基于 Zynq 的人脸检测设计

霍芋霖 符意德
(南京理工大学计算机科学与工程学院 南京 210094)

摘 要 针对目前大多嵌入式人脸检测系统因资源限制而导致的软件方式实现速度较慢的问题,提出利用软硬件协同的方法来加速人脸检测。在 Zynq-7000 平台的基础上,使用 C 语言实现了基于 AdaBoost 级联分类器的人脸检测算法,并测试了各个模块的运行时间。结合算法实现的具体过程及其繁复程度给出了硬件加速方案。将检测算法计算量大而多的部分转移到硬件部分进行优化加速,在 Zynq-7000 平台上实现了软硬件协同的人脸检测,最后给出了相应模块的加速结果。

关键词 人脸检测,AdaBoost,硬件加速,Zynq
中图法分类号 TP319.4 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.10.060

Face Detection Design Based on Zynq

HUO Yu-lin FU Yi-de
(School of Computer Science and Engineering, Nanjing University of Science & Technology, Nanjing 210094, China)

Abstract Compared with the desktops, there are mainly two problems of the embedded face detection systems, which are less resource and slower speed. Here we put forward the method of using software-hardware codesign to accelerate the face detection process. Firstly, we implemented the face detection algorithm on platform Zynq-7000 with C programming language, which is based on AdaBoost cascade classifiers. Then, the performace of the algorithm was tested with Xilinx SDK. Afterwards, a software-hardware partion strategy of the algorithm was raised for acceleration. By transplanting the most computational part of the algorithm to the hardware implementation, we realized the software-hardware codesign of the face detection system on platform Zynq-7000. At last, we presented the acceleration results of two hardware modules, and summarized the related work.

Keywords Face detection, AdaBoost, Hardware acceleration, Zynq

1 引言

人脸检测最初是作为人脸识别中的定位环节而被提出使用的。随着检测技术的不断发展,人脸检测被应用到更多的环境中,如人脸跟踪、姿势估计和表情识别等技术都会用到人脸检测功能[1]。

目前,常用的人脸检测算法是由 Viola 和 Jones 提出的 AdaBoost 人脸检测算法[7],OpenCV(Open Source Computer Vision)对其进行了扩展和实现,完成了实时人脸检测。因此,不少研究者通过移植 OpenCV 以在嵌入式环境下实现实时的人脸检测,但其速度以及资源的利用问题仍然需要进一步优化[2]。由于嵌入式环境下的纯软件或者纯硬件均不能同时获得速度和资源这两方面的优势,研究者提出了采用部分硬件加速的方式来实现人脸检测,并借此来寻求一个速度与资源的平衡点。如文献[3]中采用了 DSP(Digital Signal Processing)加速的方式,这是一种比较常用的方法。在文献[4]中,Acasandrei L. 等人采用了 AMBA(Advanced Microcontroller Bus Architecture)总线硬件 IP 核(Intellectual Property Core)加速器。也有不少学者对软硬件结合的加速方式进行了研究,如文献[5]中施跃华等人对基于 AdaBoost(Adaptive Boosting)算法的人脸检测进行了软硬协同设计的研究,并通过改进纯硬件的实现方式,在 SoC 的环境下实现了人脸检测的软硬协同设计。汤欲涛等人也在文献[6]中利用 Zynq-7000 实现了实时人脸检测系统的设计,但其只对图片采集进行了加速,检测部分仍用软件方式实现。

本文利用文献[9]中给出的分类器放缩方式,结合 Zynq 平台以及 Xilinx 公司提供的 Vivado 系列软件,实现了人脸检测设计。Zynq-7000 All Programmable SoC(System on Chip)[10]是一种 ARM+FPGA(Field-Programmable Gate Array)的体系结构平台,它集成了 ARM Cortex A9 双核以及最多可达相当于 500 多万个逻辑门的可编程逻辑单元,能够灵活地用于各种应用当中。Zynq-7000 的处理器子系统中集成了内存控制器和大量外设,使得 Cortex A9 的核完全独立于可编程逻辑单元,可以独立工作。同时,其 FPGA 部分用于扩展子系统,有着丰富的扩展能力,内部互连超过 3000 个,连接资源非常丰富,并集成了高速串行口,非常适合于软硬件协同加速设计。

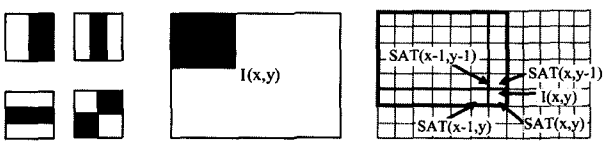
到稿日期:2015-09-14 返修日期:2015-12-28
霍芋霖(1990—),女,硕士生,主要研究方向为嵌入式软件等,E-mail:1602266035@qq.com;符意德(1964—),男,硕士,副教授,CCF 高级会员,主要研究方向为智能控制与嵌入式系统、普适计算。

ZedBoard 则是一款基于 Zynq-7000 的低成本、高性价比的开发套件，它包含了几乎所有必需的接口以及各种支持功能，是一款理想的设计验证平台。

2 人脸检测算法

本文采用的检测算法是基于 Haar 特征的 AdaBoost 级联分类器算法，该算法可以认为是一种通用的目标检测算法，人脸检测只是其中一种应用。AdaBoost 算法是一种用来训练分类器的迭代算法，最早是由改进的 Boosting 算法得来，后来 Paul Viola 和 Jones 又提出了积分图和级联分类器的概念来加快检测速度，并将其融合于 AdaBoost 检测算法，得到了最终的 AdaBoost 人脸检测算法[7]。

Haar 特征是一系列的黑白矩形框，主要用来描述人脸特征，如图 1(a)所示，其特征值等于黑白矩形框内像素灰度值的加权和。



(a)Haar 特征　(b)积分图　(c)积分图增量计算

图 1　Haar 特征和积分图

由于一张图中可以有上万个 Haar 特征，为了计算方便，引入了积分图(Summed Area Table)的概念，如图 1(b)所示。对于一个正 Haar 特征，积分图的定义为：

$$SAT(x,y)=\sum_{0\leqslant x'\leqslant x,0\leqslant y'\leqslant y} I(x',y') \tag{1}$$

它所表示的值是点(x,y)左上方所有像素的灰度值之和，$I(x',y')$表示待检测图像上该像素点的灰度值。积分图中存储的是整幅图像的每个像素点 $I(x,y)$处的积分值，实际应用中，积分图 $SAT(x,y)$采用增量方式计算，如图 1(c)所示。

$$SAT(x,y)=SAT(x,y-1)+SAT(x-1,y)+I(x,y)-SAT(x-1,y-1) \tag{2}$$

其中，有 $SAT(-1,y)=SAT(x,-1)=SAT(-1,-1)=0$。

在利用积分图计算特征值时，只需经过 4 次查找和 3 次加减运算即可获得 1 个矩形的特征值。例如，对于 Haar 特征中的一个矩形 $r=(x,y,w,h,0°)$，其特征值的计算方式为：

$$RecSum(r)=SAT(x,y)+SAT(x+w,y+h)-SAT(x,y+h)-SAT(x+w,y) \tag{3}$$

此处采用的分类器是 OpenCV 库中的 22 级正脸级联分类器，共有 2135 个 Haar 特征。在该级联分类器中，级联结构由 22 个强分类器组合构成，每一个强分类器则是由若干个弱分类器组成，越靠后的强分类器分类能力越强，所包含的弱分类器也越多。一个弱分类器就相当于一个 Haar 特征，而能通过全部强分类器的图片窗口则认为是人脸窗口。

3 算法分析及硬件加速模块划分

根据上节所述算法原理，在 Zynq-7000 扩展式处理平台上实现了基于 AdaBoost 级联分类器的人脸检测算法，其检测流程如图 2 所示。

下面结合人脸检测算法的具体实现步骤及耗时情况，对软硬件功能进行划分。其各个模块在单个人脸和多个人脸两种情况下的耗时如表 1 所列。

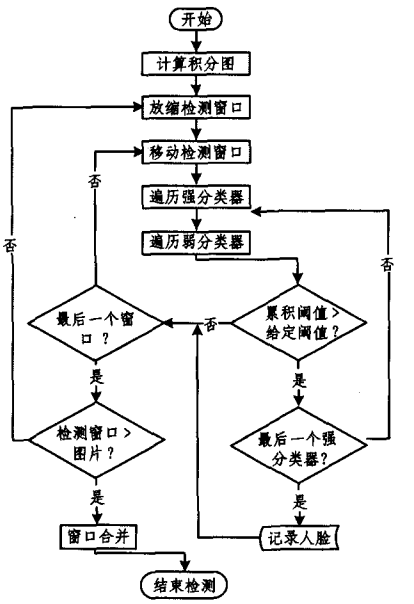


图 2　人脸检测流程

表 1　算法模块运行时间

	积分图计算(ms)	窗口检测(ms)	窗口合并(μs)
单个人脸	23.4(0.69%)	3541.0(99.31%)	0.032(≈0%)
多个人脸	23.4(0.43%)	5311.4(99.56%)	3.298(≈0%)

软硬件功能划分其实就是一种组合优化问题，目的是使系统的整体性能、运行时间、资源损耗等达到最优[8]。划分时需要重点考虑的因素有运行时间、复杂度和并行性。

首先，计算待检测图像的积分图，此过程并不复杂，只是因循环次数较多而导致计算量较大。程序只需遍历一次待检测图像即可计算出整个积分图，相同大小的图片的耗时相同。虽然该部分耗时并不多，但是考虑到该部分的单一性以及固定性，采用硬件来实现是最好的选择。

其次是人脸检测部分，检测部分主要由 4 个大循环构成，从外到内依次为检测窗口放缩循环、图像遍历循环、强分类器遍历循环和弱分类器遍历循环。人脸检测过程最内层循环为弱分类器遍历循环，该阶段主要完成的任务包括计算窗口对应的 Haar 特征值和累加影响因子。紧接着是次内层循环，此层循环的任务是将最内层循环得到的影响因子累加和与相应的强分类器阈值作比较，直到确定检测窗口不是人脸或者遍历完所有强分类器。其次，图像遍历循环则相对简单，只需设置好移动步长，按方向移动检测窗口即可。最外层是放缩检测窗口，从初始窗口大小开始，以一定的倍数逐步放大检测窗口以检测到不同尺度下的人脸。例如，一般初始窗口为 20×20，放大倍数选定为 1.2。由表 1 可知该过程耗时较多，因此需要进行硬件加速。分析可知，循环同外部的数据传输，如分类器的读取，几乎均是顺序实现的，并且无需其他控制操作，故分类器遍历部分适合采用硬件实现。检测窗口的移动采用固定模式，再考虑到循环之间有数据传递，同时为了减少软硬件之间的交互，该模块的全部循环均采用硬件来实现。

最后是人脸窗口的合并。该过程可以分为两步：将检测到的人脸窗口划分成不同的等价类；去掉被包含的小窗口。

划分等价类的原则主要参考两个矩形之间的顶点距离以及相互的长宽比例，该过程需要大量的比较判断和浮点运算，且占用非常少的时间，更适合于用软件来实现。

4 加速模块设计实现

依据前面选定的加速模块，利用 Xilinx 公司推出的软硬件综合软件 Vivado HLS(High-Level Synthesis)来进行硬件仿真综合实现，选用的工作频率为 100MHz。对于模块加速前后的初步测试，这里采用一张 320×240 的单人脸灰度图来完成，测试用图如图 3 所示。

图 3 测试用图

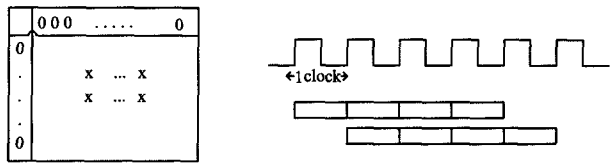
系统中有两个常量部分为人脸检测过程提供数据，分别是图像数据和分类器数据。图像数据实际由前端系统采集得到，将其转换为灰度图，存储在随机存取存储器(Random Access Memory，RAM)中以供使用。由于灰度图的每个像素值不超过 255，因此以 8bit 一个像素的方式来存储图片数据。分类器则采用现有的正脸检测分类器，在此之前将数据从 xml 文件中提取出来并分成 5 个部分：Haar 特征矩形、弱分类器阈值、左影响因子、右影响因子和强分类器阈值，把各个部分的数据分别存放于只读存储器 ROM(Read Only Memory)中。其中，由于分类器中的 Haar 特征数量较大，这里将 Haar 特征矩形分成 5 个部分来存储，即顶点 x 坐标、顶点 y 坐标、矩形长度、矩形宽度和矩形权重。

4.1 积分图模块

如前所述，积分图的计算就是在遍历图像的同时完成像素值的累加即可。而积分图中每个数据的宽度则可根据图片大小来确定，如本文采用 320×240 的灰度图，那么积分图可能出现的最大值为 320×240×255＝19584000，由对数公式计算可知至少需要 25bit 来存储积分图的每个数据。为了减小检测误差，同时进行了平方积分图的计算，同理可得，平方积分图的每个数据则需要 33bit。

由式(3)可知，积分图累加过程中涉及其前一行前一列的几个数据，为了去掉运算时的首行首列判断逻辑，给积分图添加一行和一列全为零的数据，如图 4(a)所示，这样既降低了实现复杂度，又减少了部分运算时间。接下来，对循环过程进行流水线化以进一步提升运行效率。循环内部主要进行几个赋值运算和两个积分数据的运算，综合可知需要 4 个时钟周期来完成，其流水线结构如图 4(b)所示。第一个时钟读取图片数据并完成该列的首行添加；第二个时钟则读取积分图累加需要的 3 个数据并得到累加和；第三个时钟写入积分图并完成平方积分图的累加；最后写入平方积分图。基于以上结构，大大加速了积分图的计算，表 2 列出了几种情况下积分图计算的耗时情况，其中系统设定的时钟为 10ns，Estimated clock 为综合后得到的预估时钟。最后，计算得到的结果分别存储在两个双端口 RAM 中。

在 ZedBoard 板上，纯软件条件下，大小为 320×240 的灰度图的积分图模块的运行时间固定在 23.4ms；而利用 HLS 综合得到的硬件加速模块运行时间为 $78241\times7.92\times10^{-6}$ ms≈0.62ms，相比软件实现提升了近 40 倍。



(a)增加行列后的积分图　　(b)流水线化

图 4 积分图实现

表 2 积分图的 3 种实现方式

	初始积分计算	增加行列后的积分计算	流水线积分计算
Latency	614403	307921	78241
Estimated clock(ns)	8.15	7.51	7.92

4.2 检测模块

启动并进入检测模块时，首先需要 1 个时钟周期进行部分寄存器数据的初始化，如初始窗口的位置与大小、放缩倍数等。然后开始进行人脸特征匹配，该部分是检测模块的核心部分，也是循环最内层。如图 5 所示，从 ROM 中(图中斜四边形)读取分类器数据，由于分类器数据中的 Haar 特征矩形大小是在 20×20 的检测窗口下的值，因此在检测时需要随着检测窗口的放大而进行放大(scale)。

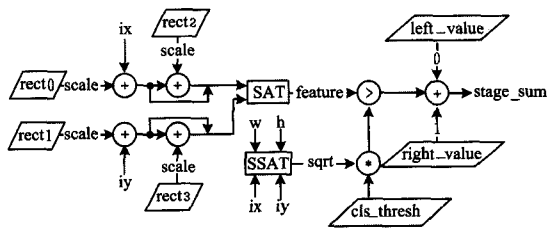


图 5 Haar 特征匹配

放大后，根据检测窗口的坐标(ix，iy)确定 Haar 特征的坐标点，从积分图(SAT)中读取积分值并计算其特征值(feature)，得到相应的 Haar 特征值后与弱分类器阈值(cls_thresh)进行比较，根据比较结果累加影响因子(left_value/right_value)，其中平方积分图(Square Summed Area Table，SSAT)是为了补偿放缩误差。在实现过程中，该部分利用流水线来减少迭代时间，流水线包括的内容即图 5 所示部分。接下来根据强分类器阈值判断是否为人脸，并放缩移动检测窗口即可完成整个检测过程。得到的人脸窗口将被记录下来并通过 AXI (Advanced eXtensible Interface)[11] 传送给 ARM 部分，最终完成窗口合并任务。

由于检测模块实现内部的不确定性，所需时钟范围波动很大，为了得到一个确定的值，下面将利用综合结果以及一些参考数据对其进行时钟估计。

首先，可以根据算法实现将四层循环分为两个计数部分：检测窗口数和分类器数。其中检测窗口数对于固定大小的图片也是不变的，如文中始终采用的是 320×240 的灰度图，那么总共需要检测的窗口个数为 94468。分类器数目则完全不是固定的，因此，分别对 3 种图片进行数据统计以找到一个平

均值，如表3所列。通过统计数据可以看出，大多图片都能通过的强分类器均值为2，而前两级强分类器总共有19个弱分类器，由于分类器的遍历基于强分类器个数，因此采用19这个数据进行下一步计算。

表3 分类器数据统计

	无人脸图片	单人脸图片	多人脸图片
强分类器数	2	2	2
弱分类器数	13	13	15

检测模块最内层循环时钟为72，又因其处于流水线中，故只要知道其循环次数即可继续向外层推出。综合以上信息，可以计算得到表4所列结果。

表4 时钟估算结果

	个数	耗费总时钟数	流水线	计算方式
检测窗口	94468	25034070	否	(216+49)×94468+50=25034070，其中49和50为额外消耗
弱分类器	19	216	是	72+(19−1)×8=216，其中8为流水线深度
时间计算	25034070×8.67×10^{-6}ms≈217.05ms(Estimated Clock:8.67ns)			

其中，流水线耗时计算公式为$L+(N-1)\times II$，L为时钟数，N为循环次数，II为流水线深度。

在ZedBoard平台上测试了单个人脸检测在纯软件条件下的运行速度，检测模块大约需要运行3541ms，与该模块的硬件加速对比可以发现，加速后快了约15倍。可见，相对于纯软件检测，硬件加速大大加快了测试的进度，减少了时间开销。

5 测试结果

测试选取了来源不同、质量不同的91张320×240灰度图，其中包括无人脸、单个人脸和多个人脸3种图片，共有285个人脸。人脸图片包括正脸和略有倾斜的脸两种，所有人脸均无明显外在遮挡，其检测结果如表5所列。

表5 测试结果

	加速前平均时间(ms)	加速后平均时间(ms)	检测率(%)	误检率(%)
单个人脸	3584.4	233.4	98.0	7.1
多个人脸	5445.3	397.1	81.5	1.1
没有人脸	3497.1	215.7	—	—

根据测试结果可发现，加速之后算法的检测率和误检率几乎不变，而影响检测率和误检率较大的参数始终都在于窗口合并过程中设定的最小合并数(即一个窗口被检测到的次数达到最小合并数才能合并为人脸，否则丢弃)。并且，最小合并数越小，人脸窗口检测率越高，但误检率也会上升。同时，检测的加速效果也保持在10倍以上，达到了实时检测的目的。

结束语 本文对常用的人脸检测算法在嵌入式环境下进行了研究，采用了基于硬件加速的方式来提升嵌入式人脸检测速度。在Zynq-7000平台软硬一体化平台的基础上实现了基于AdaBoost级联分类器的人脸检测算法，在不降低检测率的基础上达到了加速人脸检测的目的。该系统充分利用了平台的软硬件优势，不仅加快了人脸检测的速度，更为合理地利用了嵌入式系统的现有资源，大大加速了人脸检测在嵌入式环境下的运行速度。

参考文献

[1] Yang M H, Kriegman D J, Ahuja N. Detecting faces in images: A Survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(1): 34-58

[2] Zhu Ming, Lu Xiao-feng, Lu Heng-li, et al. Transplantation and optimization of AdaBoost face detection algorithm on DSP[J]. Computer Engineering and Applications, 2014, 50(20): 197-201 (in Chinese)
朱明，陆小锋，陆亨立，等. AdaBoost人脸检测算法在DSP上的移植与优化[J]. 计算机工程与应用，2014，50(20)：197-201

[3] Aby P K, Jose A, Dinu L D, et al. Implementation and optimization of embedded face detection system[C]// 2011 International Conference on IEEE Signal Processing, Communication, Computing and Networking Technologies (ICSCCN). 2011: 250-253

[4] Acasandrei L, Barriga A. AMBA bus hardware accelerator IP for Viola-Jones face detection[J]. IET Computers & Digital Techniques, 2013, 7(5): 200-209

[5] Shi Yue-hua, Zhao Feng, Zhang Zhong. SoC software and hardware co-design for face detection system based on AdaBoost algorithm[J]. Information Technology, 2008, 32(9): 151-153 (in Chinese)
施跃华，赵峰，张忠. AdaBoost算法的人脸检测系统的SoC软硬件设计[J]. 信息技术，2008，32(9)：151-153

[6] Tang Yu-tao, Liang Rui, Liu Chang. Face detection system design based on the SOPC[J]. Electronic Design Engineering, 2014, 22(21): 165-168(in Chinese)
汤欲涛，梁锐，刘畅. 基于SOPC的人脸检测系统的设计[J]. 电子设计工程，2014，22(21)：165-168

[7] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[C]// Proceedings of the 2001 IEEE Computer Society Conference on IEEE Computer Vision and Pattern Recognition, 2001(CVPR 2001). 2001, 1: 511-518

[8] Kang Hong-yan. Research on Application of Hardware-software Co-design in Embedded System[J]. Journal of Heze University, 2010, 32(5): 48-51(in Chinese)
康鸿雁. 嵌入式系统中软硬件协同设计技术应用研究[J]. 菏泽学院学报，2010，32(5)：48-51

[9] Acasandrei L, Barriga A. Accelerating Viola-Jones face detection for embedded and SoC environments[C]// Institute of Electrical and Electronics Engineers. 2011: 1-6

[10] Lu Jia-hua, Jiang Zhou, Ma Min. Hardware-software codesign in embedded system practical guide: Based on Xilinx Zynq[M]. Beijing: China Machine Press, 2012: 152-165(in Chinese)
陆佳华，江舟，马岷. 嵌入式系统软硬件协同设计实战指南：基于Xilinx Zynq[M]. 北京：机械工业出版社，2012：152-165

[11] Xilinx Inc. AXI Reference Guide v14.3 [EB/OL]. (2012-11-15) [2015-03-12]. http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug761_axi_re-ference_guide.pdf