基于 DM642 的人脸检测系统的设计与实现

重庆大学硕士学位论文

学生姓名：聂江浩

指导教师：张小洪　　副教授

专　　业：计算机软件与理论

(软件工程领域)

学科门类：工　学

重庆大学软件工程学院

二 OO 九年十月

Design and Implementation of Face Detection System Based on DM642

A Thesis Submitted to Chongqing University

in Partial Fulfillment of the Requirement for the

Degree of Master of Engineering

By

Nie Jianghao

Supervised by Ass. Prof. Zhang Xiaohong

Major: Computer Software and Theory

(Software Engineering)

College of Software Engineering of

Chongqing University, Chongqing, China

October 2009

摘　　要

人脸检测技术一直是计算机视觉研究领域的一大热点，其目标是在给定的静态图片或视频序列中判断是否存在人脸，如果有则找出并确定所有人脸的位置和大小。人脸检测作为所有人脸处理技术的基础步骤，在身份识别、视频监控、基于内容的检索、新一代人机界面等领域有着日益广泛的应用。目前人脸检测技术的主流产品都是基于通用的台式电脑，其庞大的体积、巨大的功耗和不稳定的性能都是限制该技术进一步产业化的因素。根据这一现状，本课题设计并实现了一套基于 DSP(Digital Signal Processor)的嵌入式人脸检测系统。本文主要工作如下：

（1）人脸检测算法可以分为基于先验知识的检测算法和基于统计模型的检测算法两类。本文通过分析这两类算法，并结合系统硬件条件，采用基于统计模型的 Adaboost 算法，它是一种基于积分图和矩形特征的分级分类器算法,具有较高的检测效率和鲁棒性。同时在详细介绍了 Adaboost 算法之后，本文通过大量实验，选择较好的性能参数。

（2）在研究过程中，本文从硬件和软件两方面对系统进行了规划。硬件方面，本文选用 TI 公司的 TMS320DM642 芯片作为主控芯片，辅以摄像头、CPLD、视频编解码芯片、储存芯片和电视机搭建了系统硬件平台。软件方面，本文基于 CCS(Code Composer Studio)3.1，利用其集成的实时操作系统 DSP/BIOS 进行软件开发，在 RF5(Reference Framework 5)框架下，将工作流程分为图像采集、处理和显示三个任务，任务间使用消息进行同步通信并传递图像数据。另外，本文在算法实现过程中裁剪并移植了开源计算机视觉库 OpenCV，并利用其加快开发速度。

（3）系统依照 TMS320C6000 系列芯片的开发流程进行开发，并利用 CCS 提供的各类调试工具对 C 语言代码进行分析调试，对需要优化的部分根据 DM642 的硬件特点采用浮点转定点运算、使用 EDMA 传输等方法进行优化，以加快处理速度。

实验结果表明，本系统能较为准确的检测到人脸，并基本满足实时性要求，这为今后人脸检测技术的进一步应用奠定了一定的基础。

关键词：人脸检测，Adaboost，DM642，OpenCV，性能优化

ABSTRACT

Face detection has always been a hot topic in the computer vision areas and its goal is to identify and determine the location and the region of the faces in the static images or video sequences. As the basic step of all face processing technologies, face detection has been widely used in the areas of identification, video surveillance, content-based retrieval and new generation of human-machine interactions as far. Most of the current face detection products have been based on the desktop computer and they have some common disadvantages such as vast sizes, high power consumptions and unstable systems, which prevented them from being widely used. According to this situation, this subject designs and implements an embedded face detection system based on DSP (Digital Signal Processor). The main contributions of this thesis are as follow:

(1) It is well-known that face detection algorithms can be divided into two types: one is the knowledge-based algorithms and the other is the algorithms based on statistical models. By analyzing the current types of face detection algorithms and the system's hardware conditions, this thesis utilizes the Adaboost face detection algorithm based on statistical models. The Adaboost algorithm with high detection efficiency and robustness employs integral images, rectangle features and cascade classifiers. After describing the Adaboost algorithm in detail, this thesis optimizes its parameters by a large amount of experimentations.

(2) The thesis plans out the system’s general framework of hardware and software in the researching procedures of this system, which chooses TI's TMS320DM642 chip as the master chip to construct the face detection platform with the CCD camera, CPLD, video codec chips, memory chips and television. This thesis develops the face detection software system based on the real-time multi-task operating system DSP/BIOS in the platform of CCS (Code Composer Studio) 3.1. In RF5 (Reference Framework 5), the thesis divides the system work flow into the following three tasks: capture, process and display, and uses messages to synchronize and communicate among tasks. Besides these, the thesis cuts and transplants the open source computer vision library OpenCV and this will speed up the development progress.

(3) The thesis designs the code in accordance to the software development flow of TMS320C6000 chips, and then uses the debug tools provided by the CCS to analyze the performance of C code. In the following, according to the hardware property of the DSP,

this thesis transforms the floating-point calculation into fixed-point calculation and utilizes the EDMA to optimize the processing speed.

The experimental results show that the designed system can detect the human face accurately and basically meet the real-time requirements. This will lay a certain foundation for the future applications of face detection technology.

Keywords: Face Detection, Adaboost, DM642, OpenCV, Optimization

目　录

中文

1 绪　论

1.1 课题研究的背景和意义

生物特征识别技术是一种基于人体的生理特征或行为特征进行个人身份鉴定的自动化技术。常用于生物特征识别技术的特征有手形、指纹、面相、虹膜等，行为特征有说话、签字等。基于这些特征，人们已经发展了手形识别、指纹识别、人脸识别、虹膜识别、语音识别、笔迹识别等多种生物特征识别技术。由于生物特征与个人身份具有一一对应的关系，其真实性和可靠性毋庸置疑，因此许多国家都把生物特征识别技术作为重大基础战略技术加以研究。

人脸识别技术作为当前生物特征识别技术的一大研究热点，以其独特的主动性，非接触性和用户友好性受到研究人员的青睐。人脸识别的技术研究起源于上世纪六十年代末，早期的人脸识别都是在默认已经得到了一个正面人脸或者人脸数据很容易获得的前提下进行的。但是随着利用人脸信息的研究范围不断扩大和开发实际系统需求的不断提高，研究人员发现，这种假设下的研究并不能满足实际需求。于是人脸检测作为一项独立的研究内容开始发展起来。

人脸检测的目标是从一副图片或者一帧视频中确定是否包含人脸，如果有则标定出所有人脸的位置、大小和姿势。其基本思想是运用知识或者统计的方法对人脸建立模型，比较所有的待检测图像区域与人脸模型的匹配度，从而确定图像中人脸的区域[1]。

随着人脸检测技术的发展，人脸检测的应用领域也在逐步扩大，在许多场合都只需要检测到人脸的存在，而无需进行人脸身份的识别。比如说疲劳驾驶预警系统，确定人脸位置后，通过计算一段时间内眼睛的闭合时间来判断驾驶员的疲劳状况，以此作为预警的依据。另外在视频监控或者运动分析领域，可以通过人脸位置的变化来分析判断一个人的行为等。这些都是人脸检测技术的可应用范围，并且对人脸检测提出了实时性，高鲁棒性等要求。人脸检测在人脸处理技术中的位置和作用如图 1.1 所示。

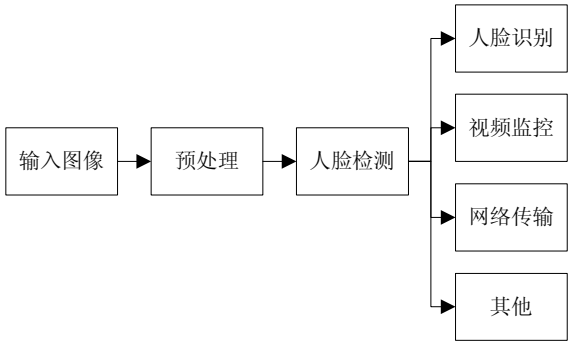


图 1.1 人脸检测在人脸处理技术中的位置

Fig 1.1 Position of face detection in the face processing technology

数字信号处理器（Digital Signal Process，DSP）是数字信号处理技术的核心器件。自从 1982 年德州仪器公司（Texas Instruments，TI）推出了第一块 DSP 芯片起，DSP 就以其特有的稳定性、可重复性、集成方便、特别是可编程性高和易于实现自适应处理等特点，给数字信号处理技术带来了巨大的变革，并被广泛应用于工控、通信、医疗、消费类电子产品等领域。特别是近年来随着半导体工艺的发展和微计算机体系结构的改进，DSP 芯片的功能越来越强大，48 亿次的高速处理能力以及出色对外接口能力，使得计算机视觉信号的处理已经成为可能，大部分的非实时应用也具备了实时应用的硬件基础。

总的来说，人脸检测技术具有相当广阔的前景，它正从热点研究范围内逐步渗透到人们的日常生活中。而将嵌入式 DSP 技术和人脸检测技术集合起来，具有较高的创新意义和实用价值。

1.2 国内外研究现状

1.2.1 人脸检测技术的发展现状

人脸检测技术是所有人脸应用技术的基础，它最早被提出于人脸识别技术的定位环节，近年来由于在视频监控、安防、基于内容的检索等领域的应用价值，开始作为一个独立的研究课题受到普遍重视。

目前国外有很多针对人脸检测的研究机构，其中著名的大学有卡内基梅隆大学（CMU）、麻省理工学院（MIT）、洛克菲勒大学（Rockefeller）、康奈尔大学（Cornell）等。美国军方每年组织的人脸识别大赛则是大大促进了人脸检测和人脸识别的研究.。国内的人脸检测起步相对较晚，目前，国内清华大学、浙江大学、南京理工大学，哈尔滨工业大学、中科院计算所和自动化所等一些研究机构针对人脸检测和人脸识别的研究工作展开得较好。2005 年清华大学电子系人脸识别课题组负责人苏光大教授主持承担的国家“十五”攻关项目《人脸识别系统》通过了由公安部组织的专家鉴定，达到了国内领先水平和国际先进水平。国内也有许多公司在开发人脸检测和识别产品，比较成功的有上海银晨科技股份公司和北京行者数码控股有限公司等，特别是后者的行者人脸识别系统在 2008 年北京奥运会期间得到了大规模的应用。

随着人脸检测技术的发展，国际上发表的相关论文数量也大幅度增长，如 ICFG（IEEE International Conference on Automatic Face and Gesture Recognition）、CVPR（Conference on Computer Vision and Pattern Recognition）、ICIP（International Conference on Image Processing）等重要国际会议上每年都有大量关于人脸检测的论文，其数量占到了人脸研究论文的约 1/3。而且 MPEG7 标准组织也已经建立了人脸识别草案小组，人脸检测算法也是一项征集的内容。

1.2.2 DSP 的发展现状

数字信号处理是指通过一个模拟向数字的转化（A/D 转化），将现实生活中的模拟信号转化成数字信号传给处理器，经过计算处理后，再将数字信号转化成模拟信号（D/A 转化）的过程。DSP 正是为了高效低耗地实现这一过程而产生的。

DSP 的发展可大致分为三个阶段：

在 DSP 出现以前，人们只能使用通用微处理器（Micro Process Unit，MPU）来处理数字信号。直到上世纪七十年代才有人提出 DSP 的理论和算法基础。一般认为，世界上第一个单片 DSP 芯片是 1978 年 AMI 公司发布的 S2811。1979 年美国 Intel 公司发布的商用可编程器件 2920 是 DSP 芯片的一个重要里程碑。这两种芯片内部都没有现代 DSP 芯片所必须有的单周期乘法器。1980 年，日本 NEC 公司推出的 μPD7720 是第一个具有硬件乘法器的商用 DSP 芯片，从而被认为是第一块单片 DSP 器件。但是这个时期的 DSP 系统都是由分立的元件组成，应用也仅限于军事和航天等部分。

随着大规模集成电路技术的发展，1982 年 TI 公司推出了世界上第一代 DSP 芯片 TMS32010。这种 DSP 器件采用 NMOS 技术制作，虽功耗和尺寸稍大，但运算速度却比 MPU 快了几十倍。随着 CMOS 技术的进步与发展，第二代基于 CMOS 工艺的 DSP 芯片应运而生，其存储容量和运算速度成倍提高，成为语音处理、图像硬件处理技术的基础，如 TI 公司的 TMS32020，日本 Fujitsu 公司推出的 MB8764 等。八十年代后期，第三代 DSP 芯片如 TI 公司的 TMS320C30 系列问世，运算速度进一步提高，其应用于范围逐步扩大到通信、计算机领域。

九十年代 DSP 发展最快，TI 公司相继推出第四代 DSP 芯片 TMS320C40/C44、第五代 DSP 芯片 TMS320C5X/C54X、第二代 DSP 芯片的改进型 TMS320C2XX、集多片 DSP 芯片于一体的高性能 DSP 芯片 TMS320C8X 以及目前速度最快的第六代 DSP 芯片 TMS320C62X/C67X 等。

经过近三十多年的发展，DSP已经广泛应用于通信、工控、医疗、个人消费类电子产品等领域，并且将继续沿着高性能，低功耗，加强融合和扩展应用领域的方向不断发展。我们有理由相信，DSP产品将会在人们的日常生活学习中扮演越来越重要的角色。

1.3 论文主要内容及结构安排

本课题的目标是基于 TMS320DM642 DSP 芯片设计和实现一个人脸检测系统。按照算法，硬件和软件三大部分的实现，将本文结构安排如下：

第一章：基于在课题研究中查阅的资料，介绍人脸检测技术和数字信号处理技术产生的背景和意义，概述了国内外发展动态；

第二章：基于人脸检测特定的问题和现象，介绍分析人脸检测技术中的经典算法；

第三章：详细介绍本系统采用的 Adaboost 人脸检测算法及其实现；

第四章：介绍本系统的硬件设计方案；

第五章：阐述基于 DSP 的软件设计开发方法；

第六章：论述本系统的实现方案以及优化方法；

第七章：总结本文所作的工作，指出本系统的不足之处和待改善的地方，并展望未来工作。

2 人脸检测算法综述

2.1 人脸检测问题分类

人脸检测包括的内容非常广泛[2]，从不同的角度有多种分类方式，相应需要注意的地方也有所侧重，如表 2.1 所示。

表 2.1 人脸检测问题分类及其侧重点

Tab 2.1 The classification and key point of face detection problems

分类角度	类别以及侧重点	
图像来源	静态图片（包括数字图片、数码照片等，算法的侧重点在于普遍性和鲁棒性）	动态视频（包括监控录像，影视资料等，算法的侧重点在于检测速度）
图像颜色	彩色（算法多注重肤色特征）	灰度（算法多注重能代表人脸的本质特征）
人脸姿态	正面（人脸的正面图像或正面旋转图像）	侧面（包括人脸的俯仰、侧面等图像）
人脸数目	单个（已知人脸个数的特例）	未知（需判断有无人脸，检测各个位置，大小的人脸）
图像背景	简单（无背景或背景比较简单）	复杂（指背景的类型和特征不受约束，某些区域可能在色彩、纹理等特征上与人脸相似）

总的来说，人脸检测技术涉及到以下几个需要解决的问题：

（1）描述：如何描述一个人脸？是基于人脸的几何特征来描述一个人脸，还是基于统计特征来描述？

（2）搜索方法：如何来确定人脸的位置？对于灰度图像，一般采用的是遍历搜索整张图像的方法，另外有些算法为了提高速度，采用层叠分类器对图像进行筛选，先通过简单的分类器排除大部分非人脸区域，然后在较小的区域内遍历搜索。对于彩色图像，则可以通过分析人脸的肤色信息，来排除掉非人脸区域。

（3）大小：如何检测图像中不同尺寸大小的人脸？在基于局部特征的检测方法中，人脸是通过特征点以及相应特征点的位置关系来确定的，那么检测不同尺寸大小的人脸就转化成判断是否存在满足相对位置关系的特征点。在基于人脸整体的检测方法中，一般通过分类器来判别是否存在人脸，而分类器又常常是针对某个尺度的，所以常用的做法是将需要判别的候选区域进行缩放，以达到分类器

所需要的尺度为标准。

（4）速度：如何提高算法的速度？怎么合理的选择人脸的描述方法，搜索方法和检测方法来提高算法的处理速度？

（5）精度：如何提高人脸识别的准确性？通常算法的速度和精度是相互矛盾的，所以要根据具体实现的要求，来合理的平衡两者的关系。

2.2 人脸检测经典算法

由于人脸检测问题分类众多，需要考虑的方面也很广[1][2]，到目前为止，还没有一种通用的方法能解决众多情况下的人脸检测问题。综合查阅的文献资料，人脸检测的方法可以大致分为基于先验知识的检测方法和基于统计模型的检测方法两类，各自的细分如图 2.1 所示。

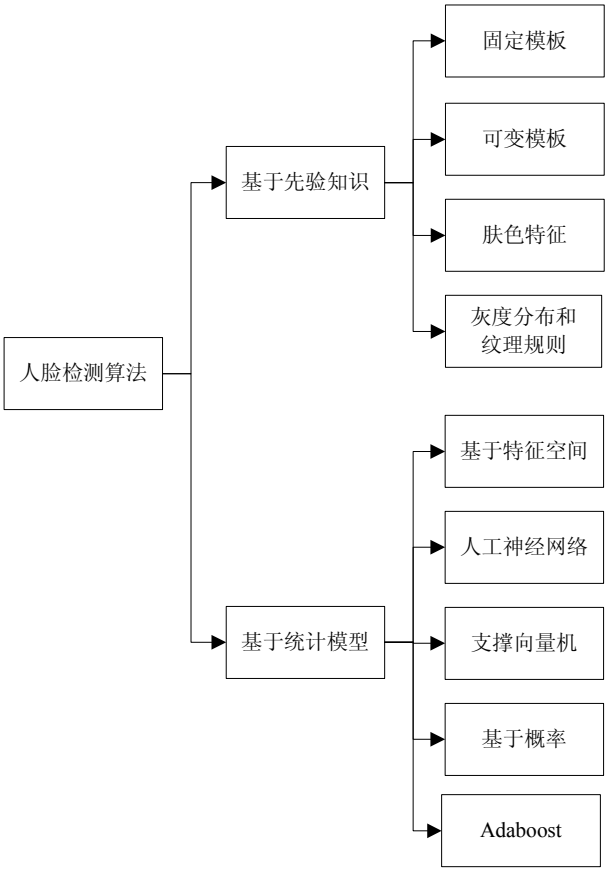


图 2.1 人脸检测算法分类

Fig 2.1 Classification of face detection algorithms

2.2.1 基于先验知识的检测算法

基于先验知识的人脸检测算法源于人们对人脸信息的认识，比如人脸的几何特征、灰度纹理特征、肤色特征等。研究人员利用这些知识，总结出一些规则来描

述人脸和它们之间的关系。具体算法可分为：

①基于固定模板的检测算法

此类算法根据人脸的先验知识，先确定人脸轮廓模板以及各个器官特征的子模板，通过计算图像区域和人脸轮廓模板的匹配度以确定候选人脸区域，然后利用器官特征的子模板验证上一步确定的候选人脸区域是否为人脸。T.Saka 等使用脸部轮廓，眼睛，鼻子，嘴巴等子模板建立人脸模型，对照片中的正面人脸进行检测[3]。梁路宏等考虑眼睛在描述人脸过程中的重要意义，先使用双眼模板进行粗匹配，然后使用不同长宽比的人脸模板进行筛选，最后使用马赛克规则进行验证[4]。

基于固定模板的检测算法表现直观、易于实现、计算量小，但是也存在着很明显的缺点：检测结果受图像噪声的影响很大，而且由于表情等原因，人脸的特征变化很大，很难找到共同的模板来描述人脸的共性。

②基于可变模板的检测算法

此类方法的模板中包含一些可变的参数，根据测试样本的具体情况，能够自适应的进行调整，以提高模板的适应性和检测精度。这类算法原理上与固定模板匹配相同，都是通过计算检测样本和参考模板之间的度量差值是否大于某个阈值来判断测试样本是否为人脸，只是其中包括一个可变的参数模板。如 Govindaraju 等使用变形模板匹配头顶轮廓线和左右两条面颊轮廓线，实现人脸定位[5]。

基于可变模板的检测算法由于包含了一些可变参数，引入自适应机制，使得模板能够检测不同大小，具有不同偏转角度的样本。但是同样的问题在于很难设计出一个通用的模板，另外由于需要动态的调整的参数，计算样本和模板间的差值，此类算法计算量偏大，时间较长。

③基于肤色特征的检测算法

肤色是人脸的重要信息，它不依赖于面部的细节特征，在颜色空间中的分布相对比较稳定，并且可以和大多数背景图像区分开来。虽然由于种族，地域等原因，人脸肤色在人眼看来是不同的，但通过研究发现，这些不同主要表现在亮度方面，除去亮度因素，肤色特征在色度空间上具有明显的聚类性，因此肤色特征对于旋转、表情变化等情况都可以适用，具有广泛的实用性。

肤色特征一般使用肤色模型来表示，检测时首先应该建立肤色模型，然后根据使用肤色模型检测出的肤色像素在色度上的相似性和空间上的相关性分割出可能的人脸区域，最后根据区域的几何特征或灰度特征与人脸特征进行匹配，以排除其它色彩类似肤色的区域。对于一些简单的情况，仅根据肤色像素的聚集特征就可以完成区域分割，如 Yoo 等利用肤色像素的连通性进行区域分割，然后使用椭圆拟合各个区域，根据椭圆长短轴的比率判断其是否为人脸[6]。而对于复杂的情况，则必须考虑光照以及背景的类肤色区域等因素对判断肤色连通区域的影响。Garcia

等将肤色在色度空间分化为不同的类型，将同一类型且位置相邻的肤色像素聚类为区域，再根据区域的几何特征和色调相容性进行归并，归并过程中利用小波特征进行区域验证[7]。Yang 等根据色度的一致性和空间距离将肤色像素聚类成区域，然后逐步归并直到获得满足一定先验知识的椭圆区域为止，最后检查区域中由眼睛、嘴巴等特征形成的暗区或空洞，以确定是否为人脸[8]。

基于肤色特征的检测方法不依赖面部的具体特征，对于旋转、表情变化等特殊情况都能适用，具有较好的适用性。判断出彩色图像中哪一个区域是肤色区域，就可以排除掉在灰度图像中很像人脸区域而在彩色图像中却根本不是肤色的区域。此类算法较为简单，计算量相对较小。缺点在于彩色图像中的色彩信息很容易受到光照、光源色彩、背景图像等外在因素的影响，鲁棒性不强，较难找到一种普适性良好的肤色模型。

④基于灰度分布和纹理规则的检测算法

人脸的核心区域具有明显的灰度分布特征，如眼睛的灰度值明显要比额头或者脸颊的灰度值低，鼻梁的灰度也相比两侧要高。而且人脸局部特征的分布总是存在着一定的规律，例如人眼总是对称分布在人脸的上半部分，鼻子和嘴唇中心点的连线基本与两眼之间的连线垂直等。于是，可以利用一组描述人脸的局部特征分布的规则来进行人脸检测，当满足这些规则的图像区域找到后，则认为一幅人脸已被检测出来，然后可以对候选的人脸区域进行进一步的验证，以确定候选区域是否为人脸。如 Yang 等提出的基于镶嵌图（mosaic image）的检测方法[8]。Yang 等将人脸区域分别划分为 4×4 和 8×8 的马赛克块，然后根据一定的规则进行检验，最后利用人脸器官的边缘特征进一步验证[9]。卢春雨等对镶嵌图法进行改进[10]，将人脸按照器官的分布规律划分为 3×3 个可自适应调整大小的马赛克块，并使用基于灰度和梯度统计特征的规则来验证待检测区域是否为人脸。Miao 等在水平方向提取可能对应眉毛、眼睛、嘴等器官的马赛克边缘后，通过计算所有边缘的重心，与重心模板进行匹配，最后利用灰度和边缘特征验证匹配的结果[11]。

人脸的纹理特征分为皮肤、毛发和其它类等 3 种，这些独特的纹理可以被用来区别其它物体。Augusteijn 等利用是否类似人脸纹理来推断图像中是否有人脸存在[12]。Dai 等利用空间灰度共生矩阵（SGLD）纹理图信息作为特征进行低分辨率的人脸检测[13]。

利用人脸五官特征及其分布规则的方法能在一定程度上适用于具有复杂背景的人脸检测，同时具有较高的检测速度，但是要想进一步提高其适用性，则需要综合更多的人脸特征。实际上这就是一个如何理解人脸图像的问题，它也是所有基于先验知识的方法进一步发展所需要解决的关键性问题。

2.2.2 基于统计模型的检测算法

基于统计模型的检测算法通过对人脸样本和非人脸样本的学习，采用统计学上的模式识别方法，总结出能代表人脸的相关特征，从而建立一个能够区分人脸和非人脸的分类器。检测时对整张图像进行遍历搜索，用训练出的分类器对每一个待检测区域进行人脸检测，如果检测到人脸，则标定出人脸的位置和大小。具体的算法可分为：

①基于特征空间的人脸检测算法

这类算法的基本思想是将人脸区域投影到某一特征空间，然后根据人脸在特征空间的分布情况判断待测样本是否为人脸。

主成份分析（Principal Components Analysis，PCA）算法是这类算法的代表。它对图像根据统计特性做正交变换，以消除原图像向量各分量间的相关性。变换得到对应特征值依次递减的特征向量，这些特征向量即为特征脸。最后根据图像与特征脸的匹配度来判断是否为人脸。Moghaddam 等[14]根据人脸在特征脸空间的投影比较密集这一特性，利用前若干张特征脸将人脸向量投影到主元子空间，并且还考虑与其正交的补空间，使用 DIFS（Distance In Feature Space）和 DFFS（Distance From Feature Space）在这两个空间内进行度量，取得了较好的效果。

基于特征空间方法的还有因子分解（Factor Analysis, FA）和 Fisher 准则方法（Fisher Linear Discriminant analysis, FLD），值得提出的是，Fisher 脸能压制图像之间与识别信息无关的差异，在光照变化时鲁棒性较好。

②基于人工神经网络的人脸检测算法

人工神经网络（Artificial Neural Network，ANN）算法是通过建立一个网络模型，将抽象的统计特征信息包含在神经网络的结构和参数中。由于人工神经网络对于解决复杂的整体的难以描述的问题具有独特的优势，很多研究人员将神经网络应用到人脸检测领域中。

Rowley 等[15]设计了一个使用两类神经网络来进行人脸检测的框架，如图 2.2 所示。第一类神经网络用于判断图像中人脸的姿势，并作为使用下一类神经网络的依据；第二类神经网络分别对应检测正面、侧面和半侧面人脸，最后将结果提交输出。另外此框架采用自调整的策略，对分类器一边训练，一边测试，对于测试过程中分类错误的结果作为反例样本加入到下一次训练过程中，这样能减少样本集的大小，并且逐步提高神经网络的分类能力。

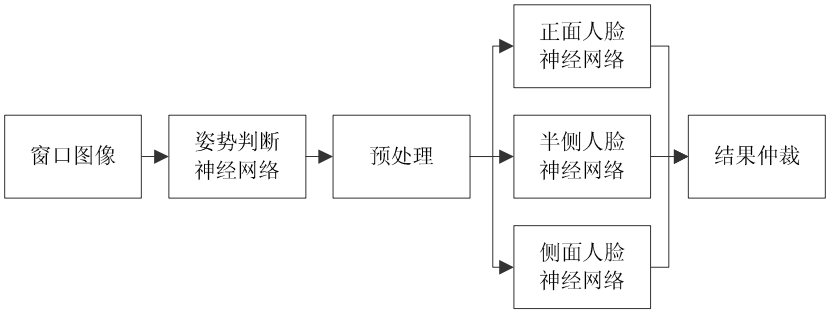


图 2.2 Rowley 神经网络框架

Fig 2.2 The framework of Rowley's neural network

在第二类神经网络中，对于正面姿势的人脸图像只采用正面人脸神经网络进行检测，框架采用一个 3 层的前向型神经网络，输入层接收 20×20 像素的图像数据；隐藏层节点分为若干组，以对应不同的人脸区域；输出层节点输出 1 到-1 区间的值表示这个区域是否为人脸。Rowley 等[16] [17]使用相同的人脸样本和通过自适应过程收集的"非人脸"样本训练了多个正面人脸检测神经网络，并对它们的检测结果进行对比，以进一步减少误检率。

基于人工神经网的方法还有 Juell 等[18]和 Kouzani 等[19]提出的基于人脸器官检测的多级网络方法，以及 Anifantis 等提出的双输出人工神经的检测算法[20]等。

③基于支撑向量机的人脸检测算法

支持向量机（Support Vector Machine，SVM）法是在统计学习的理论基础上发展起来的一种新的模式识别方法，它基于结构风险最小化原理（Structural Risk Minimization Principle，SRM），常用于分类和回归问题。

Osuna 等首先将 SVM 方法用于人脸检测问题[21]，其区分"人脸"和"非人脸"区域的基本思路是使用 SVM 对每一个检测窗口进行分类。在训练过程中，使用了大量的人脸样本和采用自调整方法收集的"非人脸"样本，并通过逼近优化来减少支持向量的数量。梁路宏等[22]采用 SVM 与模板匹配相结合的方法，在模板匹配限定的子空间内采用自调整方法收集"非人脸"样本进行训练，降低了训练的难度和最终支持向量的规模，使得检测速度大大提高。

④基于概率的人脸检测算法

基于概率模型的方法主要分两种：

一种方法的思路是根据计算输入区域属于人脸模型的后验概率，对所有可能的图像窗口进行判别。Schneiderman 等利用贝叶斯原理将后验概率估计转化为一个似然度求解问题，提出了一种基于后验概率函数估计的人脸检测器[23]。Schneiderman 等还将此方法应用于检测正面旋转人脸和侧面人脸，同时采用由粗到精进行搜索和多分辨率信息复用的策略来加快检测速度[24]。属于这一类概率模

型方法的还有 Weber 等提出的视点不变性学习(Viewpoint-Invariant Learning，VIL)方法[25]等。

另一种概率模型是用于描述信号统计特性的隐马尔可夫模型(Hidden Markov Models, HMM)，目前该方法也被应用于人脸检测与识别。由于正面人脸由上到下的五个区域（头发、额头、双眼、鼻子、嘴）具有固定的顺序，Nefian 等使用一个包含五种状态的一维连续 HMM 将头部图像按照这五个区域划分为互有重叠的条块，对各块进行 K-L 变换，选取前若干个变换系数作为观测向量训练 HMM[26]。Nefian 等还提出了嵌套的 HMM 人脸检测方法[27]，该方法同时考虑到人脸由左到右各个特征的自然顺序，使用二维 HMM，并且采用二维 DCT 变换的系数作为观察向量。属于这一类概率模型的方法还有 Meng 等采用的使用 HMM 描述人脸的小波特征中不同级间的相关性等方法[28]。

⑤Adaboost 人脸检测算法

这是新近提出来的一种基于积分图和举行特征的算法，也是本课题采用的算法。该算法将在下一章详细介绍。

2.2.3 基于统计模型的检测算法总结

基于统计模型的检测方法是解决复杂的人脸检测问题的有效途径，大多适用于复杂背景图像中的人脸检测，它具有如下优点：

①不依赖于人脸的先验知识和参数模型，可以避免由于不完整或不准确的知识造成的检测错误；

②采用了样本学习的方法获取人脸模型的参数，统计意义上更为可靠；

③通过增加学习的实例可以扩充检测模式的范围、提高检测系统的鲁棒性。

但是此类算法的缺点同样存在：

①过于依赖训练样本。由于基于统计模型的算法是从样本集中建立出一个能够区分人脸和非人脸的分类器，所以对样本的依赖性较强。目前非人脸的样本选取是一个较难的问题，一些人脸和非人脸样本即使是人也难以区别，在受图像质量和检测速度等因素的约束下，提高图像的分辨率又很困难。因此此类算法在要求一个较高质量的训练样本和测试样本的情况下，可以尝试与先验知识相结合，充分利用人脸的知识模型，以提高检测率和检测速度；

②算法速度相对较慢。基于统计模型的算法需要对图像进行遍历搜索，以检测每一个可能是人脸的区域，这样使得计算复杂度和计算量大大提高；另外许多算法（如基于神经网络和支撑向量机的算法）采用自调整的方法将上一次分类器分错的样本添加到“非人脸”样本中，来进行下一次的训练，这样不加约束的非人脸样本空间过于庞大，迭代运算的次数过多，这也是造成算法计算复杂度高，检测速度相对较慢的原因之一。

3 Adaboost 算法及其实现

Adaboost 算法是由 Yoav Freund 和 Robert E.Schapire 在 1997 年提出的一种基于统计学的分类器算法[29]，其基本思想是利用大量分类能力一般的弱分类器（weak classifier）通过一定方法叠加(boost)起来，构成一个分类能力很强的强分类器。理论证明，只要每个弱分类器分类能力比随机猜测要好，当弱分类器个数趋向于无穷时，强分类器的错误率将趋于零。

3.1 矩形特征与积分图

在介绍 Adaboost 算法之前，先介绍 2 个重要的概念。

3.1.1 矩形特征

使用矩形来作为人脸检测的特征向量，称为矩形特征。用特征而不是直接使用像素点来做检测主要是由于两个原因：一是特征能够针对特定区域内容进行编码，而这一点很难使用有限的训练数据通过像素点做到；二是使用特征进行检测比使用像素点进行检测的速度要快很多。

本文使用图 3.1 所示的几类简单矩形特征来描述人脸，矩形特征的物理意义如图 3.2 所示。

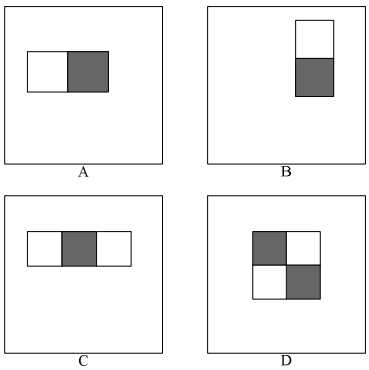


图 3.1 本文使用的矩形特征

Fig 3.1 Rectangle features in this paper

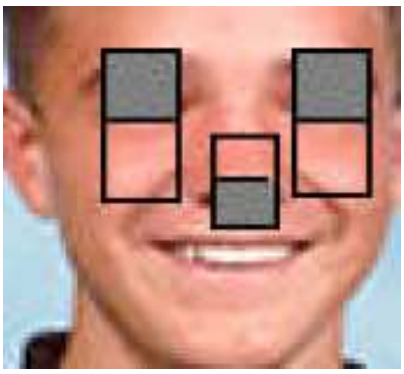


图 3.2 矩形特征的物理意义

Fig 3.2 Physical meaning of rectangular features

矩形特征值用来度量一个矩形特征，比如图 3.1 中的 A 和 B 的矩形特征值就是灰白两个矩形内像素点的灰度值之差，C 的矩形特征值为两个白色矩形像素点灰度值与中间灰色矩形像素点灰度值之差。

3.1.2 积分图

矩形特征值的计算量非常大，特别在图像较大，实时性要求较高的情况下，直

接取像素点灰度值再计算特征值明显不能满足要求。Paul Viola 等提出了一种积分图的概念[30] [31]，如图 3.3 所示。

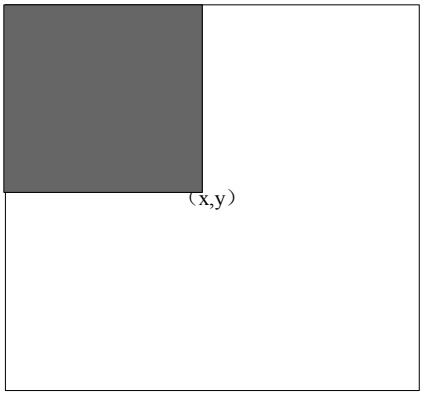


图 3.3 积分图

Fig 3.3 Integral image

图 3.3 中点（x,y）的积分图为点（x,y）左边和上面的所有像素点（即灰色部分像素点）值之和，计算公式为：

$$ii(x,y)=\sum_{x'\le x,y'\le y} i(x',y') \tag{3.1}$$

其中 $ii(x,y)$ 是积分图，$i(x',y')$ 是原图的值。使用以下公式即可对点（x,y）的积分图进行迭代求解：

$$\begin{aligned} ii(x,y)&=ii(x-1,y)+s(x,y)\\ s(x,y)&=s(x,y-1)+i(x,y)\end{aligned} \tag{3.2}$$

其中，$s(x,y)$ 是第 x 列上从 0 到第 y 个像素点值的和，并且 $s(x,-1)=0, ii(-1,y)=0$ 。

使用积分图可以计算任意矩形内的像素点值之和，如计算图 3.4 中矩形 D 内像素点之和。

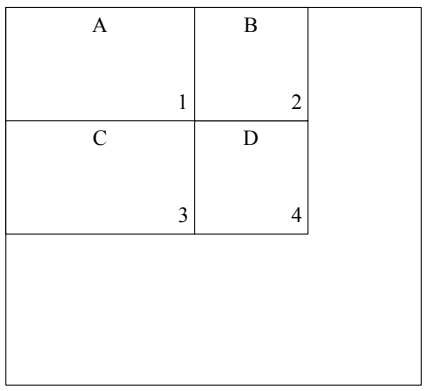


图 3.4 积分图计算示意图

Fig 3.4 Computing the integral image

如图所示，点 D1 的积分图为矩形 A 内像素点值之和，点 D2 的积分图为矩形 A,B 的像素点值之和，点 D3 的积分图为矩形 A,C 的像素点值之和，点 D4 点积分图为矩形 A,B,C,D 内的像素点之和。那么矩形 D 内像素点值的和的计算公式为：

$$S(D) = ii(D4) + ii(D1) - ii(D2) - ii(D3) \tag{3.3}$$

这样，3.1 节采用的矩形特征的特征值就只与端点的积分图相关，而不必重复计算每个像素点的灰度值，而且使用积分图的计算也只是简单的加减运算，这样就大大减小了计算复杂度，特征值的计算也只是常量级的。如图 3.5 所示的矩形特征，就可以简单的用 $F = S(A) - S(B)$ 来计算。

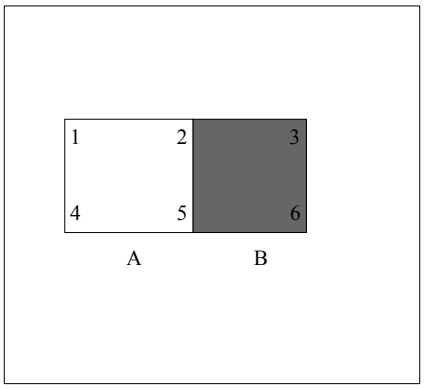


图 3.5 矩形特征计算示意图

Fig 3.5 Computing the rectangle feature

3.2 训练过程

Adaboost 算法是一个分级分类器算法，其分类器按规模和功能来划分，从小到大可以分为三类：弱分类器，强分类器和级联分类器。下面分别就这三类分类器的训练过程展开描述。

3.2.1 训练弱分类器

每一个弱分类器对应着一个特征，训练最优弱分类器和选择最好的特征即是同一个过程。在 Adaboost 算法中，那些被组合起来的弱分类器之所以被称为“弱”，是因为单个分类器并不被期望能对训练数据作出准确的分类。即便是一个最优的弱分类器，它对训练数据的分类准确率也可能只有 51%。对于一个弱分类器，只要它的分类准确率比随机分类的结果（50%）高就行，这样在每一轮训练过后，根据本轮分类器对所有样本的分类结果修改每个样本的权重，使分类错误的样本权重增加，并投入到下一轮的训练过程中去。

一个弱分类器由下式表示：

$$h_j(x) = \begin{cases} 1 & P_j f_j(x) \le P_j \theta_j \\ 0 & \text{否则} \end{cases} \tag{3.4}$$

其中 x 表示图片，f_j 表示当前特征，$h_j(x)$ 表示弱分类器，θ_j 表示阈值，P_j 指明不等式的方向。

从式中可以看出，一个弱分类器是由特征 f_j 和阈值 θ_j 决定的，训练一个弱分类器，就是根据当前的权重分布，找到一个最优的阈值，使得这个弱分类器对所有训练样本的分类误差总和最低。在 Adaboost 算法中，一个弱分类器仅仅对应一个特征，那么每一轮弱分类器的训练过程其实就是选择一个能最好的将人脸样本和非人脸样本分离开来的矩形特征。训练时对于每一个特征都要确定一个最优阈值，使得该阈值对所有样本的分类效果最好。这样，每一轮训练过程都可以得到一个在当前权重分布情况下分类效果最好的特征，而该特征所对应的弱分类器就是该轮所选出的最优弱分类器。也就是说，选取一个最优弱分类器就是选择那个对所有训练样本的分类误差在所有弱分类器中最低的那个弱分类器。

最优弱分类器的训练过程如图 3.6 所示。

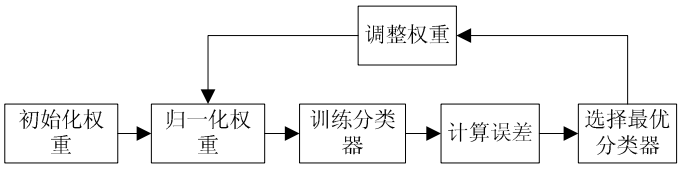


图 3.6 最优弱分类器训练过程示意图

Fig 3.6 Procedure of training the best weak classifier

具体描述如下：

①给定一系列样本（x_1,y_1），(x_2,y_2)….(x_n,y_n)，其中 x_i 是图片数据，y_i 对应人脸样本和非人脸样本取值 1 和 0；

②初始化权重 w，对于人脸样本 $w=\frac{1}{2l}$，非人脸样本 $w=\frac{1}{2m}$，其中 l 为人脸样本数，m 为非人脸样本数；

③循环，对于给定的分类器的数目 T，for t = 1,2…T

- 归一化权重：$w_{t,i}=\frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$； (3.5)
- 对于每一个特征 j 训练一个弱分类器 h_j，然后计算其误差：
- $\varepsilon_j=\sum_i w_i|h_j(x_i)-y_i|$； (3.6)
- 选择误差最小的那个弱分类器作为最优弱分类器；
- 更新样本权重：$w_{t+1,i}=w_{t,i}\beta_t^{1-e_i}$，其中当样本 x_i 正确分类时，e_i 取

值为 0，否则取值 1，$\beta_t=\frac{\varepsilon_t}{1-\varepsilon_t}$。 (3.7)

3.2.2 训练强分类器

Adaboost 算法可以理解为对特征的贪心选择过程，一个特征对应了一个弱分类器。Adaboost 分类器包含了许多个弱分类器，按照一定的方法将这些弱分类器串联起来就形成了一个强分类器。

经过 T 次训练后，得到了 T 个最优弱分类器，将这 T 个弱分类器按下面的公式组成一个强分类器：

$$h(x)=\begin{cases}1 & \sum_{t=1}^{T}\alpha_t h_t(x)\geq\frac{1}{2}\sum_{t=1}^{T}\alpha_t \\ 0 & \text{否则}\end{cases} \tag{3.8}$$

其中 $\alpha_t=\log\frac{1}{\beta_t}$。

当使用这个强分类器进行检测时，相当于让组成该强分类器的所有弱分类器投票，再对投票结果按照弱分类器的错误率加权求和，最后将加权求和的结果与随机投票结果进行比较，以得出最终的结果。

3.2.3 训练级联分类器

将多个弱分类器组合而成的强分类器虽然拥有较好的检测率，但是在检测速度和误检率上不一定能达到要求，因此考虑将多个强分类器级联起来形成一个级联分类器，以达到更高的检测速度和更低的误检率。

级联分类器如图 3.7 所示。

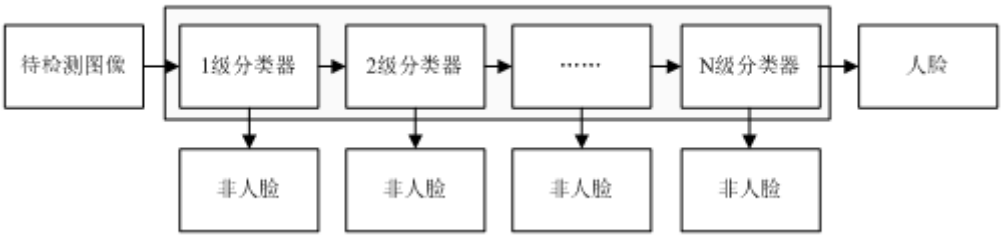


图 3.7 级联分类器示意图

Fig 3.7 The cascade classifiers

图中每一级分类器都是一个强分类器，组成每一个强分类的弱分类器的数目都随着级数而增加，另外每个强分类器都能检测出大多数人脸样本，而排除绝大部分的非人脸样本。这样由于前面的强分类器使用的特征数目少（弱分类器少），计算非常快，越向后走，通过分类器的待检测图像越少，尽量后面弱分类器的层数

越来越多，计算量越来越大，但是在实际应用中，大多数非人脸区域都在前面被排除掉了，所以能引发后续计算的图像很少。

级联分类器误检率 F 由下面的公式表示：

$$F = \prod_{i=1}^{K} f_i \tag{3.9}$$

其中 K 是强分类器的数目， f_i 是第 i 层强分类器的误检率。

检测率 D 由下面的公式表示：

$$D = \prod_{i=1}^{K} d_i \tag{3.10}$$

其中 K 是强分类器的数目， d_i 是第 i 层强分类器的检测率。

从上面两个公式我们可以看出，要训练出一个具有检测率 D 和误检率 F 的级联分类器，就要训练出 K 个检测率为 d_i 和误检率为 f_i 的强分类器。比如希望级联分类器的检测率达到 90%，则可以训练 10 个具有检测率 99%的强分类器（$0.99^{10} \approx 0.9$）。同样如果希望误识率为 0.0006%，只要每层的强分类器的误识率不高于 30%就可以了（$0.3^{10} \approx 6*10^{-6}$）。

要提高强分类器的检测率，一个简单的办法就是降低其阈值，但是这样又会增加其误检率，所以提高检测率和降低误检率是一个矛盾的组合；另外实验证明，增加弱分类器的数目能有效地提高检测率和降低误检率，但是这样又会加大计算量，提高计算复杂度。因此，在构造一个级联分类器时，需要在降低分类器阈值和增加弱分类器数目这二者中寻找一个平衡点。

级联分类器的训练过程描述如下：

①选定每级分类器的最大误检率 f 和最小检测率 d；

②选定整个级联分类器的误检率 F_{target}；

③设人脸样本数为 P，非人脸样本数为 N；

④初始化级联分类器的的误检率 $F_0=1$；检测率 $D_0=1$；计数器 i=1；

⑤循环，直至 $F_i \le F_{target}$

- i = i + 1;
- $n_i = 0$; $F_i=F_{i-1}$;
- 循环，直至 $F_i \le f * F_{i-1}$
  - $n_i = n_i + 1$;
  - 利用 Adaboost 算法训练出一个具有 n_i 个特征的强分类器；
  - 计算当前级联分类器的误检率 F_i 和检测率 D_i；
  - 降低第 i 层强分类器的阈值直到当前级联分类器的检测率至少达

到 $d * D_{i-1}$；

- $N = \phi$;
- 如果 $F_i > f * F_{i-1}$，用当前级联分类器检测非人脸图像，并将误检的图像放入 N 中。

3.3 检测过程

人脸检测的过程就是将待检测区域的图像依次通过级联分类器的过程，而级联分类器又是由若干个强分类器组成，强分类器又是由若干个弱分类器组成，所以本节只介绍待检测区域经过弱分类器检测的过程。

待检测区域图像初始化为与人脸训练样本一样大小，设定为检测图像左上角，开始检测时首先对检测图像进行积分图计算，这样就可以利用积分图的特性对待检测区域进行快速的特征值计算。特征值计算完毕后跟当前层阈值进行比较，如果大于该阈值则进入下一层弱分类器，循环直至最后一层，通过所有弱分类器的区域即认为是人脸区域，否则便判定当前待检测区域图像为非人脸图像，将其排除。具体流程如图 3.8 所示。

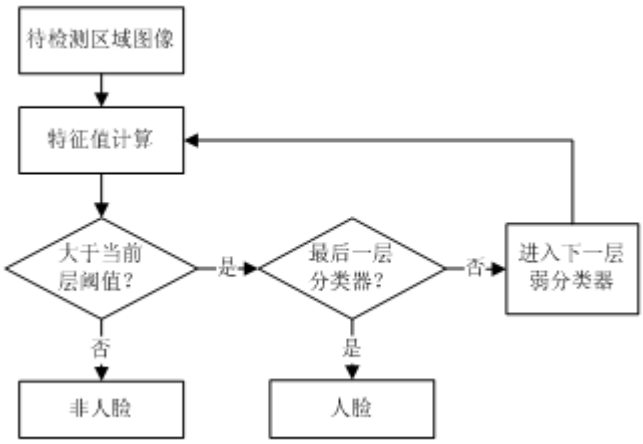


图 3.8 人脸检测过程示意图

Fig 3.8 Procedure of face detection

用初始化大小的待检测区域遍历完整个图像后，将该区域大小按比例放大，以检测不同大小的图像区域。放大比例既不能过大，因为这样会漏检掉大小处于相邻两个窗口之间的人脸；同时也不能太小，因为如果相邻窗口的大小间隔太小，检测过程中会多出很多多余的计算，影响检测速度。

但是，采用这一方法会带来一个问题：同一人脸可能会在不同的尺度和邻近的位置上被检测到很多次，所以最后还需要一个合并过程将所检测出来的矩形区域

进行合并，以便得到一个唯一的人脸区域和大小。

3.4 算法实现

根据上文描述的算法，本系统先在 VC 环境中编写程序实现该算法，流程图如下：

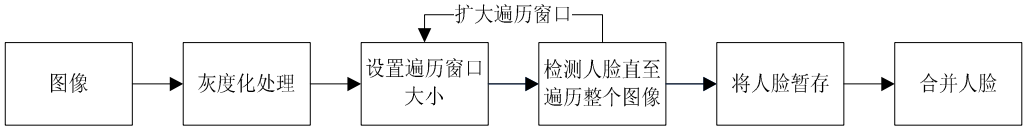


图 3.9 人脸检测系统流程图

Fig 3.9 working flow of face detection system

图像采集后，将其转化为灰度图像作为待检测图像。考虑到摄像头的最佳工作距离为 1 到 5 米，设置初始化检测窗口大小为 40×40，用初始大小的窗口遍历完整张图像后，按 1.1 的倍率逐步增大检测窗口的大小，直至与待检测图像大小大致相当。另外人脸区域的合并策略采用中心距离法，即临近人脸窗口的中心距离小于某一阈值时，我们判定这些窗口包含的是同一张人脸。具体检测窗口的检测流程可以参见图 3.8。

4 系统硬件设计

系统的设计目标是：在一定范围内（0 到 5 米），系统能检测到所有存在的人脸，并将其显示出来。根据这一目标，将系统硬件分为图像采集设备、图像处理设备、图像显示设备和存储器等 4 大主要部分，如图 4.1 所示。

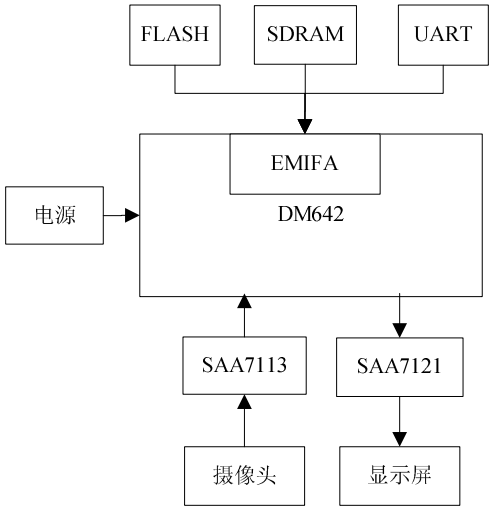


图 4.1 人脸检测系统硬件框图

Fig 4.1 Hardware components of the face detection system

4.1 摄像头

对于图像处理系统来说，摄像头的选择是很重要的，应该根据实际情况加以挑选，既要性能良好，也要考虑到经济实用。摄像头的模拟制式分为 PAL 制式和 NTSC 制式。摄像头的分辨率是很重要的参数，分辨率越高，图像越清晰，细节表现的越好，有利于以后对图像的分析处理。此外，摄像头的最低照度和信噪比也是影响图像效果的重要参数。

本系统采用翔飞公司生产的 2052X 型 CCD 摄像头采集输入图像。CCD 摄像头是一种新型摄像器件，具有体积小、重量轻、功耗小、分辨率高等特点，并且在动态范围、灵敏度和自扫描方面具有其它摄像器件无法比拟的优势。基于半导体器件的 CCD 摄像头，感光表面上分布着二维阵列的光电传感器，一个象素点对应一个传感器，当光线被聚焦在感光表面时，每一个传感器都把光信号转换成电信号，并在帧间隔时间内存储在缓存中。在读出这些信号时，按照从上到下的顺序，逐行进行读出显示或做数据处理。

2052X 摄像头的主要性能参数如下：

- CCD 类型：SONY 1/3" CCD
- 像素：PAL：752H×582V；NTSC：768H×494V
- 信号制式：PAL/NTSC
- 分辨率(水平中心)：500TVL
- 最低照度: 0.5LUX/F1.2
- 信噪比: ≥48dB
- 工作电压: DC12V±5%
- 工作温度: -20℃--50℃

4.2 图像解码芯片

本系统采用 Philips 公司的 SAA7113 解码芯片[32]，将视频输入口输入的标准 PAL 制式信号转化成 YUV 4:2:2 格式的 8 位数字信号，提供给 DM642 芯片做处理。

SAA7113 数字视频解码器用来把复合视频输入信号 CVBS 或分离视频信号 S-VIDEO 解码成标准 ITU 656 YUV 4：2：2 格式的 8 位数据，它支持 PAL、NTSC 制式，支持隔行扫描和多种数据输出格式，可自动监测 50Hz、60Hz 的场频信号，并可在 PAL 制式和 NTSC 制式下自动转换。芯片内部有亮度、色调、色饱和度控制电路，并提供有 I2C 总线接口，可通过其 I2C 接口对芯片内部电路进行控制。该芯片具有如下特点：

- 支持四路模拟输入，内置信号源选择器；
- 有两个模拟预处理通道；
- 内置两个模拟抗混叠滤波器；
- 两个片内 9 位视频 A/D 转换器；
- 行/场同步信号自动检测；
- 多种数据输出格式。

对 SAA7113 的控制主要包括对输入模拟信号的预处理、色度和亮度的控制，输出数据格式及输出图像同步信号的选择控制等。在 SAA7113 所提供的多种数据输出格式中，ITU656 格式在 8 位输出管脚上直接输出与像素时钟相对应的像素灰度值，此种数据格式与其它格式相比对灰度图像的采集将更直接。

4.3 图像编码芯片

系统显示部分将 DM642 处理过后的数字信号经过 Philips 公司的 SAA7121 编码芯片[33]进行 D/A 转换，生成模拟信号，通过 VP0 端口传输给电视显示。

SAA7121 支持 PAL 与 NTSC 格式的视频编码, 将数字视频信号编码成普通电视所能接收的 PAL 或 NTSC 制式的复合电视信号。SAA7121 支持 NTSC-M、PAL

B/G 和子标准，并具有 Y、C 和 CVBS 三个信号的数模转换器。SAA7121 提供有 I2C 总线接口，可按主动方式或从动方式工作。单一的 3.3V 供电,可通过其 I2C 接口对芯片内部电路进行控制。该芯片具有如下特点：

- 三个片内 10 位视频 D/A 转换器分别对应 Y,C 和 CVBS, 两倍过采样；
- 实时载波控制；
- 支持 Master 和 Slave 模式；
- 多种数据输出格式。

4.4 DM642 处理芯片

TMS320DM642 是 TI 公司推出的一款针对视频和图像处理领域应用的定点多媒体处理芯片[34]，其最高时钟频率为 720MHz，最大处理能力为 5760MIPS。它在 C64x 的基础上增加了很多外围设备和接口，是实现实时视频系统的较好平台。它的主要外围设备有：

- 三个可配置的视频接口，同时支持三路视频输入或者输出；
- 一个 10/100M 的以太网接口；
- 64-bit 的外部存储单元接口，支持连接同步或异步存储单元；
- 16 个通用输入输出口 GPIO，以及 I2C、PCI 和 HPI 等众多协议接口。

4.4.1 DM642 的 CPU 结构

DM642 是 TMS320C6000DSP 平台中的高性能定点处理 DSP，使用先进的超长指令架构，在一个指令周期能够并行处理多条指令。为了满足视频和图象处理的需要，DM642 采用 VelociTI 体系结构。DM642 专门针对数字多媒体处理，其指令集跟 C6000 系列的其他平台兼容。

DM642 的 CPU 结构主要包括以下几个部分[35]。

①程序读入、指令译码及分配：该部分包括程序取指单元、指令译码单元和指令分配单元；

②程序执行：该部分包括 2 个对称数据通路，2 个对称的通用寄存器组和 8 个对称的功能单元，以及控制寄存器组和控制逻辑、中断逻辑等；

③芯片测试和仿真端口及其控制逻辑： DM642 的 CPU 采用哈佛总线结构，其程序总线与数据总线相互独立，片内的程序存储器与数据存储器相互独立，取指令与执行指令可以并行运行。程序总线连接片内程序存储器与 CPU，片内程序存储器用来保存指令代码。程序总线宽度为 256 位，每一次取指操作取 8 条指令。执行时，每条指令占用 1 个功能单元，取指、指令分配和指令译码单元在每个周期内都可以读取并传递 8 条 32 位指令。

DM642 的 CPU 起主要作用的是两套功能单元，每一套包括 4 个功能单元和 1

个寄存器组，4 个功能单元分别是.L1、.S1、.M1、.D1 功能单元组和.L2、.S2、.M2、.D2 功能单元组。两个寄存器组每个包含 32 个 32 位寄存器共 64 个通用寄存器。支持在 C62xx VelociTI.2 VLIW 体系中的便携 16 位和 32/40 位定点数据类型，同样，DM642 寄存器组还支持大量的 8 位数据和 64 位定点数据类型。两套功能单元，连同两侧的寄存器组，组成 CPU 的 A，B 部分。每部分的四个功能单元可以自由分享自己部分的 32 位寄存器，此外，两部分之间有一条单独的数据总线连接另一部分的所有寄存器，构成数据交叉通路，这样两套功能单元可以从对方的寄存器组交叉访问数据。DM642 CPU 数据交叉通路访问需要多个时钟周期，这使得同一个寄存器在同一个执行包作数据交叉通路时可用作多重功能单元。DM642 CPU 的所有功能单元都可以通过数据交叉路径访问操作数。在一个时钟周期内，在同一部分的寄存器可以被所有的功能单元访问。在 DM642 CPU 中，当指令试图通过数据交叉路径读取在先前一个时钟周期更新过的寄存器时，就会出现延迟时钟。

4.4.2 流水线结构

流水线结构使得多条指令在同一时间、不同部件内得到处理。C6000 系列芯片中所有指令均按照取指、译码和执行三级流水线运行，每一级流水线又占有不同的 CPU 周期：取指操作需要 4 个周期，译码操作需要 2 个，而执行操作根据不同的程序指令有不同的 CPU 周期。

流水线取指操作的 4 个周期为程序地址产生，程序地址发送，程序访问等待，程序取指包接收。译码操作的 2 个周期为指令分配和指令译码。在指令分配周期中，一个取指包中的 8 条指令根据指令的并行性被分成若干个执行包，1 个执行包的指令被分别分配到相应的功能单元。同时，源寄存器、目的寄存器和有关通路被译码，以便在功能单元完成指令执行。执行操作根据定点和浮点操作分别占用不同的 CPU 周期，定点流水线的执行操作最多有 5 个周期，而浮点流水线最多有 10 个。执行不同类型的指令需要不同数目的周期。

4.4.3 DM642 存储结构

DM642 的片内内存采用 2 级高速缓存（cache）结构，程序和数据拥有各自独立的高速缓存。片内的第一级程序缓存称为 L1P，大小为 16k bytes，第一级数据缓存称为 L1D，大小也为 16k bytes。CPU 与 L1P 及 L1D 直接相连，因此，这两块 Cache 都工作在 CPU 全速访问状态。程序和数据共享的第二级存储器为 L2，大小为 256k bytes，二级缓存的分段和大小分配可以由用户决定，既可以将二级缓存全部用作缓存，也可以根据实际情况将一部分二级缓存用来存储程序代码或数据。

当 CPU 发出访问请求后，先将存储器地址发送到 cache 控制器以确定所需数据是否已在 cache 中。如果命中，则直接对 cache 进行访问。比如 CPU 的取指如果命中 L1P，则 CPU 将在单个周期内得到所需的取指包，如果没有命中 L1P，而

是命中的 L2，CPU 将被阻塞 0-7 个周期，具体阻塞的周期数取决于当前执行包的并行度，以及流水线工作状态。如果也没有命中 L2，则 CPU 被阻塞，请求转给 EDMA。EDMA 一旦读到所需的数据，就会直接将数据由 L2 送入 L1，再提交给 CPU。L1D 和 L1P 还提供了一种默认的流水处理机制，能够减少一级缓存未命中时的阻塞周期。

4.4.4 DMA 与 EDMA

DMA（Direct Memory Access）即直接存储器访问，是 DSP 中一种重要的数据访问方式，它可以在没有 CPU 参与的情况下由 DMA 控制器完成存储空间内的数据传输。数据传输的源/目的地址可以是片内程序或数据存储器、片内集成外设、外部存储器和扩展总线上的扩展存储器。

EDMA（Enhanced Direct Memory Access）是在 DMA 基础上发展起来的增强直接存储器访问，DM642 提供了 64 个独立的 EDMA 通道，可以在没有 CPU 参与的条件下，实现片内存储器、片内外设在及外部存储空间之间的数据高速传输，通道的优先级可以通过编程进行控制。EDMA 控制器主要由事件和中断处理寄存器、事件编码器、参数 RAM、硬件地址产生器组成，负责 L2 与其它外设之间的数据传输。

4.4.5 DM642 视频端口

DM642 视频端口外设能够运行在视频采集、视频输出或者传输码流接口(TSI)采集等方式下。下面分别介绍各种工作模式的主要特点。

①视频采集模式

- 80MHz 的采集速率；
- 两路 8/10 位的数字视频输入，数字视频为 YUV422 格式，有 8bit 或者 10bit 两种精度，支持 ITU-R BT.656 标准；
- 一路 16/20 位的数字视频输入，为 YUV422 格式，支持 SMPTE260M、SMPTE274M、SMPTE296M、ITU-BT1120 等标准；
- 支持 YUV422 到 YUV420 格式的转换以及 8 位 YUV422 模式下的亚采样；
- 通过 A/D 转换器可以直接获取两路 10 位或一路 20 位的原始视频。

②视频输出模式

- 显示速率能达到 110MHz；
- 一路连续视频输出，数字视频输出为 YUV422 格式，8/10 位精度；
- 一路连续 16/20 位 YUV422 格式的数字视频输出；
- 支持 YUV420 到 YUV422 格式的转换，在 8 位 YUV422 模式下，输出 2 倍插值；

- 能产生行同步，场同步信号和消隐信号

③传输流接口(TSI) 采集模式

- 传输流接口能以 8 位并行，最大 30M/秒的速率接收数据；
- 可以采集 MPEG-2 码流数据。该传输流将音频，视频，数据程序流全部复合成一个传输流；
- 支持同步检测；
- 能并行数据接收；
- 使用硬件计数器机制为数据添加时间戳；
- 具有纠错机制。

4.4.6 DM642 视频 FIFO

DM642 在视频口和内存之间设置了一个视频输入输出缓存区，用来存储输入输出的视频数据，该缓存区使用 EDMA 进行数据访问。

视频 FIFO 能根据不同的视频采集和输出模式进行配置，例如在 BT.656 格式下将视频 FIFO 分成 AB 两个通道，每个通道又分成 Y、Cb、Cr 三个分量缓存区，享有不同的写指针和读寄存器。同样在视频输出时，每个通道也分成 Y、Cb、Cr 三个缓存区，享有不同的读指针和写寄存器。

4.5 存储芯片

由于 DM642 芯片内部只集成了 256K 的 SRAM，因此需要在外部存储器接口上扩展存储空间。SDRAM 主要是用来存储 DSP 系统运行过程中的代码、数据、堆栈等数据。

TMS320DM642 访问片外存储器时必须通过 EMIF。DM642 使用的是 64 位 EMIFA，不仅具有很高的数据吞吐率，而且可以与目前几乎所有类型的存储器直接连接。EMIF 有四个片使能，能够支持 64bit, 32bit, 16bit 和 8bit 的外部器件，可寻址空间高达 1G（80000000–BFFFFFFF），本系统采用了两片 Micron 公司的 MT48LC4M32 芯片作为 SDRAM，存储空间为 0x80000000 至 0x81FFFFFF（32M，位于 CE0 空间）。

系统采用的 Flash 芯片包括一片 MX29LV320B(NOR Flash4M×8bits)和一片 K9F2808UOC(NAND Flash16Mbytes)。MX29LU320b 它是 4Mx8 或者是 2Mx16 的 Flash memory。系统是把 Flash 映射到 EMIF 的 CE1 子空间的前 O.5MB 的空间。由于 4MB 的 Flash ROM 有 22 根地址线，而 DM642 只有 20 根地址线，1MB 的空间。Flash 容量大于 DSP 所提供的空间，对此加入了 CPLD。DM642 只与 Flash 有 19 根地址线相连，剩下的 3 根地址线（A[3.2.1]）连接到 CPLD 上，由 CPLD 来控制，相当于对地址空间进行了分页。

K9F2808U0C 是 NAND 结构 Flash 存储器件，它是一款性价比很高的大容量数据存储器。Flash 存储器的应用范围极广，从现代计算机优盘到嵌入式系统中都有应用。大致说来，Flash 操作包括检错（对 Flash 内部坏扇区的检测）、写操作（写入数据）、读操作（从 Flash 中读出数据）、空间管理和擦除操作。

系统上电后，DSP 自动将存储在 Flash 中的代码复制到 SDRAM 中运行，以实现系统自启动。

5 DSP 软件设计开发

本系统基于 TI 公司的 CCS 进行开发，软件的开发流程要与其环境配置相一致。图 5.1 描述了 TMS320C6000 系列的 DSP 程序开发流程[36]。其中阴影部分为大多数程序的通用开发流程，其他部分则是为了加强 C6000 系列软件开发流程而加入的部件。

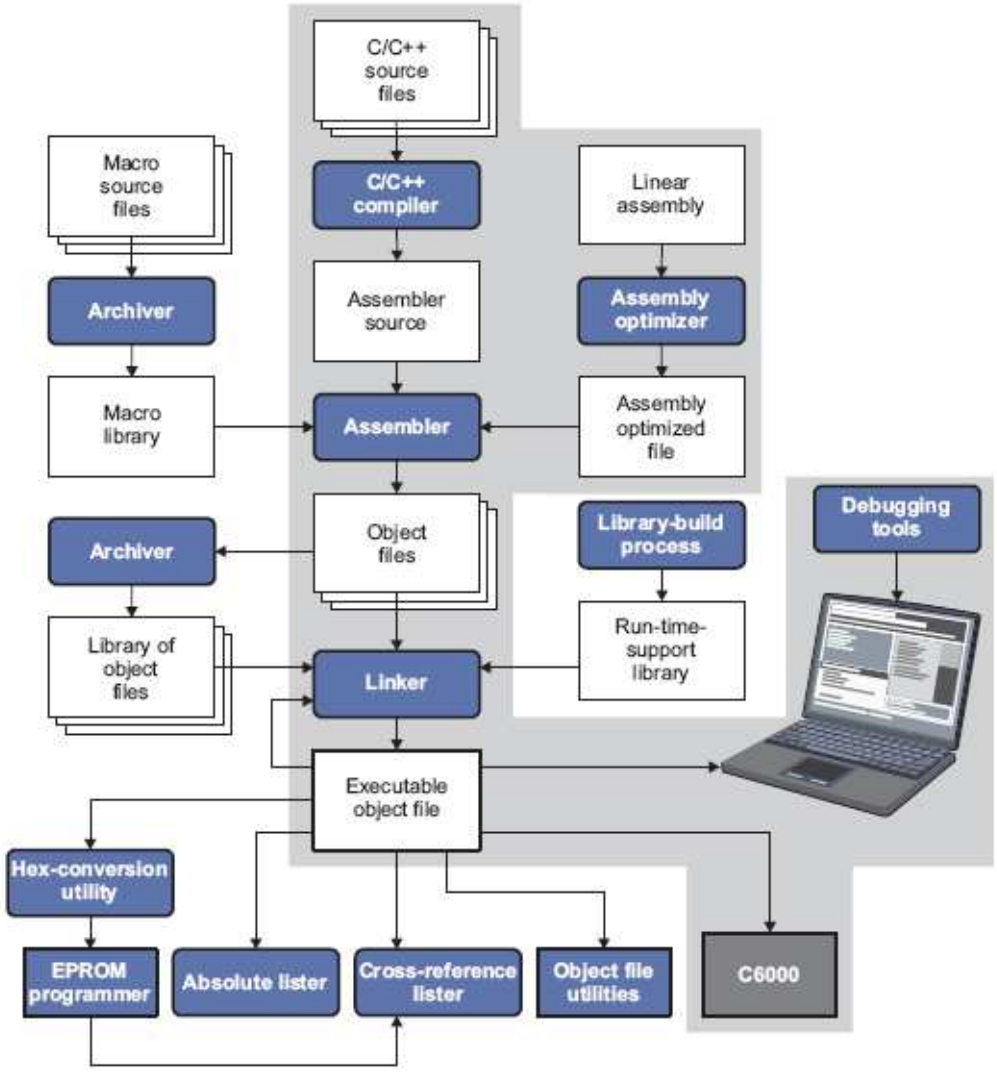


图 5.1 TMS320C6000 系列 DSP 软件开发流程示意图

Fig 5.1 Software developing flow of TMS320C6000 Chips

图中的开发工具与 GCC 开发工具基本相同，其中重要的开发工具如下：

- C 编译器(C compiler)：将输入的 C 语言代码编译生成汇编语言代码，由：编译器、优化器(optimizer)和交叠工具(interlist feature)三个工具组成：

- 编译器使用户能进一步完成编译、汇编和链接；
- 优化器对代码进行修改优化，提高程序的执行效率；
- 交叠工具把 C 语句和对应的汇编语句交叠列出。

- 汇编器(assembler)：将汇编语言文件转换成机器语言的目标文件，机器语言格式为通用目标文件格式(Common Object File Format ,COFF)。
- 链接器(1inker)：将多个目标文件以及外部链接库组合成单个可执行目标文件。
- 归档器(archiver)：将一组目标文件打包并形成一个库文件。归档器也可以通过添加、删除、替换文件来调整库。
- 建库程序(1ibrary-build process)：用来建立满足开发者自己要求的“运行时支持库”。标准的运行支持库函数以源代码的形式放在 rts．src 文件中。
- 运行时支持库(run time support libraries)：包括 C 编译器所支持的 ANSI 标准运行支持函数、编译器公用程序函数、浮点运算函数和 I/O 函数等，它的目标代码是以 little-endian 模式生成的。

5.1 使用 CCS

CCS（Code Composer Studio）是由 TI 公司发布的基于 Windows 平台的 DSP 开发环境，是目前最流行的 DSP 开发软件之一，其构成模块和接口如图 5.2 所示。CCS 除扩展了基本的代码生成工具以外，还集成了 C 编译器、汇编器和链接器等，并支持 RTDX（Real Time Data Exchange）技术，可在不中断目标系统运行的情况下，实现 DSP 与其他应用程序的数据交换。此外，CCS 还包括了调试工具，如断点、探针工具和分析工具等，这使得开发者能够在 CCS 这个集成环境下完成所有的软件开发流程。

本系统使用 CCS3.1 进行开发，使用前需要针对特定开发板（DM642）和仿真器（SEEDXDS510PLUS）进行配置，配置界面如图 5.3 所示，CCS 开发界面如图 5.4 所示。

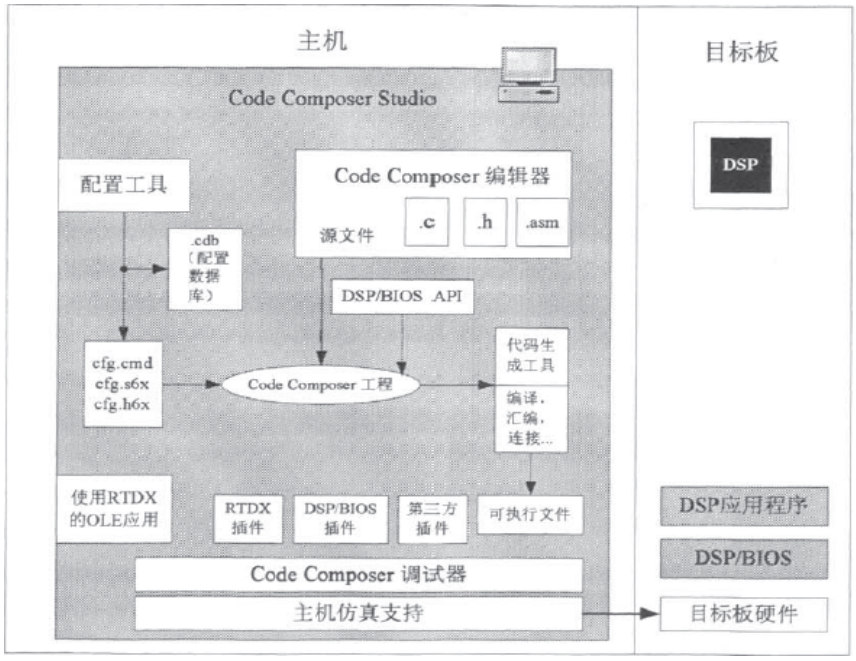


图 5.2 CCS 构成模块与接口示意图

Fig 5.2 Components and interfaces of CCS

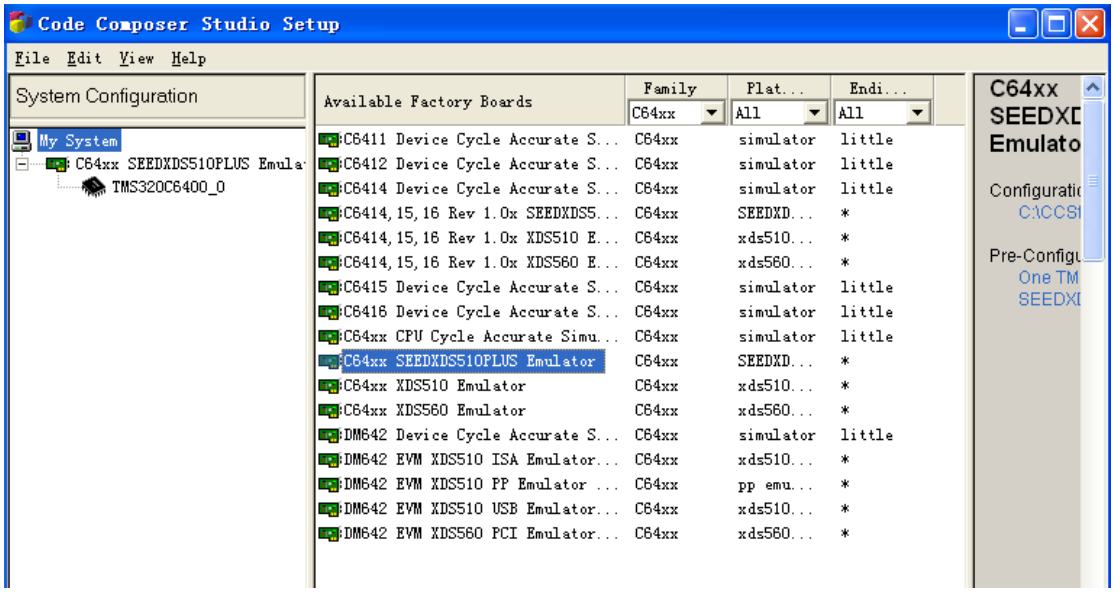


图 5.3 CCS 配置界面示意图

Fig 5.3 Configuration interface of CCS

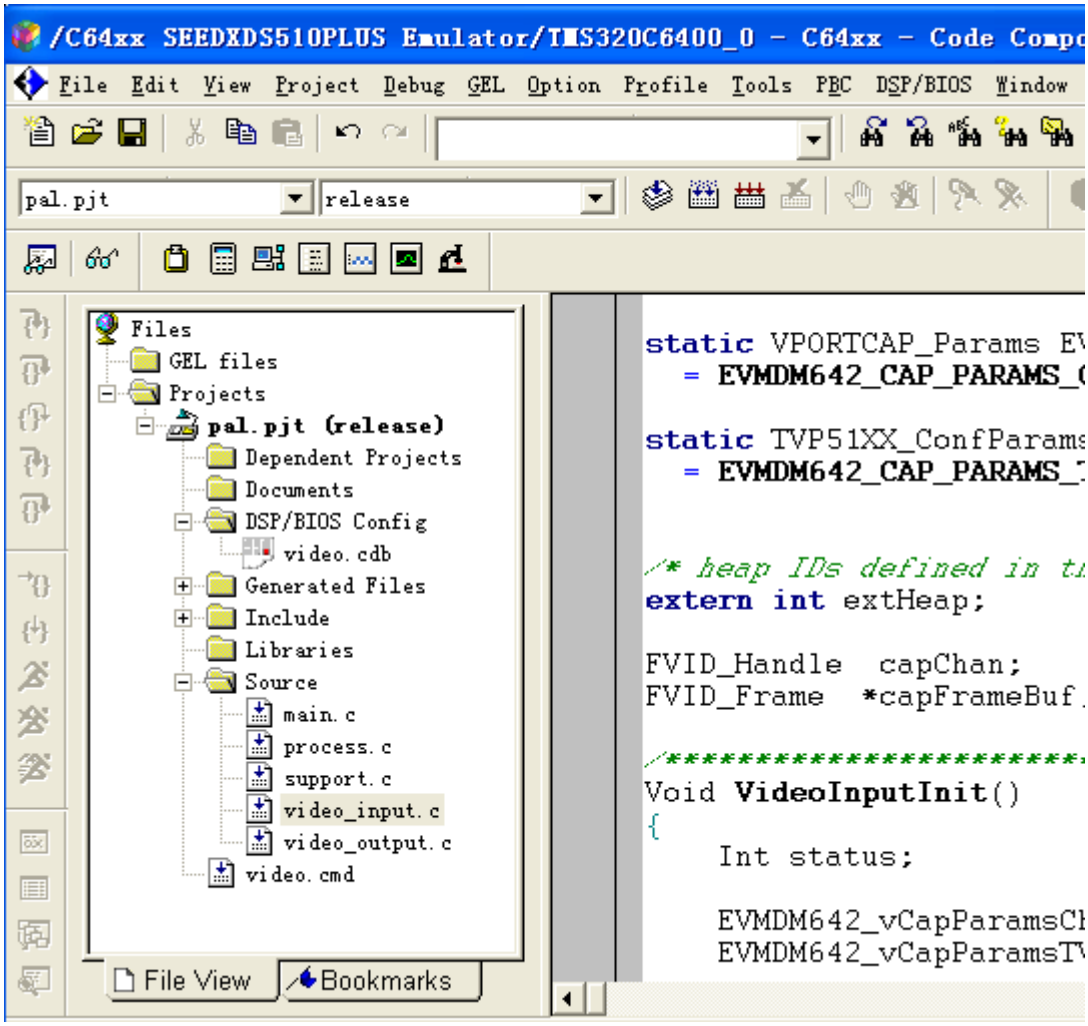


图 5.4 CCS 开发界面示意图

Fig 5.4 Developing interface of CCS

CCS 还集成了 DSP 的实时多任务系统 DSP/BIOS，它具有实时时钟管理、任务的调度管理、任务间的同步与通信管理、中断服务管理、内存管理和外设驱动管理等功能。使用 DSP/BIOS 工具可以方便地控制 DSP 的硬件资源，灵活地协调各个软件模块的工作，加快软件开发和调试速度。

5.2 使用 DSP/BIOS

5.2.1 实时多任务操作系统

随着嵌入式微处理器技术的不断发展，一个微处理器能同时处理的任务也由一个变为多个，因此程序也由单任务的结构变成多任务结构。多任务操作系统也逐渐变为嵌入式系统的应用主流。

实时多任务操作系统（Run Time Operating System, RTOS）是开发实时嵌入式软件的基础和平台，它是一段嵌入在目标代码中的可执行程序，其他应用程序都

是工作在它的基础之上。RTOS 最关键的部分是实时多任务内核，它的基本功能包括定时器管理、任务管理、存储器管理、消息管理、资源管理、系统管理等，这些管理功能以内核函数的形式交给用户调用，也就是 RTOS 的标准 API 函数。

RTOS 的引入解决了嵌入式软件开发标准化的难题。操作系统根据用户指定的各个任务的优先级，合理地在不同任务之间分配 CPU 资源，从而满足数据实时处理的需要，而且能够较好的利用 CPU 资源。DSP 作为一种高速嵌入式计算器件，对实时数据处理往往具有更高的要求。另外在许多嵌入式系统中，DSP 不仅要完成实时数据处理任务，还需要对整个系统的运行进行控制，而这个功能只有通过操作系统才能较好地实现。

使用 DSP 实时操作系统，可以使开发人员从繁重的硬件相关的编程中脱离出来，更加专注程序控制流程和算法的研究实现。而且集成了 DSP 实时操作系统的程序，具有更好的可读性、可维护性和可移植性，这能很好的帮助开发管理大型系统以及提高开发效率。

5.2.2 DSP/BIOS 概述

DSP/BIOS 是 DSP 的实时多任务操作系统，它主要的设计目的是实现任务的实时调度和同步以及主机/目标系统的通信和实时监测 [37] [38]。DSP/BIOS 组件包括抢占式多任务内核、实时分析工具和配置工具。它还提供底层的内核函数接口，用于支持线程管理、系统实时分析、调度软中断、后台运行函数以及外部硬件中断与各种外设的管理。

DSP/BIOS 组件由以下 3 部分组成：

①DSP/BIOS 配置工具

DSP/BIOS 配置工具有一个类似 Windows Explorer 的外观，基于 DSP/BIOS 的应用程序都通过一个扩展名为.cdb 的 DSP/BIOS 的配置文件进行配置。它可以：

- 进行 DSP/BIOS 模块的参数设置；
- 作为一个可视化的编辑器建立 DSP/BIOS 对象，如系统任务等；
- 进行芯片支持库（Chip Support Library）的参数设置。

DSP/BIOS 对象都可以通过配置工具静态建立或者在程序中调用 XXX_create 函数动态建立。大多数对象在程序运行过程中是一直使用的，所以可以使用配置工具静态创建这些对象，包括任务、软中断、输入输出流、周期函数及事件日志等。利用配置工具，DSP/BIOS 的对象可以被预先创建及设置，这样不仅可以合理利用内存空间、缩短代码长度、优化内部数据结构，而且可以在程序编译前通过验证对象的属性来预先发现错误。

DSP/BIOS 的配置界面如图 5.5 所示：

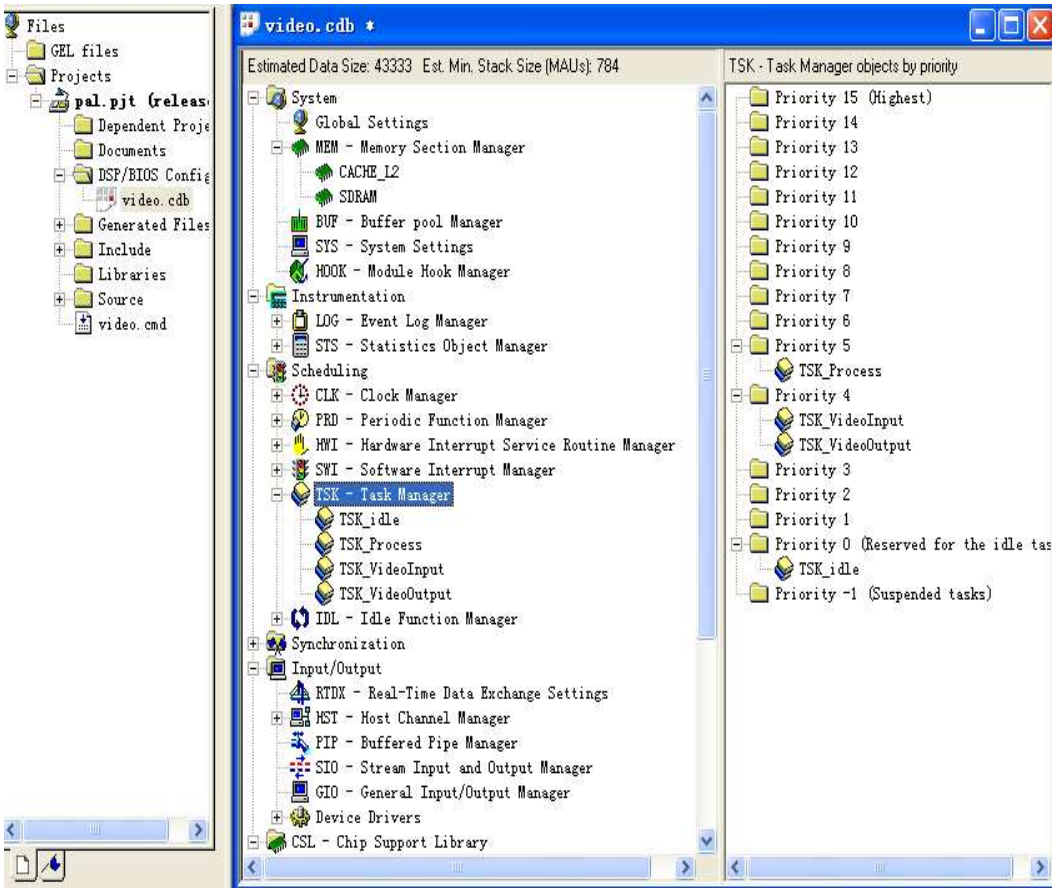


图 5.5 DSP/BIOS 配置示意图

Fig 5.5 Configuration interface of DSP/BIOS

②DSP/BIOS 实时内核与 API

DSP/BIOS 的函数接口分为多个模块，应用程序通过调用这些 API 来使用 DSP/BIOS。所有的 API 都是以 C 可调用的形式提供的，遵从 C 的调用约定，汇编代码也可以调用 DSP/BIOS 的 API。CCS 中的 DSP/BIOS 主要的 API 模块包括以下几种：

- MEM 模块：内存模块，用于定义目标系统的内存使用，系统将根据此信息自动产生.cmd 文件；
- LOG 模块：日志模块，管理 LOG 对象，LOG 对象在目标系统程序执行时实时捕捉事件，开发人员可以通过系统日志判断系统的运行状态，并利用其进行调试；
- STS 模块：统计模块，管理统计累积器，在程序运行时，它存储关键统计数据并能通过 CCS 浏览这些统计数据；
- CLK 模块：时钟管理模块，可引发定时器中断；
- PRD 模块：周期函数模块，用于实现周期性的函数，这些函数的周期性

执行通常是为了响应发送或接收数据流的外围设备的硬件中断；

- HWI 模块：硬件中断模块，提供对硬件中断服务例程的支持，可设置相应的中断服务子程序，共有 7 个优先级；
- SWI 模块：软件中断模块，管理软件中断，当目标程序通过 API 调用发送 SWI 对象时，SWI 模块安排相应函数的执行；软件中断共有 15 个的优先级，但都低于硬件中断的优先级；
- TSK 模块：任务管理模块，用于管理和调度任务，并设置任务的优先级。
- IDL 模块：空闲功能模块，管理空闲函数，在目标系统程序没有更高优先权的函数运行时，空闲函数会自动运行；
- SEM 模块：旗语管理模块，用于任务或线程之间的同步；
- MXB 模块：邮箱管理模块，用于任务间的同步或通信；
- QUE 模块：队列管理模块，用于任务或线程的队列管理；
- RTDX 模块：实时数据交换模块，允许数据在主机和目标系统之间进行实时交换；
- HST 模块：主机模块，管理主机通道对象，它允许应用程序在主机和目标系统之间进行数据交换，主机通道通过静态配置设为输入或输出；
- PIP 模块：数据通道模块管理数据通道，它被用来缓存输入和输出数据流。这些数据通道提供一致的软件数据结构，可以使用它们驱动 DSP 和其它实时外围设备之间的 I/O 通道；
- SIO 模块：流式 I/O 管理模块，可用于设备驱动模块与任务或软件中断之间的数据交换；
- DEC 模块：设备驱动程序接口；
- CSL 模块：片内定时器模块，控制片内定时器并提供高精度的 32 位实时逻辑时钟，它能够控制中断的速度，使之快则可达单指令周期时间，慢则需若干毫秒或更长时间。

③DSP/BIOS 实时分析工具

DSP/BIOS 实时分析工具可以辅助 CCS 实现程序的实时调试，以可视化的方式观察程序的性能。相比传统的本机调试方法，DSP 程序的实时分析要求在目标处理器上运行监测代码，使 DSP/BIOS 的 API 和对象可以自动监测目标处理器，实时采集信息并通过 CCS 分析工具传输给主机。实时分析包括：程序跟踪、性能监测和文件服务等。

开发人员可以通过 DSP/BIOS 提供的 Log 和 Statistics 模块在极小的系统开销代价下获取 DSP 系统的运行状态和数据。常用的实时分析工具包括 CPU 负荷图、程序模块执行状态图、信息显示窗口、主机通道控制、状态统计窗口等。

5.2.3 DSP/BIOS 具体配置

每个基于 DSP/BIOS 的程序都有一个 DSP/BIOS 的.cdb 配置文件，在对系统进行编程之前，需要利用这个配置文件对 DSP/BIOS 的参数进行设置。本系统在开发之前，事先主要对 System 模块的 Global Settings 属性和 MEM 模块进行了设置。

①System 模块中的 Global Settings 属性中的设置，如图 5.6 所示：

- 芯片的时钟频率(DSP Speed In MHz)设置为 720MHz。DSP/BIOS 中的 DSP 运行时钟频率由用户设置，并以此来配置涉及到时间量的各个寄存器值，如定时器等；
- 片上支持库(Chip Support Library)选择 DM642；
- DSP 工作模式(DSP Endian Mode)设置为小端（little）；
- 用户初始化函数 User Init Function）设置为_dm642_init，我们在此函数中实现硬件初始化。

Global Settings properties

Property	Value
Target Board Name	c64xx
DSP Speed In MHz (CLKOUT)	720.0000
DSP Type	6400
Chip Support Library (CSL)	DM642
Chip Support Library Name	cslDM642.lib
DSP Endian Mode	little
Call User Init Function	True
User Init Function	_dm642_init
Enable Real Time Analysis	True
Program Cache Control - CSR(PCC) ...	mapped
621x/671x - Configure L2 Memory S...	False
641x - Configure L2 Memory Settings	True
Program Cache Control - CSR(PCC) ...	Cache Enabled -...
641x - Program Cache Control - CS...	Cache Enabled -...
L2 Mode - CCFG(L2MODE)	SRAM
641x L2 Mode - CCFG(L2MODE)	4-way cache (256k)
L2 MAR0-15 - bitmask used to init...	0x0000
Enable All TRC Trace Event Classes	True
CDB search path in COFF file	

图 5.6 Global Setting 示意图

Fig 5.6 Global Setting

②MEM 模块的设置属性

MEM 模块提供了对 DSP 目标系统的存储器管理。通过对此模块进行配置，配置工具会自动生成工程文件链接时需要的.cmd 文件，在该文件中确定了数据、程序、堆栈存放的地址。

本系统将 L2 全部的 256KB 空间作为 cache，而将程序所有的代码和数据保存在 SDRAM 中，SDRAM 大小设置为与硬件相符的 32MB。

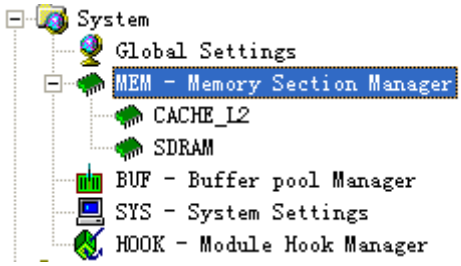


图 5.7 MEM 模块设置示意图

Fig 5.7 MEM modules setting

5.3 开发 DSP 驱动

DSP 的驱动开发可以分为基于 DSP/BIOS 的驱动开发[39]和不基于 DSP/BIOS 的驱动开发两类。不基于 DSP/BIOS 的驱动开发需要参照芯片说明书，在驱动接口函数中按照特定的时序和顺序，向寄存器写值。上层应用函数直接调用该驱动接口，其具体做法可以参照 6.2 节硬件初始化中串口芯片的初始化过程。本节主要介绍基于 DSP/BIOS 的驱动开发原理和过程。

5.3.1 基于 DSP/BIOS 的外设驱动开发模型

TI 公司在其推出的 DSP/BIOS DDK（Device Driver Kit）中，定义了标准的设备驱动模型，并提供了一系列的 API 接口。在实际开发过程中，硬件驱动程序最终是以函数库的形式封装起来，应用程序通过调用相关的 API 函数与不同外设通信，而无需关心底层硬件的具体操作。这样对于功能相似型号不同的外设，不需要重新编写整个驱动程序，只需对底层硬件驱动加以修改，就可以实现驱动程序的移植与复用，这样就大大简化了驱动程序的开发过程，提高了程序的可维护性和可移植性。

IOM 是 DSP/BIOS 的设备驱动模块的一种接口方式，可在 DSP/BIOS 的图形化配置界面中将硬件设备驱动模块为 IOM 模式。

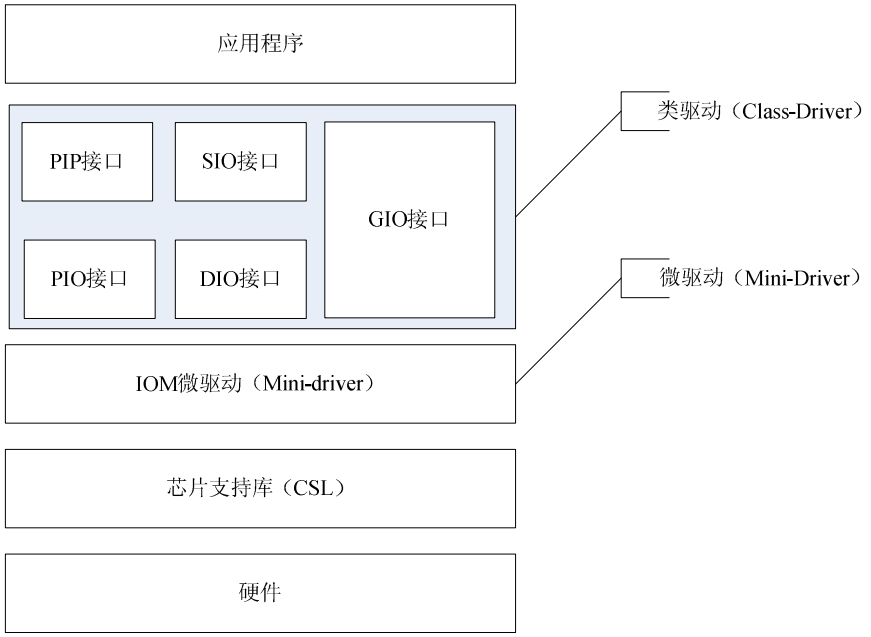


图 5.8 基于 DSP/BIOS 的驱动开发层次模型

Fig 5.8 Module of driver development based on DSP/BIOS

如图 5.8 所示，IOM 模式将设备驱动分为两层：类驱动(Class-Driver)层和微驱动(Mini-driver)层。

类驱动层程序与具体设备无关，实现的主要功能包括：向上提供应用层程序调用所需的函数接口，并协助应用程序对外设进行同步和阻塞操作；向下与微驱动层相连，将向上提供的 API 接口函数与具体的微驱动层程序建立映射关系。另外类驱动程序还提供了保存设备的数据的数据缓冲区。对于 IOM 模型的驱动开发，只要外设驱动模型选定了，类驱动程序就定下来了，基本不需要做修改。

类驱动程序以接口的形式直接在应用程序中出现，并且根据数据输入/输出的处理方式不同，分 3 类驱动程序：管道型类驱动(PIP/PIO)、流输入输出型类驱动(SIO/DIO)和通用输入输出型类驱动(GIO)。

①PIP/PIO 模型：基于数据管道的 I/O 模型，每个管道都在维护自己的一个缓冲区。其中 PIP 模块负责创建管道、数据管道输入/输出。PIO 适配模块提供负责缓冲管理、信号同步、将 API 及参数与下层微驱动程序接口。

②SIO/DIO 模型：基于数据流的 I/O 模型，对 I/O 操作进行异步处理，可以同时进行数据的读写、处理操作。其中 SIO 模块负责创建通道、数据流输入/输出。DIO 适配模块负责缓冲管理、信号同步以及将 API 及参数与下层微驱动程序建立连接。

③GIO 模型：GIO 类驱动是一种通用输入输出接口，灵活性很强，且没有适配层，调用的 API 函数，可通过阻塞线程读写数据，直接与微驱动通信，主要用来

设计新型的设备驱动模型。

PIP/PIO、SIO/DIO、GIO 模块集成在 DSP/BIOS 中，PIP 通道和 SIO 管道可在 DSP/BIOS 的输入输出模块图形化界面中静态设置并创建，也可以在应用程序中动态创建。PIO 和 DIO 适配模块(Adapter)创建在 DSP/BIOS 的设备驱动模块图形化界面中完成。

微驱动层程序集成了实际硬件相关的代码，设计微驱动程序是外设驱动开发中的重点。IOM 接口将微驱动程序与类驱动程序联系一起，包括定义微驱动程序读写的数据包(IOM_Packet)；定义初始化、打开或关闭通道的功能函数包(IOM_Fxns)；提交 I/O 数据传输与控制等任务，确保微驱动程序与类驱动程序运行相一致。微驱动程序根据类驱动层发出数据包中的不同参数，对底层硬件采用不同的操作。微驱动层程序使用统一接口标准的功能函数包，应用程序通过调用其中的函数接口就可以由适配模块或 GIO 类驱动调用微驱动，控制底层硬件设备。这些统一接口标准的功能函数包括：

- mdBindDev：绑定设备与微驱动；
- mdCreateChan：创建设备通道；
- mdControlChan：控制设备通道；
- mdDeleteChan：删除设备通道；
- mdSubmitChan：执行 IOM 数据包命令；
- mdUnBindDev：从微驱动层释放设备。

微驱动程序与类驱动层的接口格式是统一的，但对底层硬件的操作是根据具体硬件的不同而变化的。这样就可以在底层硬件发生变动时，无需对整个驱动体系进行改变，而只需修改相应的硬件驱动。提高了系统的可重用性和可移植性。

5.3.2 视频设备驱动开发

在 DSP/BIOS 定义的三种类驱动中，SIO 和 PIP 分别需要使用适配器 DIO 和 PIO 来与微驱动进行通信。而 GIO 类驱动与 SIO/DIO 和 PIP/PIO 不同，它采用基于流的同步 I/O 数据传输模式，内置 IOM 适配层，可以直接与微驱动进行通信，适合大流量数据的传输。

GIO 模块与其他两个模块相比，有一个很重要的特性，就是可以方便的进行扩展，它可以通过扩展 API 函数来支持新的应用领域。这种可扩展 API 的特性可以很好的运用在视频驱动开发方面。例如将视频数据同步机制应用在系统程序时，这种扩展能够添加一个单独的函数来交换视频缓冲区，而这种缓冲区交换的机制在实时视频信号的采集与显示过程中能发挥重要的作用。所以，本文在实现视频驱动时采用通用输入输出模块 GIO，在应用程序中直接地调用 GIO 的 API 函数和 IOM 微驱动程序进行交互。

TI 公司推出的 DM642 芯片是一款面向视频处理的 DSP 芯片，它针对 GIO 模型又做了进一步改进，提出了专门针对视频设备的 FVID 模型。FVID 模型是建立在 GIO 模型之上的，它主要有以下几个 API 函数：

FVID_create：创建通道对象；

FVID_alloc：向应用程序分配视频端口缓冲区；

FVID_control：向微驱动发送控制命令；

FVID_exchange：交换缓冲区；

FVID_free：释放缓冲区；

FVID_delete：删除通道对象。

此外，FVID 模型还拥有一个专门描述视频数据帧的数据结构 FVID_frame，此结构中包含了常用的视频信号的信息，如行数、列数、YUV 结构、场频等。

视频处理流程如图 5.9 所示，先使用 FVID_create 函数创建缓冲区通道，然后调用 FVID_control 函数进行芯片初始化，并开始采集或显示图像。接着调用 FVID_alloc 函数向系统申请采集或播放的数据帧，处理完以后再调用 FVID_exchange 函数将修改后的数据帧返回驱动程序。从图中可以看出，应用层程序调用的 FVID 接口函数会由类驱动层逐步向下映射至微驱动层，而微驱动层程序直接对底层硬件设备进行操作，完成整个视频的采集或显示的过程。

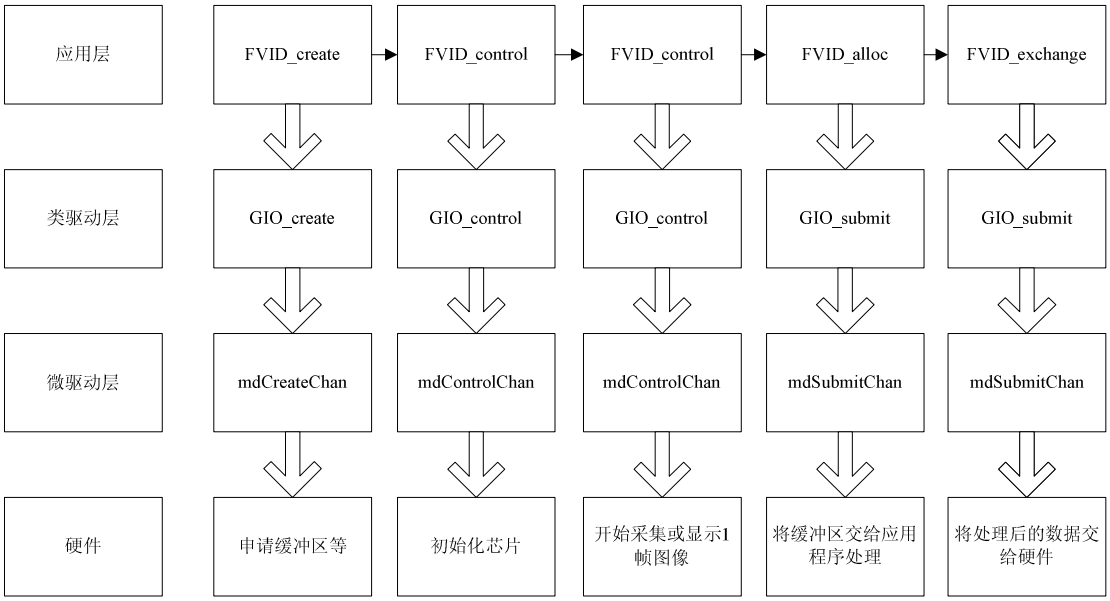


图 5.9 视频处理流程图及相关驱动函数

Fig 5.9 Procedure of video processing and related functions

5.4 RF5 框架

随着技术的发展，DSP 不断向着多功能、高性能、低功耗方向发展，其硬

件技术的更新速度不断加快，然而相关软件开发技术的发展速度却远远落后于硬件的发展速度。开发人员往往困扰于大型 DSP 应用系统复杂的硬件架构和软件设计。eXpressDSP 是 TI 公司提出的一项实时软件开发技术，它集成了 CCS 开发平台、DSP/BIOS、 DSP 算法标准 XDAIS (TMS320 DSP algorithm standard)和第三方支持等四部分，利用它可以加快用户开发 DSP 软件的过程。

Reference Frameworks 是一种 eXpressDSP 技术下的软件框架，它将 DSP 应用程序、XDAIS 算法和底层结构结合在一起，就如同一个应用软件的蓝图，其中已经包含存储器管理、线程模块和通道封装等复杂的基础结构，因此，开发人员可以不用考虑这些基础结构是怎样工作的，而只需要考虑个人的系统特征。到目前为止，TI 公司共提供了 RF1、RF3 和 RF5[40]三种框架标准，其中 RF5 能支持 1～100 个通道和大量符合 eXpressDSP 标准的算法，可应用于基于多通道下复杂算法的应用程序中。与低标准的参考框架相比，RF5 不但支持更多的通道和算法，而且用到了线程模块，因此，RF5 非常适合应用于线程间有着复杂联系的应用程序，如视频处理等。

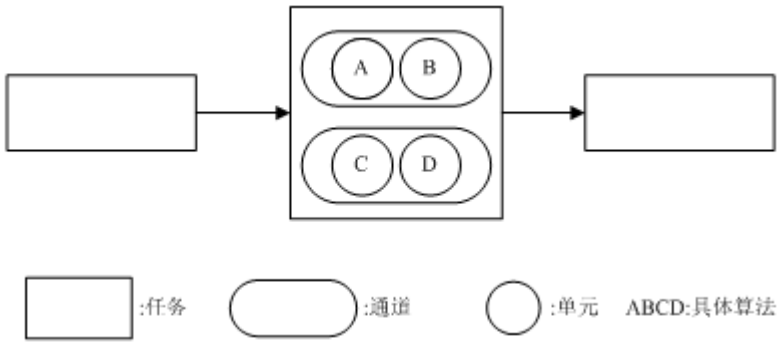


图 5.10 RF5 结构示意图

Fig 5.10 Components in RF5

RF5 分为任务、通道、单元和标准算法共 4 个基本的数据处理部件，它们之间的关系如图 5.10 所示。

由上图可以看出，一个任务可以包含一个或多个通道，每个通道中可以包含一个或多个单元，而每个单元中则封装了一个算法。利用任务可以同时处理一个或多个通道，这样可以在任务间惊醒数据通信和设备驱动会话。利用通道可以按顺序执行多个单元。开发过程中，任务与通道的不同点在于，任务有由用户编写的具体的执行代码，该部分代码通常是与外界进行数据通讯、控制通道执行等。每个任务都是在死循环中不断执行自己的代码，完成检查控制信息、接收数据、执行通道、发送数据等操作。

（1）任务（Task）：

任务处于四个基本数据处理元素中的最顶层，它总是顺序的执行所包含的通道，任务在一个比较高级的级别上把数据组织在一起，它们可以与别的任务，设备驱动以及别的类似结构进行通讯。每个任务都在不断的等待消息，处理数据，并将结果发送给其他的任务，同时有可能还要发送同步消息给其他任务已实现任务间的通讯。任务是进行数据处理的基本单元，有的任务只是做简单的数据处理，它可以不包含任何的通道；有的则是处理是相对复杂的过程，有可能需要包含多个通道。

（2）通道(Channel):

RF5 提供了通道结构的目的是为了更方便的封装和执行算法，一个通道包含一组单元（Cell），其主要工作就是顺序地执行所包含的单元，具体流程为：首先初始化通道模块，然后建立通道对象，注册该通道所包含的单元对象，接着依次执行每个单元，执行完成后销毁对象，最后退出。每个通道可以包含多个单元，每个单元进行初始化后都要注册。

（3）单元（Cell）：

基于 RF5 的应用常常包含大量的算法，为了方便管理算法，RF5 提出了单元的概念，一个单元就是包含一种 XDAIS 算法的容器，通道通过单元来调用算法。单元并不做数据处理，它只是提供一个调用算法的接口，真正做数据处理的是包含在单元中的这些 XDAIS 算法。这样的结构设计能规范开发流程，便于维护、扩展和移植。

（4）算法：

此部分即为实现具体功能的算法，但是必须按照 eXpressDSP 算法标准开发，使用标准的资源管理接口。

本系统在实现时，考虑到系统整体实现的复杂度以及算法的数量，决定简化 RF5 的使用，仅采用任务级的实现。在图像处理任务中直接调用人脸检测算法处理数据，而没有使用到通道和单元。这样，RF5 框架下的数据通信模式也仅使用到了任务级的数据通信：同步通信 SCOM(synchronization communication)。

SCOM 消息是用户自定义的一个数据结构，用来描述任务申请的数据缓冲区，其目的是在任务之间进行数据交换。为实现数据交换，任务申请若干个一定大小的数据缓冲区，以供其他任务数据进行数据读写。每个任务需要知道其他任务的缓冲区位置，并阻止多个任务同时访问自己的缓冲区。任务通过调用 SCOM_putMsg 函数将 SCOM 消息放置一个 SCOM 队列中，发送给其他的任务，或者通过调用 SCOM_getMsg 函数从队列中获取消息。一般情况下，发送消息指明接收线程所要读取的数据缓冲区的地址（以指针形式），接收消息指明发送线程所

要写入的数据缓冲区的地址。

SCOM 可以理解为同步标记，它可以用来判断模块内存是否正在被其他任务所使用，这样就能防止内存访问冲突。整个系统中包含很多存储区，这些内存区很有可能在某一时刻正在被某一任务访问，而使用 SCOM 消息能保证一块内存在任意时刻只有一个任务在访问。当它访问完后，放弃了这一内存块的占有权，再通过 SCOM 消息来允许相关联的任务访问。

6 人脸检测系统的实现

本文在第三章中详细介绍了系统采用的 Adaboost 算法，在第四章中描述了系统硬件的搭建方案，并在第五章中采用了软件开发平台和框架。在这一章中，本文将介绍如何在搭建好的软硬件平台上实现人脸检测系统。

本系统的流程图如图 6.1 所示，在图像采集和图像显示两大模块间插入人脸检测模块，如果检测到人脸，则标定出人脸的位置大小等信息，并在电视机上显示出来；如果没有人脸，则直接显示采集到的图像。

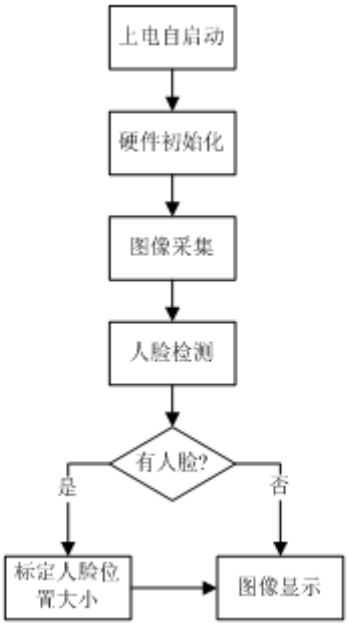
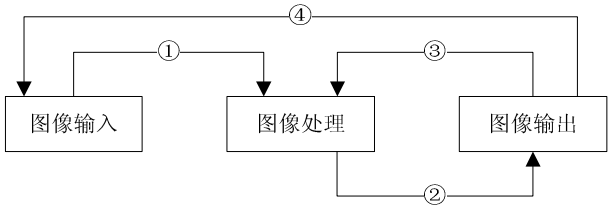


图 6.1 人脸检测系统流程图

Fig 6.1 Work flow of face detection system

系统基于 eXpressDSP 技术下的 RF5 框架开发，将系统主要功能分化为图像输入，处理和输出三个任务，任务并行工作，相互间使用消息通信（消息内包含图像数据），如图 6.2 所示：



①采集完毕 ②处理完毕 ③输出完毕④开始下一帧采集

图 6.2 系统任务间通信图

Fig 6.2 Communication among system tasks

- 图像输入任务采集完一帧图像后，发送消息①给处理任务，示意图像采集完毕；
- 处理任务在接到输入任务的消息①和输出任务的消息③后，开始图像处理，处理结束后发送消息②通知输出任务图像已处理完毕，同时将处理后的图像数据发送给输出任务；
- 输出任务接到数据后将其显示在电视机上，输出完毕后发送消息③给处理任务，示意输出完毕，可以传入下一帧播放图像数据，同时发送消息④给输入任务，示意可以开始下一帧原始图像的采集。

下面将按顺序依次介绍系统各个模块。

6.1 自启动

6.1.1 自启动的原理

DM642 启动方式是通过上电时采样引脚 AEA[22，21]的电平来设置的。可以选择 NO BOOT（AEA[20，19]均为低电平）和从外部存储器自启动（AEA[20，19]均为高电平）以及 HPI/PCI 启动。

如果选择从外部存储器 Flash 启动，则在系统结束复位状态后，DM642 将 Flash 空间(位于 CE1 空间，起始地址 90000000H)的前 lkB 复制到片内 0 地址开始的存储单元中，并从 0 地址开始执行。这一次加载由 DSP 自带的一级 BootLoader 自行完成，但是 lKB 的程序作为用户程序显然不够，因此，这 1K 的程序要做成加载器，也就是二级 BootLoader，利用它把大型用户程序从 Flash 搬入内存。当装载完成后，指向程序的入口地址 c_int00()处并开始执行应用程序，这个过程也被叫作二次加载[41]。

二级 BootLoader 主要完成 3 个功能：

①初始化 EMIF 口，配置寄存器；

②从 Flash 的前半页的 1kB 以后开始把程序的各个段和数据拷贝到指定的存储器物理地址中；

③跳到 C 程序的人口点处 c_int00()，这里要先修改 ISP，再跳转到复位中断，因此在二级 BootLoader 的最后总是一条跳转指令。由于二级 BootLoader 在 Flash 中的位置是 0x90000000—0x90000400，而二级 BootLoader 又是放在用户程序里的，因此，在编译用户程序时，应该将 Flash 开始的 1KB 空间配置成 Flash-Boot 空间，专门用来存放用于二次加载的二级 Boofloader，1KB 以后从 0x90000400 开始的 Flash 空间才用来存放用户程序。

使用二级 BootLoader 加载应用程序的过程图 6.2 所示：

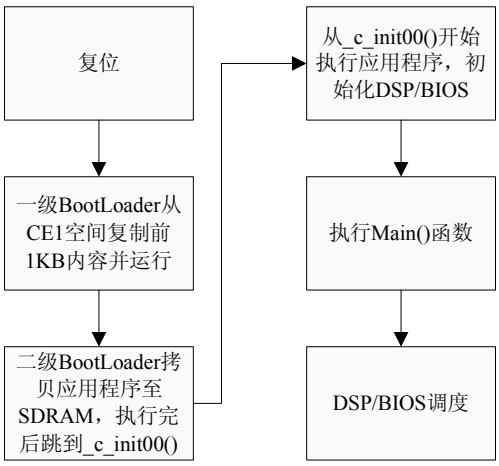


图 6.3 DM642 自启动过程

Fig 6.3 Boot procedure of DM642

6.1.2 自启动的实现过程

实现 DM642 的自启动可以分为以下几个步骤：

①建立应用程序

TI 代码产生工具产生的目标文件是一种模块化的文件格式，即 COFF（Common Object File Format）格式。程序中的代码和数据在 COFF 文件中是以段的形式组织。COFF 文件是由文件头、段头、符号表以及段数据等数据结构组成；编译器会将我们的应用程序转换为 COFF 格式.out 文件，编译器通过一个连接命令文件（.cmd）将不同的程序段放至对应的存储空间中。这个文件里面就包含 DSP 和目标板的存储器空间的定义以及代码段、数据段是如何分配到这些存储器空间的。

相比普通的 DSP 应用程序，自启动的应用程序在配置上有两点不同：

DSP/BIOS 的 MEM 模块中增加一个 BOOT 段，将 CE1 空间的前 1KB 配置用于存放二级 BootLoader 程序；

.cmd 文件中，增加一段配置命令，将 bootloader 程序拷贝至 BOOT 段中。配置命令如下所示。

```
SECTIONS
{
    .boot_load              > BOOT
}
```

②编写二级 BootLoader 程序

二级加载程序写在一个汇编程序 boot.asm 之中，主要是用来初始化化 EMIF 接口，拷贝应用程序以及调转程序入口，需添加到应用程序之中。关键代码片段如图 6.3-6.5 所示：

```
;*************************************************************************
;* Configure EMIF
;*************************************************************************

        mvkl   emif_values, a3  ; load pointer to emif values
        mvkh   emif_values, a3

        mvkl   EMIF_BASE, a4    ; load EMIF base address
        mvkh   EMIF_BASE, a4

        mvkl   0x0009, b0       ; load number of registers to set
        mvkh   0x0000, b0

emif_loop:
        ldw    *a3++, b5        ; load register value
        sub    b0,1,b0          ; decrement counter
        nop    2
 [ b0]  b      emif_loop
        stw    b5, *a4++        ; store register value
        nop    4
```

图 6.4 配置 EMIF 接口代码片段

Fig6.4 Codes of configuring EMIF

```
;*************************************************************************
;* Copy code sections
;*************************************************************************
        mvkl   COPY_TABLE, a3   ; load table pointer
        mvkh   COPY_TABLE, a3

        ldw    *a3++, b1        ; Load entry point

copy_section_top:
        ldw    *a3++, b0        ; byte count
        ldw    *a3++, a4        ; ram start address
        nop    3

 [!b0]  b copy_done              ; have we copied all sections?
        nop    5

copy_loop:
        ldb    *a3++,b5
        sub    b0,1,b0          ; decrement counter
 [ b0]  b      copy_loop        ; setup branch if not done
 [!b0]  b      copy_section_top
        zero   a1
 [!b0]  and    3,a3,a1
        stb    b5,*a4++
 [!b0]  and    -4,a3,a5         ; round address up to next multiple of 4
 [ a1]  add    4,a5,a3          ; round address up to next multiple of 4
```

图 6.5 拷贝代码片段

Fig 6.5 Codes of copying data

```
;*************************************************************************
;* Jump to entry point
;*************************************************************************
copy_done:
        b      .S2 b1
        nop    5
```

图 6.6 跳转代码片段

Fig 6.6 Codes of jumping to the enter point

③将 COFF 文件转为二进制文件

编译器生成的 COFF 格式的.out 文件不能直接烧写，现行的方法一般都是先把待加载程序的.out 文件转成 HEX 格式，然后去掉 HEX 格式文件的文件头，再通过烧写程序写到 Flash 里去。本系统使用的是开发板商提供的 out2hex 工具将.out 文件转化为.hex 文件。

④将二进制文件烧录到 Flash 中

Flash 的写操作和读擦除操作必须通过往指定地址写指定命令的方法来实现。写入时只有数据为 0 时才进行写入，数据为 1 时则什么也不做，所以每次烧写之前都需要将 Flash 擦除干净。本系统使用开发板商提供的 burnboot 程序通过仿真器的 JTAG 接口将 BootLoader 下载到板子的 SDRAM 中，然后再烧写到 flash 中。

6.2 硬件初始化

系统开始运行后所作的第一件事情就是初始化硬件。在 5.2.3 节所述，用户的初始化函数为_dm642_init，所以初始化硬件的操作主要就在该函数中完成，函数实现如图 6.6 所示。

```
void dm642_init()
{
    EMIFA_configArgs(
        0x00092078,
        0xffffffd3,
        0xF3A88E02,
        0x22a28a22,
        0x22a28a42,
        0x57115000,
        0x0000081b,
        0x000a8529,
        0x00000002,
        0x00000002,
        0x00000002,
        0x00000073
    );

    CACHE_setL2Mode( CACHE_256KCACHE );
    CACHE_clean(CACHE_L2ALL, 0, 0);
    CACHE_enableCaching(CACHE_EMIFA_CE00);
    CACHE_enableCaching(CACHE_EMIFA_CE01);

    // Init the EVM
    EVMDM642_init();
    EVMDM642_LED_init();
}
```

图 6.7 硬件初始化函数

Fig 6.7 Hardware initialization function

硬件初始化包括 DM642 芯片内部的初始化和外设芯片的初始化两部分，其中在外设芯片初始化前又必须对 EMIF 进行初始化。在函数 EVMDM642_init()中主要实现芯片支持库（CSL）、内部集成电路（IIC）、通用 IO 接口（GPIO）、串口等部分的初始化，以及视频端口的使能等。

下面以串口初始化为例，说明硬件初始化的过程和原理。串口初始化函数 commConfig()
在 EVMDM642_init()中调用，具体实现如下：

```
/*******************************
void commConfig()
{
  set_baud(0,13);//57.6Khz
  set_efr(0xd0);
  set_fcr(0x0f);
  set_mcr(0x00);
  set_tcr(0x1f);
  set_tlr(0xf8);
  set_xon_xoff(0,0,0,0);
  set_lcr(0x03);
  set_ier(0x00);
}
```

图 6.8 串口初始化函数

Fig 6.8 Serial port initialization function

图中的函数是对串口寄存器的设置，实现方法为：找到寄存器地址后，修改其值，如图 6.8 所示。具体寄存器的修改数值和使能顺序需要参照芯片说明书（datasheet）进行设置。

```
/*set lcr*********************************
void set_lcr(Uint8 val)
{
  volatile  Uint8 *adder;
  adder=(Uint8 *) UARTA_LCR_ADDER;
  *adder=val;
}
```

图 6.9 lcr 寄存器设置函数

Fig 6.9 Set the LCR register

6.3 图像采集与显示

如在 5.3.2 节中介绍的，本系统使用基于 DSP/BIOS 的 FVID 来实现视频设备的驱动，FVID 向上层程序提供视频设备的调用接口，向下屏蔽了具体设备的实现。

图像采集和显示是两个并行的任务，每个任务都分为初始化，启动和执行三部分。其中初始化和启动两部分基本上都是相同的，使用各自芯片的参数调用 FVID 的 API 进行配置设置。在执行函数中，则必须要考虑到任务同步的问题。

本系统采用的 SCOM 消息结构定义如图 6.10 所示：其中包含的关键数据有图像 Y,U,V 三个分量的指针。

```
typedef struct _ScomMessage {
    QUE_Elem queElem;
    int  sizeLinear;
    void *bufLinear;
    void *bufY;
    void *bufU;
    void *bufV;
} ScomMessage;
```

图 6.10 SCOM 消息定义代码

Fig 6.10 Structure of SCOM message

采集任务在采集完一帧图像后，将图像数据包含在消息中发送给处理任务，并等待显示任务的消息，以便开始下一帧图像的采集，相关代码如图 6.11 所示。

```
while ( 1 )
{
    pMsgBuf->bufY = capFrameBuf->frame.iFrm.y1;
    pMsgBuf->bufU = capFrameBuf->frame.iFrm.cb1;
    pMsgBuf->bufV = capFrameBuf->frame.iFrm.cr1;

    /* Send message to process task with pointers to
       captured frame */
    SCOM_putMsg( fromInputToProc, pMsgBuf );

    /* Capture a new frame */
    FVID_exchange( capChan, &capFrameBuf );

    /* Wait for the message from the output task */
    SCOM_getMsg( fromOutToInput, SYS_FOREVER );
}
```

图 6.11 图像采集任务同步代码

Fig 6.11 Synchronize the video capture

显示任务则先发送消息给处理任务示意其将处理后的图像数据传过来，获得图像数据后，将其显示出来，并发送消息给采集任务，相关代码如图 6.12 所示。

```
while ( 1 )
{
    pMsgBuf->bufY = (void *)disFrameBuf->frame.iFrm.y1;
    pMsgBuf->bufU = (void *)disFrameBuf->frame.iFrm.cb1;
    pMsgBuf->bufV = (void *)disFrameBuf->frame.iFrm.cr1;

    /* Give the process thread the buffer to copy the data */
    SCOM_putMsg( fromOutToProc, pMsgBuf );

    /* Wait for the message from the process task to display new
       frame */
    SCOM_getMsg( fromProcToOut, SYS_FOREVER );

    /* Display the decoded frame.*/
    FVID_exchange( disChan, &disFrameBuf );

    /* Send message to input task to continue */
    SCOM_putMsg( fromOutToInput, pMsgBuf );
}
```

图 6.12 图像显示任务同步代码

Fig 6.12 Synchronize the video display

6.4 OpenCV 移植

6.4.1 OpenCV 简介

为了保证正确实现算法和降低开发难度，本系统引入了开源计算机视觉库 OpenCV[42]。OpenCV 是 Intel 公司开发的开源图像处理和计算机视觉函数库。它由一系列 C 函数和少量 C++ 类构成，实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV 包含的主要函数库有 cv.lib，cxcore.lib，highgui.lib 三个。cvcore.lib 中包含了图像处理中基础的数据结构和操作这些数据结构的函数方法，其中主要的基础数据结构又分为静态结构如点（CvPoint）、矩阵（CvMat）、图片（IplImage）等和动态结构如内存块（CvMemStorage）、序列（CvSeq）、集合（CvSet）等，主要的函数包括对数组、矩阵、图片的操作以及一些常用的绘图函数。cv.lib 中包含了图像处理中的一些常用的基础算法函数，如图像处理中的边缘、角点检测算法，机器学习中贝叶斯算法，决策树算法等。highgui.lib 则为 PC 机用户提供了简单易用的图形用户接口。更高一级的算法如立体图像处理、运动跟踪检测等算法包含在 cvAux.lib 等库中，但是在本系统中没有使用到。

为了快速的实现和检测算法的正确性，本文先在 PC 机上开发出人脸检测系统，然后再将系统移植到 DM642 开发板。

6.4.2 EMCV 简介

EMCV（Embedded Computer Vision Library）是 OpenCV 在 TI C6000 系列芯片上移植版本，它是 2008 年 7 月在 sourceforge 上立项的一个开源项目。目前 EMCV

已经移植了 IplImage, CvMat，CvSeq 等基础数据结构，可以对矩阵进行基本操作，并且可以创建和释放图片。但是对于图像处理的通用算法以及算法中的使用到的某些数据结构和函数，EMCV 都还没有移植。

6.4.3 移植 OpenCV

由于 DSP 开发的特定环境，OpenCV 的函数库不能直接在 CCS 中使用，而且人脸检测所需的数据类型、数据结构和函数等只占 OpenCV 其中的一部分，又因为 EMCV 已经提供了移植好的一部分基础数据结构和函数，所以本文结合 EMCV，将 OpenCV 中的人脸检测部分代码进行裁剪、移植，并针对 DM642 的软硬件特点进行了优化。关键的函数有：

```
CV_IMPL void*
cvLoad( const char* filename, CvMemStorage* memstorage, const char* name,
        const char** _real_name ) //装载分类器文件
CV_IMPL void
cvIntegral( const CvArr* image, CvArr* sumImage,
        CvArr* sumSqImage, CvArr* tiltedSumImage ) //计算积分图
CV_IMPL CvSeq*
cvHaarDetectObjects( const CvArr* _img, CvHaarClassifierCascade* cascade,
        CvMemStorage* storage, double scale_factor,
        int min_neighbors, int flags, CvSize min_size ) //检测人脸区域
```

移植时参照 PC 机上的实现，采用逐层分析的方法不断向下分解，直至找到 EMCV 中已经提供的基础数据结构，或者自己构造最底层的数据结构。

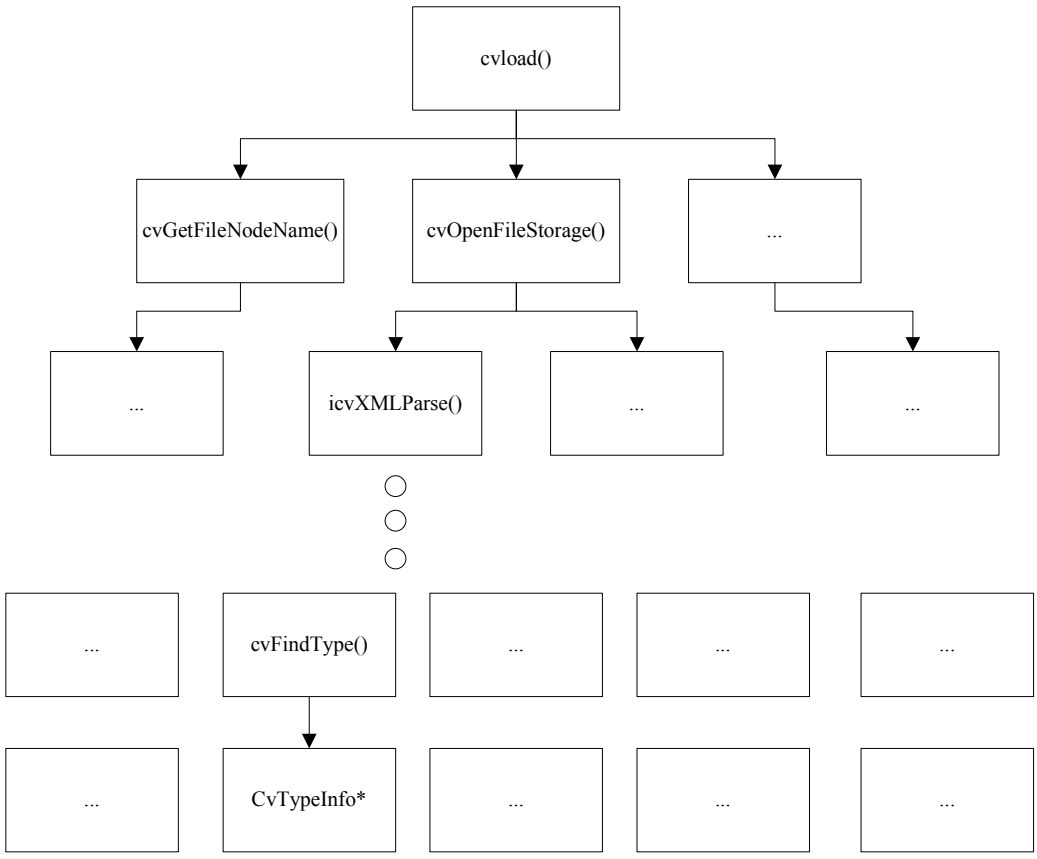


图 6.13 OpenCV 移植示意图

Fig 6.13 Procedure of transplating OpenCV

如图 6.13 所示，本文在移植 cvLoad()函数时，发现其调用了 cvOpenFileStorage()等函数，而这些函数又向下调用了另外的函数，如 icvXMLParse()等，这样一层一层向下分解，直到最后一层函数，如 cvFindType()，而在这个函数的实现中，用到 CvTypeInfo*这个结构指针，而 CvTypeInfo 这个结构已经由 EMCV 提供，这样到这一层就不必再向下分解了。如果发现最后一层中所需的数据结构是 EMCV 没有提供的，那么就必须参照 OpenCV 实现。

移植过程中，本文将所有函数和数据结构按树状结构排列，这样便于查找和发现问题，代码如 6.14 所示。

```
/*Level 2 func: used in cvLoad()*/
CV_IMPL const char*
cvGetFileNodeName( const CvFileNode* file_node )
{
    return file_node && CV_NODE_HAS_NAME(file_node->tag) ?
        ((CvFileMapNode*)file_node)->key->str.ptr : 0;
}

/*Level 1 func: used in process.c*/
CV_IMPL void*
cvLoad( const char* filename, CvMemStorage* memstorage,
        const char* name, const char** _real_name )
```

图 6.14 树状结构的函数代码

Fig 6.14 Functions in tree-structures

6.5 人脸检测过程的实现

人脸检测在图像处理任务中实现。处理任务在初始化时，先注册一个分类器模型，然后装载训练好的分类器文件，失败则将错误信息输出。

在进行人脸检测时，处理任务在获得图像数据后，将彩色图像的灰度值提取出来，保存为待检测图像，接着使用 Adaboost 算法对图像进行扫描，以检测出所有的人脸，然后在图像上用矩形框将所有人脸标定出来，最后将处理完的数据发送给显示任务，在电视机上显示出来。相关代码如图 6.15，6.16 所示。

```
void ProcessInit()
{
    CvTypeInfo _info;
    _info.flags = 0;
    _info.header_size = sizeof(_info);
    _info.type_name = CV_TYPE_NAME_HAAR;
    _info.prev = _info.next = 0;
    _info.is_instance = icvIsHaarClassifier;
    _info.release = (CvReleaseFunc)cvReleaseHaarClassifierCascade;
    _info.clone = icvCloneHaarClassifier;
    _info.read = icvReadHaarClassifier;
    _info.write = icvWriteHaarClassifier;

    cvRegisterType( &_info );

    cascade_name = "e:\\haarcascade_frontalface_alt2.xml";
    cascade = (CvHaarClassifierCascade*)cvLoad(cascade_name, 0, 0, 0);

    if(!cascade)
    {
        fprintf( stderr, "ERROR: Could not load classifier cascade\n" );
//      exit(1);
    }
}
```

图 6.15 图像处理任务的初始化函数

Fig 6.15 Init function of image processing

```
while(1)
{
    storage = cvCreateMemStorage(0);
    pMsgBuf = SCOM_getMsg( fromInputToProc, SYS_FOREVER );

    inBuf[0] = pMsgBuf->bufY;
    inBuf[1] = pMsgBuf->bufU;
    inBuf[2] = pMsgBuf->bufV;
    //set the img as a gray image
    img->imageData = pMsgBuf->bufY;

    faces = cvHaarDetectObjects(img, cascade, storage, 1.1, 2, 0 );//1.
    //faces = cvHaarDetectObjects(img, cascade, storage, 1.1, 2, 0,cvSi

    // draw the faces out
    for( i = 0; i < (faces ? faces->total : 0); i++ )
    {
      CvRect* r = (CvRect*)cvGetSeqElem(faces, i);
      draw_rectangle(img->imageData,720,576,r->x,r->y,r->width,r->heigh
    }
    pMsgBuf = SCOM_getMsg( fromOutToProc, SYS_FOREVER );
    outBuf[0] = pMsgBuf->bufY ;
    outBuf[1] = pMsgBuf->bufU ;
    outBuf[2] = pMsgBuf->bufV ;
    for(i = 0; i < 576; i ++)
    {
        DAT_copy((Uint8 *)inBuf[0]  + i * 720,(Uint8 *)outBuf[0] + i *
        DAT_copy((Uint8 *)inBuf[1]  + i * 360,(Uint8 *)outBuf[1] + i *
        DAT_copy((Uint8 *)inBuf[2]  + i * 360,(Uint8 *)outBuf[2] + i *
    }
    cvClearMemStorage( storage );
    DAT_wait(DAT_XFRID_WAITALL);
    SCOM_putMsg(fromProcToOut,pMsgBuf);
}
```

图 6.16 图像处理任务的关键代码

Fig6.16 Key function in the processing task

6.6 程序优化

由于系统的实时性要求高，因此需要对 DSP 程序进行优化，以提高检测速度。基于 DSP 的程序优化可以分 2 层[43]：C 语言层的优化和汇编层的优化。一般在 C 语言层优化后看系统性能是否达标，以便决定是否再在汇编层进行优化，即将调用次数多，执行时间长的函数用汇编语言实现。

本系统在 C 语言层采用的优化方法如下：

（1）浮点转定点运算：由于 DM642 是一款定点 DSP，所以将程序中的浮点运算转为定点运算后，性能得到大幅度提高；

（2）使用 EDMA：由于图像数据量大，普通的拷贝效率低下，因此使用 EDMA 做数据搬移，它可在没有 CPU 参与的条件下完成数据拷贝。

（3）尽量使用内联函数：频繁的函数调用将导致系统额外开销的增多，因此尽量将一些功能简单的函数写成内联函数；

（4）使用编译器选项进行优化：在编译器中提供了分为若干等级和种类的自

动优化选项，本系统使用的如下：

●-o：使能软件流水和其他优化方法；

●-pm：使能程序级优化；

●-mt：使能编译器假设程序中没有数据存储混淆，可进一步优化代码。；

●-ms：确保不产生冗余循环，从而减小代码尺寸。

（5）删除不必要的函数和调试代码等：系统正常工作后，可将调试代码和一些不必要的函数删除，以减少系统运行开销，保证代码的紧凑。

6.7 实验结果

本系统选用的 DM642 开发板如图 7.1 所示，实验硬件环境如图 7.2 所示，实验结果如图 7.3-7.6 所示。

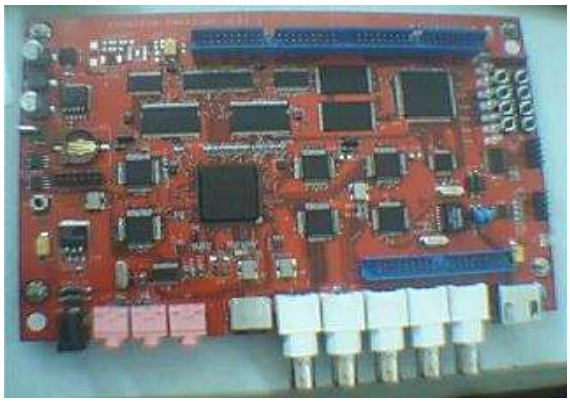


图 7.1 DM642EVM 开发板

Fig 7.1 DM642EVM board

图 7.2 实验硬件平台

Fig 7.2 The hardware platform

图 7.3 实验结果 1

Fig 7.3 Experiment result 1

图 7.4 实验结果 2

Fig 7.4 Experiment result 2

图 7.5. 实验结果 3

Fig 7.5 Experiment result 3

图 7.6. 实验结果 4

Fig 7.6 Experiment result 4

从实验结果 1 和 2 可以看出，本系统能检测出图像中的多张正面人脸，实验结果 3 表明本系统存在一定的误检率，而实验结果 4 表明本系统对偏转角度较大的人脸检测不是很理想，这是因为训练样本集的原因造成的。

在检测速度方面，对于原始大小的采集图像（720×576），系统的检测速度为 2.6 帧/秒，而将图像压缩成通用的 CIF（Common Intermediate Format）格式大小（352×288）后，检测速度可以达到 10 帧/秒，基本上满足了实时性要求。

7 结　论

7.1 本文所作的工作总结

经过二十多年的研究，人脸检测技术得到了长足的发展，但是由于光照，观察角度等外界因素和表情，姿态等内部因素，至今还没有出现一种适用于所有场合的人脸检测算法。各种算法都是在各自适用领域得到了较好的检测结果。

DSP 芯片具有运算速度快、体积小、功耗低等特点，而TI公司推出的DM642 芯片是目前市场上具有较高性能的DSP芯片，因此将高性能的芯片和当前较成熟的研究成果结合起来就具有重要的现实意义。

本文以 CCD 摄像头，DM642 开发板和电视机搭建了一个人脸检测系统。系统从 CCD 摄像头采集图像后，经过人脸检测处理，标定出人脸的位置和大小等信息，并将结果在电视机上显示出来。具体完成的工作可以分为以下几点：

（1）针对本课题要求，研究并实现了 Adaboost 人脸检测算法；

（2）以 TMS320DM642 芯片为处理器，选择合适的外围芯片和设备搭建了人脸检测系统硬件平台；

（3）研究 DSP 程序的开发流程及框架，并以此为基础开发出人脸检测系统；

（4）裁剪并移植开源计算是视觉库 OpenCV；

（5）基于 CCS 开发环境和 DM642 的硬件特点对系统进行优化。

7.2 后续工作及展望

由于毕业设计的时间有限，本人又是第一次接触 DSP，另外在人脸检测的理论和技术积累上非常有限，本次毕业设计虽然基本上完成了课题目标，但是仍有许多地方可以改进。

首先在算法的实现方面，本系统采用了第三方函数库 OpenCV 提供的一些数据结构和函数接口，但是实验表明，OpenCV 在某些细节上处理的效率还有待改善，本系统进行了一些优化，但是程度还不够，某些关键函数可以考虑自己编写。

另外在算法的选择方面，Adaboost 算法是一个鲁棒性和实效性都很强的方法，但是基于统计特征的算法有其先天性的不足，比如光照的影响，人脸姿势的影响，以及在灰度图上对似人脸区域的判断等方面都可以加以改进。这里可以考虑与其他算法相结合来实现检测，但是这样又会对系统的实时性带来影响，因此需要充分考虑这一矛盾。当然这也是人脸检测算法的一大难题。

在系统的扩展方面，由于人脸检测是所有人脸技术应用的基础，所以本系统的应用领域可以在多方面扩展，比如加入人脸识别模块，可以作为安检和门禁系统；

加入人眼位置检测和状态检测，可以作为疲劳驾驶预警系统等。

总之，随着人脸检测技术的不断发展和嵌入式芯片性能的不断提高，嵌入式人脸检测系统一定能在越来越广阔的领域得到发展和应用。

致　　谢

论文的结束标志着研究生阶段的学习生活马上要结束了。在论文的完成过程中，得到了很多人的帮助，在此表示真挚的谢意！

首先感谢我的导师张小洪副教授，感谢他在学术上对我的悉心指导和帮助。张老师从理论知识、设计思想到实验方法等各方面的指导和帮助，使我从中获益匪浅。张老师严谨的治学态度和勇于创新的科研精神是我学习的榜样。从张老师身上学到的宝贵财富将伴随我的一生。

其次要感谢洪明坚讲师的大力帮助。从前期的论文选题，到中期的实验指导，再到后期的论文撰写，洪老师给予了全面的指导，倾注了大量的心血，洪老师的严格要求和谆谆教诲使我在论文研究中不断取得进步。论文完成过程中能得到洪老师的帮助是我的幸运。

非常荣幸能在极富学术氛围的实验室中度过二年半的硕士研究生生活，感谢实验室每一位成员对我的帮助，与他们一起学习和生活将是我一生中最美好的回忆。他们是李博、葛永新、刘威、雷明、孙丽鹃、郑钢、孙向南、王宇琛、廖勇军、游磊、廖训佚、赵鹏、杨卓、吴林、李辉、王洪星、闫卫杰、胡金凤、周小龙、赵红、苟若愚、林晓泽、李相军。

另外还要感谢通信学院的张建慧同学，生物工程学院的赵灿、蔡春明同学，论文的完成离不开他们的帮助。

最后，我要感谢我深爱的父母，是他们在精神和物质上给予了我充分的支持，使我顺利的完成学业。谨以此文献给我的父母！

聂江浩

二 OO 九年十月　　于重庆

参考文献

[1] 梁路宏,艾海舟,徐光祐,张钹. 人脸检测研究综述[J]. 计算机学报, 2002, 5:449-458.

[2] 李耀东,崔霞,肖柏华,戴汝为. 自动人脸识别技术综述[J]. 计算机科学, 2002, 12:1-16.

[3] T.Sakai, M.Nagao, S.Fjibayashi. Line Extraction and Analysis Pattern Detection in a Photograph. Pattern Recognition[C]. vol.1:233-248, 1969.

[4] 梁路宏, 艾海舟, 何克忠. 基于多模板匹配的单人脸检测[J]. 中国图象图形学报, 1999, 4A(10):823-830.

[5] Govindaraju V, Srihari S N, Sher D B. A computational model for face location[C]. In: Proc IEEE Conference on Computer Vision, Osaka, Japan, 1990:718-721.

[6] Yoo T W, Oh I S. A fast algorithm for tracking human faces based on chromatic histograms[C]. Pattern Recognition Letters, 1999, 20 (10):967-978.

[7] Garcia C, Zikos G, Tziritas G. Face detection in color images using wavelet packet analysis[C]. In: Proc Multimedia Computing and Systems, Centro Affari, Florence, Italy, 1999, 1: 703-708.

[8] Yang M H, A huja N. Detecting human faces in color images[C]. In: Proc IEEE Conference on Image Processing. Chicago, 1998:127-139.

[9] Yang G Z, Huang T S. Human face detection in a complex background[C]. Pattern Recognition. 1994, 27 (1):53- 63.

[10] 卢春雨, 张长水, 闻方等. 基于区域特征的快速人脸检测法[J]. 清华大学学报(自然科学版), 1999, 39(1):101-105.

[11] Miao J, Yin B C, Wang K Q et al. A hierachical multiscale and multiangle system for human face detection in a complex background using gravity center-template[C]. Pattern Recognition, 1999,32 (10):1237-1248.

[12] Augusteijn MF, Skufca TL. Identification of human faces through texture-based feature recognition and neural network technology[C]. Proc IEEE Int Joint Conf Neural Networks 1993:392–398.

[13] Y Dai, Y.Nakano. Face-Texture Model Based on SGLD and Its Application in Face Detection in a Color Scene[C]. Pattern Recognition.1996, V01.29, no.6:1007-1017.

[14] Moghaddam B, Pentland A. Probabilistic visual learning for object representation[C]. IEEE Trans Pattern Analysis and Machine Intelligence, 1997, 19 (7):696-710.

[15] Rowley H A, Baluja S, Kanade T. Rotation invariant neural network-based face detection[Z]. Carnegie Mellon University, Pittsburgh PA: Technical Report CMU-CS-97-201,1997.

[16] Rowley H A, Baluja S, Kanade T. Neural network-based face detection[C]. IEEE Trans Pattern

Analysis and Machine Intelligence, 1998, 20 (1):23-38.

[17] Rowley H A. Neural network-based face detection[Z]. Carnegie Mellon University, Pittsburgh PA: Technical Report CMU-CS-99-117,1999.

[18] Juell P, Marsh R. A hierarchical neural network for human face detection[C]. Pattern Recognition. 1996, 29 (5):781-787.

[19] Kouzani A Z, He F, Sammut K. Commonsense knowledge-based face detection[C]. In: Proc Conference on Intelligent Engineering Systems, Budapast, Hungary, 1997:215-220.

[20] Anifantis D, Dermatas E, Kokkinak is G. A neural network method for accurate face detection on arbitray images[C]. In: Proc Conference on Electronics, Circuits and Systems, Pafos, Cyprus, 1999,1:109-112.

[21] Osuna E, Freund R, Girosi F. Training support vector machines: An application to face detection[C]. In: Proc Computer Vision and Pattern Recognition, Puerto Rico, 1997:130-136.

[22] 梁路宏,艾海舟,肖习攀,叶航军,徐光祐,张钹. 基于模板匹配与支持矢量机的人脸检测[J]. 计算机学报, 2002-1:22-29.

[23] Schneiderman H, Kanade T. Probabilistic modeling of local appearance and spatial relationships for object recognition[A]. In: Proc IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, 1998:45-51.

[24] Schneiderman H. A statistical approach to 3D object detection applied to faces and cars[Z]. Carnegie Mellon University, Pittsburgh PA: Technical Report CMU-RI-TR-00-06, 2000

[25] Weber M et al. View point-invariant learning and detection of human heads[Z]. In: Proc Conference on Automatic Face and Gesture Recognition, Grenoble, France, 2000.

[26] Nefian A V, Hayes M H. Face detection and recognition using hidden Markov models[C]. In: Proc IEEE Conference on Image Processing, Chicago, 1998:141-145.

[27] Nefian A V, HayesM H. An embedden HMM based approach for face detection and recognition[C]. In: Proc IEEE Conference on Acoustics, Speech, and Signal Processing, Phoenix, Arizona, 1999, 6:3553-3556.

[28] Meng L, Nguyen T Q, Castanon D A. An image-based Bayesian framework for face detection[C]. In: Proc IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA, 2000:302-307.

[29] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting[Z]. Journal of Computer and System Sciences, 55(1):119-139, 1997.

[30] Paul Viola, Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features[C]. Conference on Computer Vision And Pattern Recognition, Kauai, Hawaii. 2001

(1):511-518.

[31] Paul Viola, Michael Jones. Robust real time object detection[C]. Vancouver, Canada:IEEE ICCV Workshop on Statistical and Computational Theories of Vision, 2001.

[32] Philips Semiconductors. SAA7113H 9-bit Video Input Processor[S], 1999.

[33] Philips Semiconductors. Digital Video Encoder (ConDENC) SAA7120, SAA7121[S], 1997.

[34] TI. TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor[S], 2004.

[35] TI. TMS320C6000 CPU and Instruction Set Reference Guide[S], 2000.

[36] TI. TMS320C6000 Programmer’s Guide[S], 2006.

[37] TI. TMS320C6000 DSP/BIOS User's Guide[S], 2000.

[38] TI. Understanding Basic DSP/BIOS Features[S], 2000.

[39] TI. DSP/BIOS Driver Developer's Guide[S], 2002.

[40] TI. Reference Frameworks for eXpressDSP Software: RF5, An Extensive, High-Density System[S], 2003.

[41] TI. Creating a Second-Level Bootloader for FLASH Bootloading on TMS320C6000 Platform With Code Composer Studio[S], 2004.

[42] 刘瑞祯,于仕琪. OpenCV 教程——基础篇[M]. 北京航天航空大学出版社, 2007, 1.

[43] TI. TMS320C6000 Optimizing Compiler User's Guide Texas Instruments Incorporated[S], 2002.

附　录

作者在攻读硕士学位期间发表的论文

[1] 聂江浩，张建慧. 基于 DM642 的人脸检测系统的设计和优化. 重庆理工大学学报，已录用.