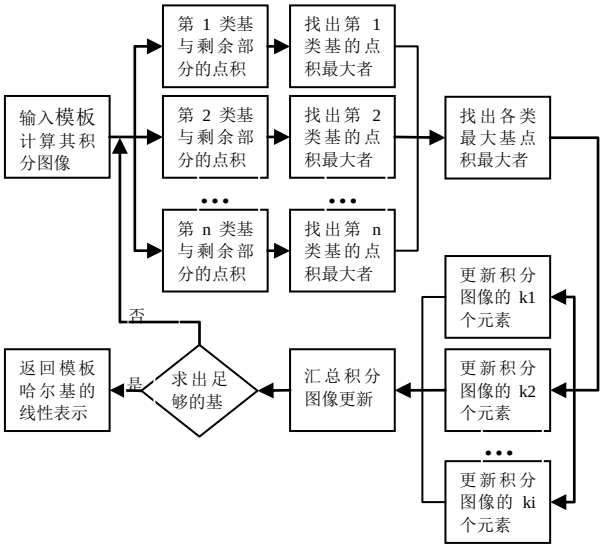


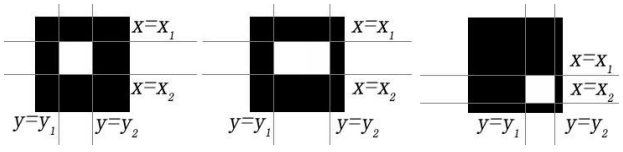
图4 基于MP的NBS模板表示的并行生成流程

Fig.4 Flow of parallel generation for NBS template representation based on MP

在流程设计时，为了充分利用硬件的处理能力，并且保证执行时的负载平衡，需采用对哈尔基分类的方法，来将候选基均匀分配给各个线程进行处理。而对于每个线程，为了减少数据的传输量和全局存储的访问频次，则采用让线程自己生成所要处理候选基的方法，这样就无需考虑候选基的存储访问问题。

3.2 哈尔基分类

对哈尔基采用“四条线算法”的分类方法。对任一个基，都视为四条直线即：$x=x_1$，$x=x_2$，$y=y_1$，$y=y_2$相交形成的，在分类时可以固定几条直线，然后变动另外剩余的直线，通过这种方法来将基分类。实验中规定 x_1和 x_2均相同的为同一类别的基，此时对 $h\times w$的模板，则共有 $h\times(h+1)/2$种不同种类的基，每个种类的基共有 $w\times(w+1)/2$个。图 5 是采用该基分类方法的示例，其中(a)和(b)属于同一类基(标记为 I)，而(c)则属于另外一类基(标记为 II)。



(a) I类基 (b) I类基 (c) II类基

图5 “四条线算法”划分基的示例

Fig.5 Example of bases classification by the “four-line” algorithm.

在处理时，每个线程可以利用自身编号计算出一组 x_1、x_2，从而映射到一类基中。通过这种方法，就可以将若干个线程作为一组，用这组线程来处理一类基，因为每类基的数目都是相同的，所以可以很好的做到负载平衡。

3.3 基于基分类的模板表示并行生成实现

对于提出的并行生成流程，在求取若干基的过程中，每次循环都涉及到积分图像的运用，因此在流程之初，需要初始化积分图像，即计算输入模板对应的积分图像。根据文章[16]，可直接借用作者在 CUDA 平台上求积分图像的并行扫描算法。具体做法就是将模板先按行从左至右并行扫描累加，然后再按列从上至下并行扫描累加，这样就很快得到积分图像。

接下来的步骤就是循环求取若干个特征基，对获取一个特征基的过程，它的并行实现包括：并行地求各基对应的点积值、并行找出最大点积和并行更新积分图像。

1)并行求点积值

这一步是涉及到负载平衡的关键步骤，首先将所有基按 3.2 节的“四条线算法”划分成不同的种类，然后对每类基分配若干线程去处理。在实验过程中，有两种情况如下。

(1)对单个图块（大小为 $w\times h$），直接为每一类基分配一个 block（大小为 S_{block}，编号为 I_{block}），此时每个 block 处理一类基，对应为（按四条线算法划分）：

$$x_1 = I_{\text{block}} \;/\; h$$

$$x_2 = I_{\text{block}} \;\%\; h$$

在 block 内部每个线程（编号为 I_{thread}）处理一个或多个基，对应为：

$$y_1 = (i \times S_{\text{block}} + I_{\text{thread}}) \;/\; w$$

$$y_2 = (i \times S_{\text{block}} + I_{\text{thread}}) \;\%\; w$$

(2)对于批处理大量图块，为了降低复杂性，一类基直接分配一个线程来处理，这样分配一个 block（大小为 S_{block}）处理一个图块，多个 block 处理多个图块，此时 block 内每个线程（编号为 I_{thread}）处理一类基，对应为（按四条线算法划分）：

$$x_1 = (i \times S_{\text{block}} + I_{\text{thread}}) \;/\; h$$

$$x_2 = (i \times S_{\text{block}} + I_{\text{thread}}) \;\%\; h$$

y_1，y_2则执行循环遍历所有情形。

在实验中，i 的取值从 0 开始，具体 i 取到多少跟具体图块大小有关，执行时根据 w 与 h 大小自动分配。

2)并行找最大点积

如流程图 4 中所示，需多次寻找最大的点积(包括局部最大点积和全局最大点积)，即首先在每类基中需要先找出本类基中点积最大的哈尔基，然后综合再次求所有基中点积最大的哈尔基。考虑到只要找出最大的那个点积，因此并不需要进行排序，再结合 CUDA 大量轻线程并行的特点，引入锦标赛排序的思想，即把每个点积分配给一个线程，将线程两两进行比较并淘汰掉点积值较小的线程，直到只剩下一个线程为止，此时该线程对应的点积即为最大。

3)并行更新积分图像

具体算法如下：假设找出的第 k 个特征基为 $a=\{x_{\text{start}}, y_{\text{start}}, w, h\}$，它对应的系数为 c，此时未更新前的积分图像为 F, $f(x,y)$ 表示坐标 (x,y) 处的像素值，则可以如公式(5)更新积分图像。

$$f(x,y)\begin{cases} f(x,y), x<x_{\text{start}} \text{或} y<y_{\text{start}} \\ f(x,y)-\min(h,(x-x_{\text{start}}))\times \\ \min(w,(y-y_{\text{start}}))\times c, \text{其他} \end{cases} \tag{5}$$

每个像素点由一个线程来负责更新，这样可以极大地并行更新。

4 实验设计与结果分析

为了测试 NBS 模板表示并行生成的提速效果、算法的正确性以及在实际应用时的性能改善，分别设计了三个实验来进行验证。它们包括单纯的 NBS 模板表示生成实验，人脸匹配实验以及图像匹配实验。

4.1 NBS 模板表示并行生成实验

为了测试 NBS 模板表示并行生成的速度，如下设计实验。考虑到单独生成一个小图块的 NBS 模板表示时速度非常快，时间开销测量不准确，所以设计实验时，采用了大量处理求平均值的方法，来测试 NBS 模板表示并行生成的提速效果。另外对于很多实际应用，是要对大量的小图块生成 NBS 模板表示的，比如进行视频匹配时，需要匹配很多帧，而每帧都需要选取多个小图块，对这些小图块生成 NBS 模板表示。所以这个设计的实验也是满足实际应用需求的。设计时共选取了 6 种不同场景，每种场景选取 1000 张图片进行实验，对每个图片提取 surf 特征点[17]，并选取 10 个特征点形成对应图块，这样共计对 6×1000×10 个图块生成 NBS 模板表示。实验中，对每个图块采用 10、15、20、25、30、35、40、50 个基分别进行重构。实验平台为 NVIDIA Tesla C2050 GPU，内建 448 个 CUDA 核心。操作系统是 Windows 7，开发工具为 Visual Studio 2010 + OpenCV2.2。

表 1 列出的是分别选取 20 个基和 30 个基进行重构时，不同大小的图块生成 NBS 模板表示所需要的平均时间开销。图 6 所示是进一步分析出的实验结果。

表 1 不同大小图块并行生成 NBS 模板的平均时间开销

Table 1 The average time cost of the NBS parallel-generate processing for different patch-size

	Size	GPU/s	CPU/s	Rate
用	20×20	0.000479	0.029	60.54
20	30×30	0.001624	0.147	90.51
个基	40×40	0.004608	0.457	99.18
重构	50×50	0.009368	1.119	119.44
	60×60	0.017694	2.279	128.80
用	20×20	0.000724	0.044	60.77
30	30×30	0.002436	0.218	89.49
个基	40×40	0.006862	0.683	99.53
重构	50×50	0.014048	1.658	118.02
	60×60	0.026594	3.421	128.63

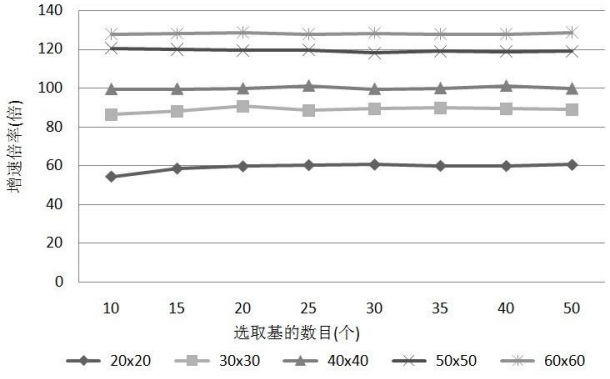


图 6 基大小和个数对算法提速效果的影响

Fig.6 The effect of base size and num on the speed-up of the algorithm

4.2 人脸匹配试验

4.2.1 人脸匹配

为了测试并行生成的 NBS 模板表示的准确性，设计了人脸匹配[18]的实验如下。总共选取 100 张不同的人物集体照，每张选取了 10 个人脸，分别用非并行算法与并行算法生成 NBS 模板表示（30 个基），然后在图像中匹配(如图 7 示例)，并且记录匹配结果，如表 2 所示。由表中的数据可以看出并行生成的 NBS 模板表示的准确性并没有下降。

表 2 人脸匹配结果

Table 2 Results of face-matching

	图像总数	人脸总数	正确匹配	准确率/%
并行	100	1000	716	71.6
非并行	100	1000	716	71.6

(a)原始图像

(b)人脸图块 (c)正确匹配 (d)实验匹配

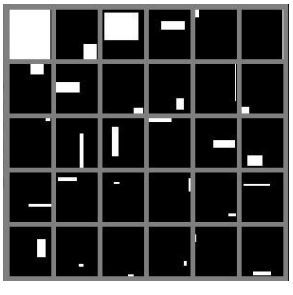
图7 人脸匹配实验

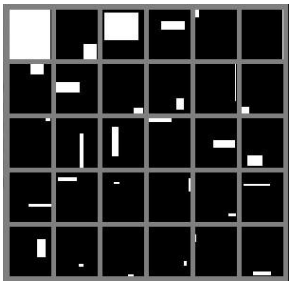
Fig.7 Experiment of face-matching

4.2.2 人脸的 NBS 模板表示

为了进一步了解并行生成的 NBS 模板表示的准确性，任取若干张人脸图块，对其基于 GPU 和 CPU 分别生成 NBS 模板表示（这里选取了 30 个基），将两个不同平台上选取的特征基和特征基对应的系数进行对比，得到的其中一组结果如图 8 所示。

(a)原图块 (b) CPU 上重构 (c)GPU 上重构





(d) 基于 CPU 采用的基 (e)基于 GPU 采用的基

图 8 人脸图像的 NBS 模板表示。

Fig.8 The NBS template representation of the face patch

实验结果显示，并行生成的 NBS 模板表示使用的基，与不是基于 CUDA 生成的 NBS 模板表示使用的基是一样的，并且每个基对应的系数也都是相似的。

由上面这些实验可以很容易得出结论，即：提出的基于 CUDA 并行生成的 NBS 模板表示是正确的。

4.3 图像匹配实验

为了测试 NBS 模板表示的并行生成在实际应用中的提速效果，测试过程中选取了图像处理中图像匹配来进行实验。首先设定一组简单的匹配规则：给定两幅图片，对其中一幅图像提取 surf 特征点，取 10 个特征点形成对应图块（这里取大小为 35×35），然后将这 10 个图块用 4.1 节中的人脸匹配方法，在另外一幅图像分别进行匹配，根据所有的匹配结果判定两图像是否相似。

基于上面的规则，共选取 6 种不同场景，每种场景选取 100 张图片（大小为 650×750）作为图库，将这 600 个图像逐一取出与图库所有图像进行匹配，记录平均匹配时间如表 3 所示。其中 C-M 和 G-M 分别表示基于 CPU 和基于 GPU 做一次图像匹配所需的总时间，C-N 和 G-N 分别表示基于 CPU 和基于 GPU 生成模板表示使用的时间。

表 3 图像匹配的平均时间开销（采用 20 个基）

Table 3 The average time cost of image matching(using 20 bases)

类别	C-M/s	G-M/s	C-N/s	G-N/s	C-N/C-M /%	G-N/G-M /%
人物	4.95	2.80	2.71	0.206	54.7	7.3
动物	4.84	2.78	2.69	0.198	55.5	7.1
室内	4.92	2.82	2.75	0.208	55.8	7.3
户外	4.87	2.79	2.80	0.197	57.4	7.1
风景	4.91	2.81	2.69	0.210	54.8	7.4
建筑	4.79	2.80	2.74	0.200	57.2	7.1
平均	4.89	2.79	2.72	0.199	55.6	7.1

由于只是做一点简单的测试对比，所以图像匹配的算法并未刻意极致优化，并且由于每次调用 GPU 时都是只简单处理 10 个图块(没有进行大批量的处理)，所以模板表示的生成提速效果不是很明显（大约 10 倍多），然而，由表 3 的数据可以很明显看出，NBS 模板表示生成在处理总时间开销中所占的比例，很明显地降低了。并且在没有进行大规模图像批处理的情况下，并行生成 NBS 模板表示的提速仍然让图像匹配所耗时间，减少了近 50%。在图像匹配这一应用，并行生成 NBS 模板的提速效果是非常明显的，可以考虑进一步挖掘 NBS 模板表示并行生成算法大幅提速其它应用的空间。

5 结论

提出了一种基于 CUDA 平台的 NBS 图像模板表示的快速并行生成算法，该算法充分运用 GPU 的并行计算能力，对于 NBS 模板表示并行生成的主要三个步骤，逐一分解成多个能够并行的子步骤，让 GPU 并行执行这些规模较小的子步骤，从而来缩减整个提取过程所需的时间。实验测试平均性能达到了 CPU 实现的 60 倍~120 倍，NBS 图像模板表示的并行生成算法在不少应用中可以发挥出很大的提速效果。

参考文献(References):

[1] Briechle K, Hanebeck U D. Template matching using fast normalized cross correlation[C]//Proceedings of SPIE. Orlando, Florida, USA: SPIE, 2001, 4387: 95-102.

[2] Schweitzer H, Bell J, Wu F. Very fast template matching[C]//Proceedings of Seventh European Conference on Computer Vision. Copenhagen, Denmark: Springer, 2002, 2353: 358-372.

[3] Lewis J P. Fast template maching[C]// Proceedings of Vision Interface. Quebec, Canada: Canadian Image Processing and Pattern Recognition Society, 1995: 120-123.

[4] Tao H, Crabb R, Tang F. Non-orthogonal binary subspace and its applications in computer vision[C]// Proceedings of International Conference on Computer Vision. Beijing, China: IEEE, 2005, 1: 864–870.

[5] Tang F, Tao H. Fast Multi-scale Template Matching Using Binary Features[C]// Proceedings of IEEE Workshop on Applications of Computer Vision. Austin, TX, USA: IEEE, 2007: 36-36.

[6] Tang F. Representing images using non-orthogonal haar-like bases[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(12): 2120–2134.

[7] Tang F, Tao H. Fast linear discriminant analysis using binary bases[C]//Proceedings of International Conference on Pattern Recognition. Hong Kong, China: IEEE, 2006, 2: 52-55.

[8] Hel-Or Y, Hel-Or H. Real-time pattern matching using projection kernels[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(9): 1430-1445.

[9] Mallat S, Zhang Z F. Matching pursuits with time-frequency dictionaries[J]. IEEE Transactions on Signal Processing, 1993, 41(12):3397-3415.

[10] Rebollo-Neira L, Lowe D. Optimized orthogonal matching pursuit approach[J]. IEEE Signal Processing Letters, 2002, 9(4): 137-140.

[11] Pan J, Pang Y W, Li X L. A fast feature extraction method[C]//Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. Taipei, Taiwan, China: IEEE, 2009:1797-1800.

[12] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[C]// Proceedings of Computer Vision and Pattern Recognition. Kauai, Hawaii, USA: IEEE, 2001,1: I511–I518.

[13] Gilbert A J, Muthukrishnan S, Strauss M J. Approximation of functions over redundant dictionaries using coherence[C]//Proceedings of 14th Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, USA : ACM,2003: 243-252.

[14] Davis G, Mallat S, Avellaneda M. Greedy adaptive approximation[J]. Constructive Approximation, 1997, 13(1): 57-98.

[15] Mita T, Kaneko T, Hori O. Joint haar-like features for face detection[C]//Proceedings of International Conference on Computer Vision. Beijing, China: IEEE, 2005, 2: 1619–1626.

[16] Berkin B, Horn B K P, Masaki I. Efficient Integral Image Computation on the GPU[C]//Proceedings of IEEE Intelligent Vehicles Symposium(IV). San Diego, CA: IEEE 2010 :528-533.

[17] Bay H, Tuytelaars T. SURF: Speeded up robust features[J]. Computer Vision–ECCV, 2006, 3951:404-417.

[18] Chen X R, Gu L, Li S Z. Learning representative local features for face detection[C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Kauai, HI: IEEE, 2001, 1: 1126-1131.