基于Zynq的图像角点及边缘检测系统的设计与实现

潘青松 张 怡 杨宗明 秦剑秀
(西南交通大学 成都 610031)

摘 要 以Zynq芯片为基础,采用软硬件协同设计的方法设计并实现整个系统。Zynq芯片内部采用ARM+FPGA的异构架构,既具备ARM处理器的灵活性,又拥有FPGA并行处理的能力。本系统的设计充分发挥了Zynq芯片的优势,在软硬件划分上,通过ARM处理器来实现图像的采集;图像角点及边缘检测用FPGA来完成,即通过硬件加速提升系统的整体性能。ARM处理器与FPGA通过AXI4总线进行数据交互,在Zynq上实现集图像采集、图像特征提取、图像显示为一体的片上系统。最终系统测试结果表明,采用硬件加速实现图像特征提取的相关算法比在ARM处理器软件上实现的算法的速度提高了6~8倍。

关键词 Zynq,FPGA,特征提取,软硬件协同设计
中图法分类号 TP368 **文献标识码** A

Design and Implementation of Image Corner and Edge Detection System Based on Zynq

PAN Qing-song ZHANG Yi YANG Zong-ming QIN Jian-xiu
(Southwest Jiaotong University,Chengdu 610031,China)

Abstract Based on Zynq chip,the whole system is implemented with hardware and software co-design method. ARM+FPGA is used as internal architecture of the Zynq,so it has both of their advantages,which are flexibility of ARM and parallel processing capability of FPGA. The two advantages are fully used in this system. Image acquisition is implemented by ARM software design. Corner and edge detection is implemented by FPGA hardware design. The ARM processor and FPGA did the data interaction through bus AX14. This system has 3 main functions,which are image acquisition,image feature extraction and image display. The results show that this system implemented by Zynq,which used hardware acceleration algorithm,is 6 to 8 times faster than that of implemented only by ARM processor.

Keywords Zynq,FPGA,Feature extraction,Hardware and software co-design

图像处理技术起源于20世纪20年代,早期图像处理主要以人为对象,以改善图像质量、满足人的视觉效果为目标[1]。20世纪60年代,随着电子计算机技术的发展,人们开始利用计算机来处理图像信息。1964年,美国喷气推进实验室(JPL)利用电子计算机对航天探测器徘徊者7号拍摄的月球照片进行图像去噪和几何校正等处理[2],成功绘制出月球表面地图,直接推动了数字图像处理技术的快速发展。随着研究的深入,人们开始研究如何利用计算机来分析解释图像,类似人类视觉理解外部世界的过程。

随着电子技术的飞速发展,越来越多的传统数字图像处理系统可以通过嵌入式系统来实现。嵌入式系统具有体积小、成本低、功耗低、可靠性强等一系列的优点,基于嵌入式技术的嵌入式图像处理系统已经广泛应用于工业检测[3]、机器视觉[4]、航空航天[5]、军事制导[6]、生物医学[7]、公共安全[8]、汽车驾驶辅助[9]等领域。目前,应用在图像处理领域的嵌入式处理器主要有ARM,DSP和FPGA,而现有的以单个ARM处理器、DSP或者FPGA为核心的嵌入式图像处理方案,已经无法满足高性能应用的要求。采用多个芯片结合的方案,往往又会导致硬件结构复杂、开发难度大以及系统不够稳定等问题。针对上述问题,本文采用Xilinx公司推出的Zynq全可编程平台,该平台是首款将高性能ARM Cortex A9硬核与可编程逻辑FPGA紧密集成到一起的异构芯片,不仅解决了芯片工艺上的难点,更重要的是解决了片内高性能处理器与可编程逻辑之间进行高速数据交互的问题。通过这样的结合,使得Zynq既具备了ARM处理器在事务管理、运行操作系统等方面的优势,又具备了FPGA可并行处理、可动态重配的优势。此外,高层次综合工具Vivado HLS采用软件的方法设计硬件,它的推出明显加快了已有OpenCV图像处理算法的硬件化过程。因此本文基于Zynq全可编程平台,采用软硬件协同设计的方法,实现从IP核设计、硬件系统集成、嵌入式Linux系统移植到顶层应用程序的全系统设计。

1 系统的硬件组成

系统的硬件由图像采集模块、硬件加速模块、存储模块、显示模块以及PS与PL通信模块5部分组成。图像采集模块通过USB摄像头采集图像;存储模块包括DDR存储器和SD卡;显示模块通过HDMI接口输出图像。其中图像采集模块、存储模块在PS端实现,硬件加速模块、显示模块在PL端完成,二者通过互联模块实现通信。系统的硬件结构框图如图1所示。

潘青松(1992-),男,硕士生,主要研究方向为嵌入式系统设计,E-mail:qingsong_pan@163.com;**张 怡**(1970-),女,副教授,主要研究方向为嵌入式系统设计;**杨宗明**(1992-),男,硕士生,主要研究方向为嵌入式系统设计;**秦剑秀**(1994-),男,硕士生,主要研究方向为电力系统及其自动化。

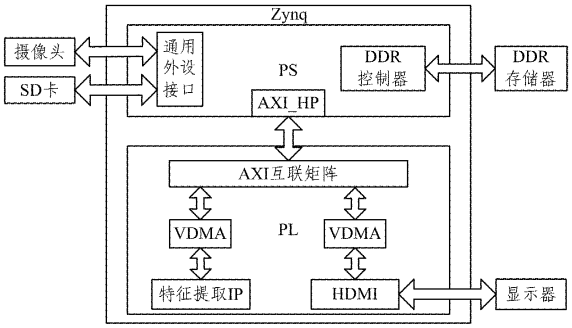


图 1　系统硬件结构框图

2　图像特征提取 IP 核的设计与实现

特征提取指的是使用计算机通过某种方式遍历图像中的每一个像素点，并决定每个像素点是否属于一个图像特征。特征提取的结果是把图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连续的区域。通过对图像特征的提取，可以获得图像中有价值的信息，减少冗余信息。角点和边缘是图像的两个重要特征，在实际中得到了广泛应用。

目前，学术界对图像的角点还没有统一的定义，通常认为角点是图像边缘曲线上曲率值最大的点，或者说是与周围像素值存在较大差异的点[10]。图像中的角点是有价值的点，不仅保留了物体的重要特征，而且其数量远小于原图像中的像素数量。角点能为图像的高层级应用提供有效的信息，如利用角点进行图像匹配、运动估计、目标跟踪等。目前，用于角点检测的算法很多，常用的主要有 Harris，SURF，SIFT，FAST 等。

边缘通常是图像的灰度或者颜色发生剧烈变化的地方，反映了物体的外观轮廓特征。边缘特征广泛存在于目标与背景、目标与目标、区域与区域之间。通过图像的边缘特征，可以对原始图像进行分割，从而分离目标与背景，这是图像理解与分析的前提。图像的边缘检测技术广泛应用于工业检测、图像分割、运动检测、人脸识别和目标跟踪等领域。本文第 4 节对常用边沿检测算法进行了对比。

目前，用于角点和边缘检测的算法很多。一般来说，特征提取效果越好的算法计算也越复杂，在工程应用中难以满足快速性的要求。FAST（Features From Accelerated Segment Test）角点检测算法与 Sobel 边缘检测算法的检测效果较好，且算法复杂度适中，在实际中有较多应用。

2.1　FAST 角点检测的设计与实现

2.1.1　FAST 算法原理

FAST 角点检测算法的目的在于解决快速角点检测的问题。在 Edward Rosten 和 Tom Drummond 于 2006 年发表的“Machine learning for high-speed corner detection”文章中提出了一种 FAST 特征，并在 2010 年对这篇论文做了小幅度的修改后重新发表。Rosten 等人对 FAST 角点进行了定义：若某像素点与其周围邻域内足够多的像素点处于不同的区域，则该像素点可能为角点[11]。首先，定义一个角点响应函数来判断像素点 p 是否为角点，如式(1)所示：

$$N=\sum|I(x)-I(p)|<Th,x\in circlez(p) \tag{1}$$

其中，$circlez(p)$是以 p 点为中心的圆周上点的集合；$I(x)$为圆周上任意一点的图像灰度值；$I(p)$为中心像素点 p（候选点）的图像灰度值；Th 为给定的阈值。

第一步　从图片中选取一个坐标点 p，以选取点坐标为圆心，作半径为 3 的 Bresenham 圆（一个计算圆的轨迹的离散算法，得到整数级的圆的轨迹点）。如图 2 所示，对应圆周上将有 16 个像素点，并对这 16 个像素点依次进行编号 $I(1)$，$I(2)$，…，$I(16)$。

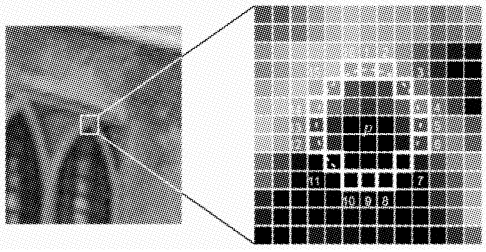


图 2　半径为 3 的 Bresenham 圆

第二步　候选点选取。给定阈值 Th，首先根据式(2)比较 $I(1)$，$I(5)$，$I(9)$，$I(13)$ 4 个像素点与中心点 $I(p)$的像素差值。

$$\begin{aligned}&|I(1)-I(p)|<Th,|I(5)-I(p)|<Th\\&|I(9)-I(p)|<Th,|I(13)-I(p)|<Th\end{aligned} \tag{2}$$

若式(2)中至少 3 个满足，则点 p 成为候选的角点；若不满足，则认为点 p 不是角点。遍历图像中所有点，做完这一步检测后，符合条件的将成为候选的角点。

第三步　候选点判断。候选点是否为角点，需要再做一次完整的测试，即利用角点响应函数依次判断候选点与圆周上点的差值。若满足条件，则 N 的值加 1。当 N 大于或等于 9 时，则可以判断点 p 为角点；否则，点 p 不是角点。

第四步　非极大值抑制。上述检测方法会带来一个问题，就是造成角点的聚簇效应，从而导致多个角点在图像的某一块重复高频率地出现。可以通过非极大值抑制的办法来消除这种状况。在以角点 p 为中心的 3×3 邻域内，若只有一个角点，则保留；若存在多个角点，则计算每个角点的分值 V_i：

$$V_i=\sum_{x=1}^{16}|I_i(x)-I_i(p)| \tag{3}$$

其中，V_i 值表示该邻域内第 i 个角点的中心值 $I_i(p)$与其对应的圆周上的点 $I_i(x)$之差的绝对值总和。V 值较小的角点将会被排除，仅保留 V 值最大的点作为该邻域内的角点。

2.1.2　设计与实现

为了便于观察 IP 核处理的结果，在 FAST 角点检测 IP 核中加入一些额外的操作，标记检测到的角点，图 3 给出了 FAST 角点检测 IP 核的处理流程。

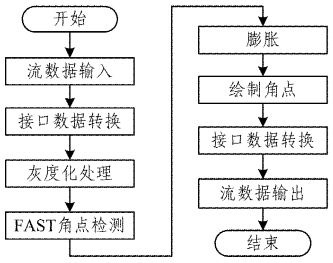


图 3　FAST 角点检测 IP 核的处理流程

2.2　Sobel 边缘检测算法的设计与实现

2.2.1　算法原理

Sobel 算子是一种常用的边缘检测算子，该算子根据像素点上下、左右邻点的灰度加权差，在边缘处达到极值这一现象检测边缘，对噪声具有平滑作用，可提供较为精确的边缘方向信息。根据微分知识，函数的极值点可以通过导数求出。基于微分的图像边缘提取中，图像 $f(x,y)$被视为一个曲面。

图像上任意一点在 x 方向的一阶偏导数为：

$$f_x(x,y)=\frac{\partial f(x,y)}{\partial x} \tag{4}$$

在 y 方向的一阶偏导数为：

$$f_y(x,y)=\frac{\partial f(x,y)}{\partial y} \tag{5}$$

根据一阶导数与梯度的对应关系，$f(x,y)$ 在位置 (x,y) 处的梯度场为：

$$\nabla f(x,y)=(G_x,G_y)=(f_x(x,y),f_y(x,y)) \tag{6}$$

梯度是一个矢量，灰度图像的梯度反映了图像的灰度变化的大小和方向。G_x，G_y 分别表示沿 x 方向和沿 y 方向的梯度分量。根据式(6)，可以获得局部产生的梯度幅值 $|\nabla f(x,y)|$：

$$|\nabla f(x,y)|=|f_x(x,y)|+|f_y(x,y)| \tag{7}$$

将梯度幅值与阈值的对比结果作为图像边缘的判断依据。梯度幅值可通过卷积模板的方式简化计算。图 4 给出了 Sobel 算子计算水平梯度分量 G_x 和垂直梯度分量 G_y 所用的卷积模板。

−1	0	+1
−2	0	+2
−1	0	+1

(a)水平方向

+1	+2	−1
0	0	0
+1	−2	−1

(b)垂直方向

图 4 Sobel 算子卷积模板

图像 $f(x,y)$ 中任意一点 (x,y) 的水平梯度分量 G_x 和垂直梯度分量 G_y 按如下公式计算：

$$\begin{aligned}G_x=&(-1)\times f(x-1,y-1)+0\times f(x,y-1)+1\times f(x+1,y-1)+(-2)\times f(x-1,y)+0\times f(x,y)+2\times f(x+1,y)+(-1)\times f(x-1,y+1)+0\times f(x,y+1)+1\times f(x+1,y+1)\\=&[f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)]-[f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)]\end{aligned} \tag{8}$$

$$\begin{aligned}G_y=&1\times f(x-1,y-1)+2\times f(x,y-1)+1\times f(x+1,y-1)+0\times f(x-1,y)+0\times f(x,y)+0\times f(x+1,y)+(-1)\times f(x-1,y+1)+(-2)\times f(x,y+1)+(-1)\times f(x+1,y+1)\\=&[f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)]-[f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)]\end{aligned} \tag{9}$$

根据式(8)与式(9)可以求得梯度的幅值：

$$G(x,y)=|G_x|+|G_y| \tag{10}$$

最后，将通过式(10)计算得到的梯度幅值与事先设定好的阈值 Th 进行比较。若梯幅值大于阈值 Th，则认为该点 (x,y) 为边缘点。

2.2.2 设计与实现

Sobel 边缘检测 IP 核的处理流程如图 5 所示。

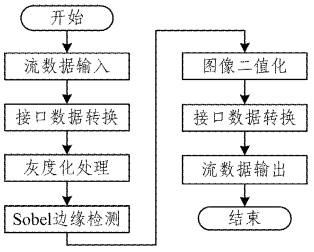


图 5 Sobel 边缘检测 IP 核的处理流程

3 系统软件设计

3.1 嵌入式实时 Linux 操作系统搭建

系统基于 SDK 软件，设计用于启动 Zynq 的 BOOT.bin 文件，并根据硬件系统对设备树文件进行修改，考虑应用层程序的需求，对内核进行必要的配置，最后实现 Linux 系统的移植。

Zynq 全可编程包括了处理器系统与可编程逻辑，启动过程比传统的以 ARM 为核心的处理器多了配置 FPGA 的过程。Zynq 芯片利用片上的 CPU 来帮助配置，在没有外部 JTAG 的情况下，PS 与 PL 都是依赖于 PS 完成芯片的初始化配置。并且 Zynq 芯片支持从多种设备启动，包括 JTAG，NAND，Quad-SPI 以及 SD 卡[12]。启动过程是分阶段进行的，在 Zynq 上加载嵌入式 Linux 系统的启动过程如图 6 所示。

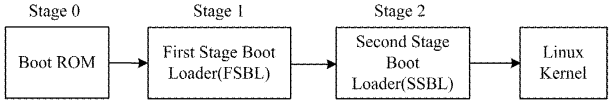


图 6 Linux 系统的启动过程

Stage-0 在系统上电运行后，处理器自动执行芯片生产厂家固化在片内 BootROM 中的代码，完成 CPU、内存以及一些外设的初始化，为下一启动阶段做准备。

Stage-1 加载 FSBL(First Stage Boot Loader)程序代码，将 bitstream 文件写入 PL(配置 FPGA)，继续初始化外设，这一阶段可以由用户修改启动代码进行控制。

Stage-2 通常，这是用于基于 BSP 的裸机程序，也可以是操作系统的启动引导程序 SSBL(Second Stage Boot Loader)。对于嵌入式 Linux 而言，这个阶段就是 u-boot，完全由用户代码控制。

在 Zynq 上移植嵌入式 Linux 操作系统，通过引脚配置可以将 Zynq 配置为 SD 卡启动。从 SD 卡启动所需要的文件包括内核镜像文件 uImage、启动引导文件 BOOT.BIN、设备树文件 devicetree.dtb 以及根文件系统 rootfs。图 7 给出了文件的组织结构。BOOT.bin 是 Zynq 的启动文件，主要包括 zynq_fsbl.elf，system.bit 和 u-boot.elf 3 个文件。在串口终端中，可以看到整个系统的启动过程，成功启动后的结果如图 7 所示。

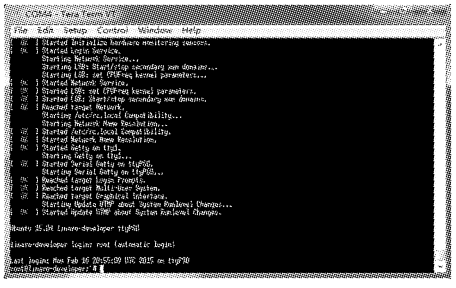


图 7 成功启动后的结果

3.2 视频图像采集 V4L2 驱动

Video for Linux 2(V4L2)，从 Linux2.5.x 版本开始就已经集成到内核中[13]，是 Linux 系统中用于视频设备的内核驱动，它为 Linux 中视频设备访问提供了通用接口，支持市面上常用的视频采集卡、USB 摄像头等视频图像采集设备。

V4L2 驱动所支持的设备是字符设备，主设备号为 81。对于视频设备，其次设备号为 0-63，设备节点为/dev/video *(* 是从 0 开始递增的整数)。V4L2 驱动程序除了提供打

开、关闭、读写等基本操作外，同时也支持中断、内存映射，这些操作都被封装在结构体 video_ device 中。video_device 是 V4L2 驱动程序的核心数据结构，用于在/dev 目录下生成设备节点文件，把操作设备的接口呈现给用户空间。因此，当应用程序对设备文件进行诸如 read，write，mmap 等操作时，Linux 内核最终是通过访问 video_device 结构中的函数来完成的。

3.3　图像处理 IP 核的配置 UIO 驱动

UIO(Userspace I/O)是运行在用户空间的 I/O 技术，将设备驱动的很少一部分运行在内核空间，而在用户空间实现驱动的绝大多数功能。UIO 驱动是在 Linux2.6.23 内核中引入的，使用 Linux 下的 UIO 驱动在用户空间编写驱动程序，能够更好地解决嵌入式系统中非标准外设设备驱动程序开发的问题。与传统的内核空间驱动不同的是，用户空间驱动 UIO 仅支持字符设备。

3.4　图像的显示 Frame Buffer 显卡驱动

Frame Buffer 是 Linux 内核中的显卡驱动程序，是一个字符设备驱动，提供对显卡的最原始操作，主设备号为 29，设备节点是/dev/fb＊(＊是从 0 开始递增的整数)。Linux 内核工作在保护模式下，在嵌入式 Linux 系统中抽象出 Frame Buffer 这个设备，以供用户空间程序直接写屏。Frame Buffer 机制对显卡硬件结构进行抽象，屏蔽掉各种硬件之间的差异，提供统一的 API 接口给上层调用。用户可以将 Frame Buffer 看作是显示内存的一个映像，将其映射到进程地址空间之后，就可以直接进行读写操作，而写操作可以立即反应在屏幕上。

4　系统测试与分析

4.1　系统测试

软硬件设计完成后，就可以进行系统的软硬件联合测试。测试结果如图 8 所示。

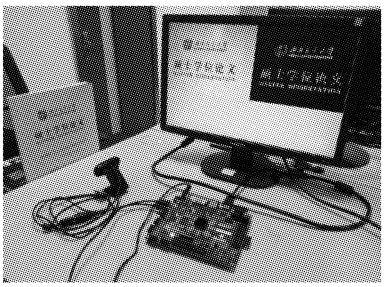


图 8　系统测试结果

图 8 中显示器上左侧为角点检测后的图像，右侧为边缘检测后的图像。

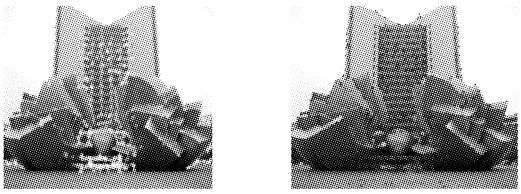
4.2　系统分析

为了比较 OpenCV 软件实现与本文设计的特征提取 IP 模块在特征提取的准确性与时间上的差异，选取车牌、建筑两种不同的场景进行测试，测试所用图像采用分辨率为 640×480 的灰度图像。测试图片如图 9 所示，FAST 角点检测的测试结果如图 10 和图 11 所示。

图 9　测试图片

(a)OpenCV 处理结果　　(b)本系统处理结果

图 10　场景一：车牌(Th=20)



(a)OpenCV 处理结果　　(b)本系统处理结果

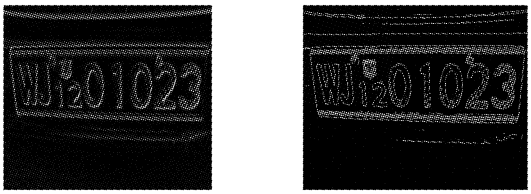
图 11　场景二：建筑(Th=30)

表 1 列出了两种方式实现 FAST 角点检测所需的时间。

表 1　FAST 角点检测耗时比较/us

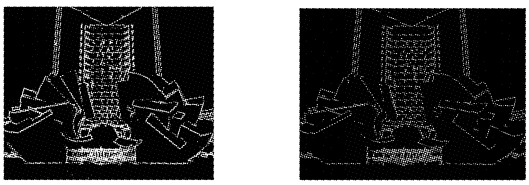
图像帧	OpenCV 耗时	本系统耗时
场景一	43873	5225
场景二	44132	5268
平均	44002	5246

同样地，对两种场景进行 Sobel 边缘检测，比较 OpenCV 软件实现与 Sobel 改进后的硬件实现在处理结果上的差异。测试结果如图 12 和图 13 所示。



(a)OpenCV 处理结果　　(b)本系统处理结果

图 12　场景一：车牌



(a)OpenCV 处理结果　　(b)本系统处理结果

图 13　场景二：建筑

表 2 列出了两种方式的耗时对比。

表 2　Sobel 边缘检测耗时比较/us

图像帧	opencv 耗时	本系统耗时
场景一	30526	5192
场景二	30591	5048
平均	30555	5148

从表 1 中的实验测试结果看出，硬件实现 FAST 角点检测的平均时间为 5246us，OpenCV 软件方式实现角点检测的平均时间为 44002us，采用硬件实现的方式在处理速度上快了近 8 倍。同样，从表 2 可以看出，硬件实验 Sobel 边缘检测相比于软件方式实现快了近 6 倍。实验结果表明，采用基于

(下转第 556 页)

[4] SASTRY G M, ADZHIGIREY M, DAY T, et al. Protein and ligand preparation: parameters, protocols, and influence on virtual screening enrichments[J]. Journal of Computer-aided Molecular Design, 2013, 27(3): 221.

[5] 蔡超前. 分子相似性的计算方法研究[D]. 上海: 华东理工大学, 2013.

[6] LIU X, JIANG H, LI H. SHAFTS: a hybrid approach for 3D molecular similarity calculation. 1. Method and assessment of virtual screening[J]. Journal of Chemical Information & Modeling, 2011, 51(9): 2372-2385.

[7] VOGT M, STUMPFE D, GEPPERT H, et al. Scaffold Hopping Using Two-Dimensional Fingerprints: True Potential, Black Magic, or a Hopeless Endeavor? Guidelines for Virtual Screening[J]. Journal of Medicinal Chemistry, 2010, 53(15): 5707-5715.

[8] LIM H, LEE N, LEE J, et al. Reducing False Positives of a Bloom Filter using Cross-Checking Bloom Filters[J]. Applied Mathematics & Information Sciences, 2014, 8(4): 1865-1877.

[9] MELSTED P, PRITCHARD J K. Efficient counting of k-mers in DNA sequences using a bloom filter[J]. Bmc Bioinformatics, 2011, 12(1): 333.

[10] RATHGEB C, BREITINGER F, BUSCH C. In Alignment-free cancelable iris biometric templates based on adaptive bloom filters[C]// 2013 International Conference on Biometrics (ICB). IEEE: 2013: 1-8.

[11] TANG J, FONG A C M, WANG B, et al. A Unified Probabilistic Framework for Name Disambiguation in Digital Library[J]. IEEE Transactions on Knowledge & Data Engineering, 2012, 24(6): 975-987.

[12] ECKERT H, BAJORATH J. Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches [J]. Drug Discovery Today, 2007, 12(5-6): 225.

[13] CHENG T, LI Q, ZHOU Z, et al. Structure-Based Virtual Screening for Drug Discovery: a Problem-Centric Review[J]. Aaps Journal, 2012, 14(1): 133.

[14] VENKATRAMAN V, PÉREZNUENO V I, MAVIDIS L, et al. Comprehensive Comparison of Ligand-Based Virtual Screening Tools Against the DUD Data set Reveals Limitations of Current 3D Methods[J]. Journal of Chemical Information & Modeling, 2010, 50(12): 2079.

[15] JAHN A, HINSELMANN G, FECHNER N, et al. Optimal assignment methods for ligand-based virtual screening[J]. Journal of Cheminformatics, 2009, 1(1): 14.

[16] RAMIREZ-MANZANARES A, PEÑA J, AZPIROZ J M, et al. A hierarchical algorithm for molecular similarity (H-FORMS) [J]. Journal of Computational Chemistry, 2015, 36(19): 1456.

[17] HU G, KUANG G, XIAO W, et al. Performance evaluation of 2D fingerprint and 3D shape similarity methods in virtual screening[J]. Journal of Chemical Information & Modeling, 2012, 52(5): 1103.

[18] GARDINER E J, HOLLIDAY J D, O'DOWD C, et al. Effectiveness of 2D fingerprints for scaffold hopping[J]. Future Medicinal Chemistry, 2011, 3(4): 405.

[19] JAIN A N, NICHOLLS A. Recommendations for evaluation of computational methods[J]. Journal of Computer-aided Molecular Design, 2008, 22(3/4): 133.

[20] HESSLER G, BARINGHAUS K H. The scaffold hopping potential of pharmacophores[J]. Drug Discovery Today Technologies, 2010, 7(4): 263-269.

[21] VAINIO M J, KOGEJ T, RAUBACHER F, et al. Scaffold Hopping by Fragment Replacement[J]. Journal of Chemical Information & Modeling, 2013, 53(7): 1825-1835.

[22] CLARK R D, WEBSTER-CLARK D J. Managing bias in ROC curves[J]. Journal of computer-aided molecular design, 2008, 22(3/4): 141.

(上接第 533 页)

Zynq 的软硬件协同设计方式实现图像特征提取相比 ARM 实现在处理速度上提高数倍，起到了硬件加速的效果。

结束语 系统以基于 Zynq 的图像角点及边缘检测系统为研究课题，以嵌入式技术为载体，以系统设计为目标，实现了图像处理技术与嵌入式技术的结合。系统采用 FPGA+ARM 的硬件架构，以及软硬件协同设计的方法，在 FPGA 中实现 FAST 角点检测模块和 Sobel 边缘检测模块，在 ARM 中完成图像的采集。充分结合二者的优势，最终完成集图像采集、特征提取硬件加速、图像显示为一体的全系统设计。实验结果表明，本系统具有较好的检测精度和较快的处理速度，能够为上层处理提供一个较理想的平台，具有一定的工程使用价值。本文在 Zynq 上实现了预定设计方案，但也存在不足之处，图像特征提取是机器视觉的前提，本文所设计的系统并未被应用到具体的领域。在未来的工作中，结合 OpenCV 视频处理函数库，可以在 Zynq 上做更高层的机器视觉应用，比如：基于角点特征的图像匹配、基于边缘特征的图像分割等。

参 考 文 献

[1] 陈军. 基于 ARM-Linux 的嵌入式产品平台构建[D]. 杭州: 浙江大学, 2004.

[2] 唐敏. 基于边缘和角点的特征提取方法与应用研究[D]. 长沙: 国防科技大学, 2006.

[3] 陈勇. 基于机器视觉的表面缺陷检测系统的算法研究及软件设计[D]. 天津: 天津大学, 2006.

[4] 陈然. 工业相机的自动调焦与图像拼接研究及在桥梁检测机器人中的应用[D]. 广州: 华南理工大学, 2014.

[5] 王阿妮, 马彩文, 晁长征, 等. 基于边缘相关的红外与可见光图像配准方法[J]. 现代电子技术, 2009, 32(10): 104-106.

[6] 尚春红, 赵明昌. 复杂背景图像中军用靶子识别算法研究[J]. 计算机应用, 2008, 28(5): 1257-1260.

[7] 宋余庆. 数字医学图像[M]. 北京: 清华大学出版社, 2008.

[8] ARBELAEZ P, MAIRE M, FOWLKES C, et al. Contour detection and hierarchical image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 33(5): 898-916.

[9] SCHAPIRE R E. The boosting approach to machine learning: an overview[C]// MSRI Workshop on Nonlinear Estimation and Classification. Berkeley, CA, 2002: 312-318.

[10] 薛金龙. 基于角点的图像特征提取与匹配算法研究[D]. 大连: 大连理工大学, 2014.

[11] ROSTEN E, DRUMMOND T. Machine learning for high speed corner detection[C]// 9th European Conference on Computer Vision. 2006: 430-443.

[12] 陆启帅, 陆彦婷, 王地. SoC 与嵌入式 Linux 设计实战指南: 兼容 ARM Cortex-A9 的设计方法[M]. 北京: 清华大学出版社, 2014: 274-278.

[13] 杨霞. 基于 OMAP3530 以太网视频采集系统的设计与实现[D]. 南京: 南京邮电大学, 2011.