**大连理工大学本科毕业设计（论文）**

**基于视频信息的无人机检测系统设计与实现**

**Design and Implement of UAV Detection Based on Video Information**

学 院（系）： 软件学院

专 业： 软件工程

学 生 姓 名： 李锦涛

学 号： 201392168

指 导 教 师： 金乃高

评 阅 教 师： 覃振权

完 成 日 期： 2017年5月25日

大连理工大学

Dalian University of Technology

# 摘 要

无人机由于其在生活和军事中的广泛应用逐步成为人们的研究热点，但是无人机由于其出色的隐蔽性对人们的生活隐私甚至是国家安全造成了极大的威胁，目前无人机的检测技术成为了人们的迫切需求，无人机的检测系统也变得越来越重要。

本文设计与实现的无人机检测系统是按照客户端与服务器模型设计的，客户端采集图像，对图像进行处理之后，将处理得到的数据传送给服务器。服务器使用本系统设计的建表查表法，得到无人机的坐标数据。服务器建表查表的核心是建表过程，我们假设在某一个时刻客户端的传感器传回一串数据，由于网络传输的不同步，各个传感的系统时间又难以统一，我们就不能确定这一串数据是否对应的是传感器在同一时间点采集到的相同数据，这里本系统采用借助外部事件或者无人机做出特定动作来对传感器进行同步，建表过程完成过后，就可以查表得到无人机的坐标。该系统的核心部分是图像处理的部分，这一部分本系统使用了级联分类器的方法和模板匹配的方法，其中模板匹配方法在Android端实现，级联分类器的方法是在pc端实现。

服务器端处理数据并定位的过程，本系统采用了针孔成像模型，使用了python语言进行了仿真模拟，并且对该方法的误差进行了分析。仿真结果表明服务器端的定位算法能以相对较小的误差计算得到无人机的坐标点。

关键词**：**无人机（UAV）；图像识别；级联分类器；模板匹配；针孔成像模型

**Design and Implement of UAV Detection Based on Video Information**

# Abstract

Unmanned aerial vehicles because of its extensive use in life and military gradually become a research hotspot.The UAV has also caused a great threat to people’s lives and national security because of its excellent concealment. At present, UAV detection technology has become an urgent need for people, UAV detection systems are becoming more and more important.

The design and implementation of the UAV detection system is basedon client and server model. The client collects the image, After processing the image, The processed data is passed to the server. The server uses our design method, get the coordinates of UAV data. The core of the our method is the process of building table. We assume that at some point the client's sensor returns a string of data, As the network transmission is not synchronized, each sensor system time are difficult to unify. We can not determine whether this string of data corresponds to the same data that the sensor collects at the same point in time. Here we use external events or UAV make specific actions to synchronize the sensor. After the completion of the construction process, we can look up the table to get the coordinates of the UAV. The core part of our system is the image processing part.In this part we use the cascading classifier method and template matching method.Template matching method is realized in Android side, cascading classifier method is realized in PC.

The process by which the server processes data and locates it, We used the pinhole imaging model, using the python language for simulation, And the deviation of the method is analyzed. The simulation results show that the positioning algorithm of the server can calculate the coordinates of the UAV by the relatively small deviation.

**Key Words：** unmanned aerial vehicle（UAV）；image recognition；cascade classifier；template matching；Pinhole imaging model

目 录

[摘 要 I](#_Toc484615298)

[Abstract II](#_Toc484615299)

[引 言 V](#_Toc484615300)

[1 绪论 1](#_Toc484615301)

[1.1 选题背景 1](#_Toc484615302)

[1.2 软件开发的目标与意义 1](#_Toc484615303)

[1.3 开发环境 2](#_Toc484615304)

[1.4 论文结构 2](#_Toc484615305)

[2 系统开发环境及相关技术 3](#_Toc484615306)

[2.1 Java Socket网络编程简介 3](#_Toc484615307)

[2.1.1 Socket定义 3](#_Toc484615308)

[2.1.2 建立Socket的基本步骤 3](#_Toc484615309)

[2.2 Java多线程 4](#_Toc484615310)

[2.3 图像处理技术简介 4](#_Toc484615311)

[2.3.1 模板匹配方法 4](#_Toc484615312)

[2.3.2 OpenCV里的模板匹配方法 6](#_Toc484615313)

[2.3.3 级联分类器的方法 7](#_Toc484615314)

[2.3.4 级联分类器的训练过程 8](#_Toc484615315)

[2.4 本章小结 9](#_Toc484615316)

[3 系统分析 10](#_Toc484615317)

[3.1 可行性分析 10](#_Toc484615318)

[3.1.1 经济可行性 10](#_Toc484615319)

[3.1.2 技术可行性 10](#_Toc484615320)

[3.1.3 运行可行性 10](#_Toc484615321)

[3.2 需求分析 11](#_Toc484615322)

[3.2.1 功能性需求 11](#_Toc484615323)

[3.2.2 非功能性需求 11](#_Toc484615324)

[3.3 本章小结 11](#_Toc484615325)

[4 系统设计 12](#_Toc484615326)

[4.1 概要设计 12](#_Toc484615327)

[4.1.1 系统项目背景 12](#_Toc484615328)

[4.1.2 系统总体设计 13](#_Toc484615329)

[4.2 详细设计 14](#_Toc484615330)

[4.2.1 检测系统流程图 14](#_Toc484615331)

[4.2.2 客户端图像处理部分的设计 15](#_Toc484615332)

[4.2.3 网络通信部分的设计 16](#_Toc484615333)

[4.2.4 检测部分的设计 17](#_Toc484615334)

[4.3 本章小结 19](#_Toc484615335)

[5 系统实现 20](#_Toc484615336)

[5.1 图像处理的级联分类器方法的实现 20](#_Toc484615337)

[5.1.1 级联分类器的训练 20](#_Toc484615338)

[5.1.2 目标检测 25](#_Toc484615339)

[5.2 图像处理的模板匹配方法的实现 26](#_Toc484615340)

[5.3 客户端服务端数据传输的实现 29](#_Toc484615341)

[5.4 本章小结 29](#_Toc484615342)

[6 仿真结果及实验 30](#_Toc484615343)

[6.1 仿真算法 30](#_Toc484615344)

[6.2 误差分析 34](#_Toc484615345)

[6.2.1 建表数据量对误差的影响 34](#_Toc484615346)

[6.2.2 坐标系下摄像头数量对误差的影响 35](#_Toc484615347)

[6.3 本章小结 35](#_Toc484615348)

[结 论 37](#_Toc484615349)

[参 考 文 献 38](#_Toc484615350)

[致 谢 39](#_Toc484615351)

# 引 言

通过手机摄像头捕捉图像，然后利用图像识别算法判断图像中是否有无人机，得到无人机的像素坐标，并利用针对本系统设计的建表查表法得到无人机坐标，这是本系统的核心思想，也是本系统采用的关键技术。随着无人机的普及，由于无人机具有出色的隐蔽性，人们的个人隐私以及社会甚至是国家的安全受到了无人机的威胁。无人机检测的基本原理就是在需要检测的空间中部署好手机摄像头传感器网络，然后通过摄像头传感器中传过来图像数据的通过各种图像处理的算法判断无人机是否在图像中，如果在图像中则给出像素坐标位置，通过像素坐标位置利用建表查表操作得到空间坐标。本系统的核心算法和操作就是图像识别算法和建表查表操作，其中建表查表操作通过python的仿真实验验证了其合理性。

本文的结构安排上，主要介绍了图像识别算法的基本思想，以及整个系统的检测定位思想和定位过程，最后通过python的仿真实验的方式对定位的核心思想建表查表操作的可行性进行了验证。该方法在一定的条件下具有很高的检测定位精度，该方法在检测过程中只需要知道传感器网络是否检测到无人机以及检测到的无人机在图像中的像素坐标点，就能定位得到无人机在空间里的坐标，在很大条件下减少了我们对已知条件的需求。软件设计上主要使用了客户端和服务器模型，客户端的安卓程序采集图像和处理图像并把得到的数据通过socket网络编程传给服务器。服务器端的数据处理程序得到无人机坐标。最终通过python的仿真实验验证了服务器端的建表查表操作的可行性。本次设计的特点之处在于结合了安卓平台上的图像处理，网络编程还有利用自己研究的方法，将三者融合在一起构造得到一个实时的基于视频信息的无人机检测系统。

总之，本系统基于客户端和服务器模型，利用图像处理、socket网络通信和自己研究的建表查表的方法达到实时检测无人机的目的，以较低的成本就能保障隐私不受无人机的侵犯，在人们的生产生活中具有很重要的实际意义。

# 1 绪论

## 1.1 选题背景

随着随着科技的发展，无人机也越来越得到大量的普及。无人机价格的持续走低给越来越多的人带来了体验无人机的机会，无人机的高速发展也给我们带来了很多生活便利，也给很多高空作业增加了很多的安全性。但是无人机技术无疑是一把双刃剑，尽管它是利大于弊的，但是无人机的弊端也会在无人机大量泛滥这种大的环境下给我们的生活生产安全带来极大的危害。虽然现在国家加大了无人机管控措施，无人机的持有者必须具有无人机驾驶执照，但是这对于无人机泛滥的滚滚洪流来说无异于是杯水车薪。

无人机各类闯入居民生活区，窥探他人隐私的新闻层出不穷。无人机由于其出色的隐蔽性和自由性，人为的防范措施变得十分的困难，无人机甚至能两次闯入白宫这种顶级机密场所而不被拦截，想人为防范无人机的入侵可想而知难度有多大。同样在机场这种安全要求极高的场所，也经常有无人机威胁飞行安全，给人们的生命安全蒙上了极大的阴影，所以无人机检测系统的需求就变得十分的迫切了。不同于雷达这种昂贵的检测系统，无人机的检测系统必须贴近人们的生活实际，而且必须价格低廉才会显得更加合理，这时候一个价格低廉部署简单的系统才会成为人们的急需，本文所讨论的无人机检测系统是基于视频信息的无人机检测系统，仅仅需要部署的是少数几个手机，利用其视频传感器和网络信号传感器就可以对一定坐标范围内的无人机进行检测，甚至是进行定位，这样的系统满足了价格低廉的特点，而且部署简单，能够大范围的进行推广，这样的系统势必会成为以后无人机检测系统的一种趋势。

无人机正在成为危害公众隐私甚至是危害国家安全的头号公敌，这时候对无人机的检测技术就成为一种不可或缺的技术了，本文实现了一种基于视频信息的无人机检测系统，能实时检测到无人机，从而保护人们的财产安全。

## 1.2 软件开发的目标与意义

本软件是基于Android客户端和Java后台的一款应用软件，主要目标是使用手机摄像头实时捕捉到视频信息中无人机的像素点的位置，根据像素点的位置后台实时计算出无人机在坐标系中的位置。通过这个系统可以利用手机机群，来对无人机进行定位，能起到保护人们的隐私不受侵犯的作用。同时也能让设计者充分了解Android平台的相关开发环境和技术，了解手机平台市场的特点。

## 1.3 开发环境

操作系统：Windows7企业版

开发环境：Android SDK-21、JDK 1.8.0

开发工具：Eclipse、Android studio

数据库系统：MySQL

开发语言：Java、XML语言

## 1.4 论文结构

本论文结构安排如下：

第1章，绪论。介绍了选题背景、软件开发目标与意义、开发环境和发展现状。

第2章，技术背景。介绍了智能手机上的Android操作系统、开发工具Eclipse、Opencv、各种图像处理技术以及对它们的分析比较以及Java的多线程技术和网络通信技术。

第3章，系统分析。从系统设计特点、可行性分析、需求分析等三方面对整个软件开发进行了前期的分析评估。

第4章，系统设计。主要分概要设计和详细设计两部分，对软件的设计过程进行了详细的介绍。

第5章，系统实现。系统的图像识别，客户端服务端的网络通信部分的实现。

第6章，仿真实验。对系统最主要的部分，也就是建表查表定位无人机的部分进行仿真，并进行误差分析

文末，结论与展望。对整个系统的开发流程进行了总结，并对软件之后的发展和应该进行的工作提出了设想。

# 2 系统开发环境及相关技术

## 2.1 Java Socket网络编程简介

### 2.1.1 Socket定义

对于Java Socket编程来说，有两个至关重要的概念，一个是服务端的ServerSocket，另一个就是客户端的Socket了。客户端和服务器通过Socket进行连接，然后就能通过这个连接进行通信了。首先ServerSocket会在服务器端监听某一个端口，当它发现客户端如果有某个Socket试图连接它时，服务器端的ServerSocket会accpet()该Socket的请求，同时在服务器端建立一个Socket与其建立连接进行通信。这样就会有两个Socket，服务端一个，客户端一个。

对于Socket之间的通信原理很简单，Socket跟Socket之间是双向连接的，例如，客户端请求将数据发送至服务器端，服务端连接Socket后处理并返回结果给Socket，客户端接收到服务端传输过来的Socket内容并显示给客户端，所以客户端也可以往对应的Socket输出流里写东西，然后服务端可以从对应的Socket的输入流就可以读出相应的内容。在Java环境下，Socket编程主要是指基于TCP/IP协议的网络编程。

### 2.1.2 建立Socket的基本步骤

建立Socket连接至少需要一对套接字，其中一个在客户端运行，我们称之为ClientSocket，另一个运行于服务端，我们称之为ServerSocket。套接字连接过程可以分为三个步骤：

（1） 服务器监听：服务端套接字并不是监听网络状态，也不是监听具体端口号，而是处于等待状态，等待客户端的连接。

（2） 客户端请求：指的是客户端的套接字提出连接请求，指出服务端的ip地址和套接字端口号，然后就向服务端的套接字提出了连接请求。

（3） 连接确认：当服务端接收到客户端的连接请求或者是监听到客户端的连接请求，服务端会建立一个新的线程，把服务端套接字的信息发给客户端进程，客户端识别出了此描述，双方正式开始建立连接。服务器端此时建立一个新的线程继续监听其他客户端的套接字的连接请求。

## 2.2 Java多线程

线程是程序运行的最小单元，被称之为轻量级进程。线程是进程之中的一个实体，是系统独立调度和分派任务的基本单位。线程不占用系统资源，只拥有程序运行必不可少的小部分资源，它可以与同一个进程的其他各个线程共享占有的系统资源。一个线程可以撤销或创建同一个进程里的其他线程，由于线程之间的制约，线程的运行呈现出三种状态，这三种状态分别是就绪，运行和阻塞。就绪状态是指线程具备运行所具有的所有系统资源，并且可以运行，在等待处理机资源。运行态表示线程正在占有处理机资源，并正在运行。阻塞态表示线程在等待某一个事件的发生，例如一个信号量，或者某一部分的系统资源，逻辑上不具备运行条件，每一个程序都至少有一个线程，多线程的程序具有并发性。

而在Java中线程的生命周期分为五个状态，这五个状态分别为：新建状态，就绪状态，运行状态，阻塞状态和死亡状态。线程对象创建后即进入新建状态，例如Thread t=new mythread();当调用t.start()方法时，则表明线程进入就绪状态。就绪状态表明线程已经做好了准备，随时等待CPU的调度，而不是表明线程执行了t.start()就能立即执行线程。当CPU正式开始调度线程时，线程才正式开始进入运行态，所以就绪态是线程运行必不可少的一个状态。而处于运行状态的线程又因为各种原因失去了处理机的资源进入阻塞态，直到其进入就绪态才能有机会获得CPU的调度资源再度进入运行态。根据进入阻塞态的原因的不同，阻塞态又可以分为三种，这三种分别为：等待阻塞，同步阻塞，其他阻塞。等待阻塞是指线程本身执行sleep()或wait()方法，线程进入等待状态。同步阻塞是指线程获取同步锁失败，进入等待状态。其他阻塞是指线程获取系统资源失败，进入阻塞状态。死亡状态是指线程执行完毕或者是线程执行过程中出现异常，线程退出了生命周期。

## 2.3 图像处理技术简介

### 2.3.1 模板匹配方法

模板匹配方法是图像处理的一个重要的方法之一，把不同的传感器设备在不同的时空条件下获取的对同一背景对象的两幅不同的图像在空间上对准，或者已知一副图像，然后根据这幅图像到另一幅图像中寻找这幅图像的方法就叫做模板匹配。通俗来时，所谓的模板就是一副小的图像，这幅图像中有一个特定的对象或背景，模板匹配就是指在一幅大的图像中搜寻这一小图像目标，已知这幅大的图像中含有这个小目标，且尺寸相同，方向相同，通过特定的算法可以在图中找到小目标，确定其坐标位置。以一张八位图像为例，模板T(m×n个像素)在被搜索图Q(s×h个像素上平移)，被搜索的区域为R(m×n个像素)，其中被搜索区域的i，j为被搜索区域的左上角在被搜索图Q上的坐标。则搜索的范围1 ≤ i ≤ s – m，1 ≤ j ≤ h – n。通过不停的比较R和T的相似程度，来完成匹配的过程，衡量模板T和被搜索区域子图R的相似程度，可以使用以下两种方法计算得出：

其中前一个式子可以展开成以下的形式

从这个式子可以看出，这个算法的第三项只跟模板有关，而第一跟第二项则需要根据被搜索区域的数据得出，随着被搜索区域在原图上移动，这两项的值也会随之变化，当D(i,j)找到了最小值时，就找到了需要寻找的目标，这种算法的匹配效率十分低下，在机器性能一般的情况下，这种匹配算法耗费的时间会变的很大，一张一百万像素点的图像，匹配一个一万像素点的模板，则需要六百秒左右。如果工程中使用这种匹配算法，效果也是得不偿失的。

模板匹配也有很多改进的思想，由于原来的算法需要遍历原图中的每一个像素点，而遍历每一个像素点的时间花费是非常巨大的，所以很多不必要的时间性能开销就浪费在一些没有必要进行的数据比对上了，所以这些改进的模板匹配方法大多都是把性能的提升，用减少数据比对次数的方法来实现的。主要有这些方法：

1. 精度匹配的方法，研究原来的模板匹配方法的实验数据，可以发现这样一个现象，在接近最小误差的被搜索区域的比对误差D(i,j)会出现一个断崖式下降，会跟其他的匹配位置出现很大的不同。对于这一现象，我们可以采用精度匹配的方法，迅速地找到最小的误差区域，找到最符合匹配的区域，从而大幅度的降低匹配的次数，从而减少匹配运算量，减少匹配时间。具体可以这样实现，先不用逐点匹配，最开始先每隔几个像素点进行匹配计算，根据匹配模板的区域大小，来决定间隔像素点的数量。这样可以初步找到最终匹配区域的范围，然后再在这个范围内逐点进行计算比较，找到最好的匹配位置，这样的话可以大大减少不必要计算的次数，而且使得目标匹配的效果更好。
2. 精确计算的方法，由于我们只需要找到最小的误差区域的位置，而并不需要就算出每个点的精确误差值，所以我们可以改进计算方法，我们把目前已经计算的误差值，作为目前允许的最大匹配误差，后面每一个点的计算只需要跟目前的误差值进行比较，如果相对误差比目前的最大允许误差大，则直接可以结束计算，这个点就一定不是匹配点，进入下一个匹配位置的计算，如果相对误差小于最大允许误差，则这个点就会被当做目前为止最接近的匹配位置，则其得出的误差就会被当做最大允许误差，然后进行下一轮的比较。由于匹配的区域和非匹配区域的D(i,j)会相差好几个数量级，这种算法可以大大降低不必要计算的次数，从而提高算法的计算效率，提高匹配速度。
3. 随机匹配的方法，由于匹配区域在原图中的出现位置不确定，所以随机选取需匹配区域的算法，可以迅速地降低最小匹配误差，可以减少非匹配区域的计算量，从而降低算法的计算量。这种方法是基于上一种方法的改进，能减少很多的计算量。

### 2.3.2 OpenCV里的模板匹配方法

在OpenCV里，模板匹配的方法有两个自带的库函数，这两个函数分别是matchTemplate()和minmaxLoc()，这两个方法是OpenCV自带的模板匹配方法的核心函数。其中matchTemplate方法有四个参数，这四个参数分别为img，templ，result和method。img是模板匹配中的需要被搜索的原图，这里要求是8位或32位的浮点图。templ是template image的缩写，指的是模板匹配的模板图。result指的是模板匹配产生的结果矩阵，如果img使用的是w×h的原图，templ使用的是s×m的模板，则结果矩阵就是(w-s+1)×(h-m+1)。method指的是模板匹配的计算方法，这里有六种计算方法，这六种方法分别为：

（1） CV\_TM\_SQDIFF，这种方法的名称叫做平方差匹配方法。这种方法利用平方差进行匹配，最好的匹配结果为0，匹配效果越差，其匹配值越大，计算公式如下：

这里的R(x,y)表示的是匹配计算误差。

（2） CV\_TM\_SQDIFF\_NORMED，这种匹配方法叫做标准平方差匹配法，匹配结果和上一种的判别方法相同，其计算公式如下：

（3） CV\_TM\_CCORR，这种匹配方法叫做相关匹配法，这种方法采用匹配模板和被搜索区域的乘法计算，所以R(x,y)越大表示匹配的程度越高，R(x,y)越小则表示匹配的程度越低。其计算公式如下：

（4） CV\_TM\_CCORR\_NORMED，这种匹配方法叫做标准相关匹配法，由于这种方法采用的也是匹配模板和被搜索区域的乘法计算，所以其匹配结果评判标准跟上一种类似。其计算公式如下：

（5） CV\_TM\_CCOEFF，这种匹配方法叫相似匹配，这类方法将模板对其均值的相对值与图像的被识别区域的均值的相关值进行匹配，1表示完美匹配，-1表示糟糕的匹配，0表示没有任何相关性的随机序列。其计算公式为：

在这个式子里

（6） CV\_TM\_CCOEFF\_NORMED，这种方法被叫做标准的相关匹配，评判标准跟上一种方法相同，其计算公式为：

这六种方法从最简单的平方差计算到最复杂的测量相关系数，其运算的精准程度在一步一步的变得精细，所以得到的匹配结果也会越来越精确。但是伴随着这种精确的结果随之而来的是越来越高运算复杂度和运算代价，所以要结合自身实际选择在匹配精度和匹配花费时间之间做出取舍。

通过matchTemplate方法得到result矩阵之后，这时候就需要使用minmaxLoc()函数得出R(i,j)矩阵的最大值和最小值，从而得出匹配的最终结果。

### 2.3.3 级联分类器的方法

图像识别中物体的检测方法大抵可以分为两种，一种是基于已有目标的检测，另一种则是基于大量数据统计的检测。前者的检测方法的标准样例就是OpenCV中的模板匹配方法了，而后者基于大量数据统计的方法的代表就是训练级联分类器的方法了。这两种方法有其各自不同的长处和短处。模板匹配方法可能算法更简单，在匹配不是很复杂的情况下，匹配效果更精确，前期准备工作也会更简单。训练级联分类器的方法则是恰恰相反，它在基于大量数据的物体检测方面的检测速度更快，但这种速度优势则要付出一定的代价，使用者要在前期准备大量的检测样本，这样显然没有模板匹配方法方便。

在OpenCV中训练级联分类器有两种不同的实现，一种是haartraining，支持haar特征，另一种则是新的实现traincascade，支持LBP，HOG和boosted特征的级联分类器，这种新的实现具有较前一种来说更好的扩展性。要使用级联分类器，要经历两个不同的工作阶段，训练和检测。先是训练能检测出物体的级联分类器，然后用训练出的级联分类器检测物体。检测有OpenCV自带的检测程序，只需要导入自己训练出的级联分类器，也就是要导入级联分类器的.xml文件，然后导入需要检测的视频文件即可。所以级联分类器方法的运用的重点在于级联分类器的训练过程。

### 2.3.4 级联分类器的训练过程

前面说过OpenCV中自带有两个程序可以训练级联分类器，这两个程序分别是opencv\_harrtraining和opencv\_traincascade。其中opencv\_traincascade是一个opencv新推出的程序，这个程序是基于opencv2的API并且使用c++编写。这两者的主要区别是opencv\_harrtraining只支持harr一种特征，而后者却支持LBP(Local Binary Patterns)和harr两种特征。其中LBP是整数特征，而harr是浮点数特征。显而易见计算机在计算浮点数和整数时的计算速度是有很大的差别的，计算整数的速度要远远快于计算浮点数的速度。所以harr特征的训练和检测过程都会比LBP特征的速度要慢许多。但是由于harr是浮点特征，所以它的检测精度会比LBP特征的检测精度要出色许多，所以提高检测精度所付出的代价是检测时间。但是无论是LBP方法还是harr方法，检测的准确率在很大程度上是依赖于训练数据的选取质量和训练参数的。好的训练参数和训练质量无疑会换取更好的检测准确率和不必要的时间浪费。

训练级联分类器需要一系列的样本数据，这些样本数据分为两类，这两类数据分别为正样本数据和负样本数据。正样本数据是需要被检测物体的各式图像。负样本数据是指不包含被检测物体的需要被检测的背景图像。负样本数据需要自己准备，负样本图像可以是任意图像，但是负样本图像中不能包含被检测物体，用于处理得出负样本的图像文件名被放在一个纯文本文件之中，这些图像用于扣取负样本，所以图像的尺寸大小要大于训练窗口的大小，这样才可以顺利扣取负样本。正样本的数据则是由opencv自带的opencv\_createsamples方法生成。opencv\_createsamples方法能生成由opencv\_harrtraining程序和opencv\_traincascade程序支持的正样本数据，该方法会生成以.vec为文件拓展名的二进制图像文件。使用者只需要准备包含待检测物体的图像文件或者标记有待检测物体的图像文件，然后opencv\_createsamples方法会生成正样本数据。需要注意的是，如果待检测物体只具有一种待检测状态，则只需要一个正样本数据即可，但是如果待检测物体具有许多种变化的状态，则需要准备成百上千种正样本数据才可能准确的检测出待检测物体，这些正样本必须包含待检测物体的各个时间和空间下的状态。准备好正样本和负样本，级联分类器的训练工作就完成了很大一部分，剩下的工作会很简单。训练好的级联分类器会由训练程序生成并存储于cascade.xml文件中，然后我们需要做的只需要在opencv自带的检测程序中导入cascade.xml文件即可。

## 2.4 本章小结

本章介绍了android客户端的开发环境androidstudio相比于Eclipse开发环境的新特性和新优势，然后介绍了服务端开发环境Eclipse的特点。也介绍了Java socket编程技术和Java多线程技术，最后介绍了opencv的基本架构和图像处理的两种基本方法，这两种方法分别是级联分类器的方法和模板匹配的方法。本章的技术简介为后面的内容打下了铺垫。

# 3 系统分析

## 3.1 可行性分析

### 3.1.1 经济可行性

随着智能手机的普及化，一台Android设备的价格也在持续下降，一些国产手机厂家生产的Android设备的价格也变得十分的便宜。由于本系统需要利用Android手机的视屏传感器和gpu图像处理的功能，所以大规模的部署需要一大批的Android手机设备，目前一些Android手机的价格甚至低于很多带wifi传输功能的视频传感器价格。更别谈跟那些带有图像处理功能的摄像头相比了，部署了这个系统之后唯一需要后期投入的就是后期维护和手机充电所需的费用了，又由于Android的设备使用寿命至少能维持三年以上，手机充电所需的费用又是微乎其微的，由于手机通过wifi进行数据连接，所以数据流量的费用又可以忽略不计，所以整个系统经济上是完全可行的。

### 3.1.2 技术可行性

本系统客户端部署在Android系统上，客户端上的图像处理部分使用的是OpenCV里自带的模板匹配的方法，OpenCV图形处理库由于其出色的可移植性和开源特性而倍受广大程序员们的欢迎，opencv在Android平台上也有其移植过后的版本，本系统在Android客户端上用android版本的opencv图形处理库完成图像处理，并在Android客户端上使用Java socket编程与服务端建立数据连接，并向服务端传回客户端得到的处理后的图像的数据坐标点，然后经过服务端的Java程序处理之后存入数据库，本系统的所有技术包括图像处理，网络编程，数据库操作都是技术可行的，其中Android系统的opencv图像处理操作虽然目前可以查阅的资料很少，但是由于opencv出色的开源特性，在其开源社区也可以寻求许多技术帮助，所以整体系统在技术上是完全可行的。

### 3.1.3 运行可行性

本系统对Android设备的整体要求不高，由于每个Android设备都肯定有cpu，cpu中自带的gpu模块肯定能完成图像处理的要求，只是存在性能上的差距，也就是处理速度上的差距。wifi是每个Android设备的必备功能之一，每个Android设备必须带有wifi功能，摄像功能也是现代手机所必须具备的功能，所以具有了图像处理功能，wifi网络连接功能和摄像功能的Android设备是每一个具有销售资格的Android设备必须带有的功能，这也正是本系统需要的基本功能，所以本系统在运行方面是完全可行的。

## 3.2 需求分析

### 3.2.1 功能性需求

本系统需要利用手机摄像头拍摄到的图片或者是视频流检测出无人机在图像中的像素点的位置，并且把这些信息传输到服务端的数据处理程序里去，并且由服务端的数据处理程序对数据进行处理并且得出无人机的坐标点位置。所以这个系统包含了调用手机摄像头的功能，并且具有利用wifi局域网络传送数据的功能，又因为要检测无人机必须利用多个客户端才能准确检测出无人机在坐标系的准确位置，所以服务端必须具有多线程处理的功能，服务端必须唤醒多个线程同时监听多个端口号，接收每一个客户端传回的无人机数据。服务端又需要把数据存储入数据库中，所以服务端具有数据库增删改查的功能。

### 3.2.2 非功能性需求

（1） 兼容性

本系统的客户端部分选取的Android API版本是Level 19，所以客户端版本只支持Android4.0以上版本的手机，由于Android4.0版本早已成为过去式，现在几乎所有的Android手机的Android版本都在4.0以上，所以客户端部分是完全兼容所有的Android设备的，服务端的程序是用Java编写的，Java的兼容性应该不存在问题，所以本系统具有良好的兼容性。  
 （2） 可移植性

本系统的核心代码部分利用了opencv中的模板匹配技术，虽然本系统只有Android的客户端，但是opencv具有出色的移植性，可以完美的移植到Windows、Linux平台，核心代码具有出色的移植性，其他部分代码的移植就不存在太大的问题，所以整体系统具有可移植的特性。

## 3.3 本章小结

本章对本系统进行了可行性分析和需求分析，先是对本系统的经济可行性进行了分析，分析结果得出本系统在经济方面是完全在可控范围之内的。然后分别对系统进行了技术可行性和运行可行性的分析，结果表明本系统在技术和运行方面都是可行的。最后对本系统进行了需求分析，分别分析了本系统的功能性需求和非功能性需求，为系统的设计打下了基础。

# 4 系统设计

## 4.1 概要设计

### 4.1.1 系统项目背景

本节内容主要介绍了本系统的有关项目背景，项目软硬件构成的基本内容，以及用到的有关技术的基本内容。

1. 系统的设计目的

本系统的设计目的是为了实现基于视频信息的无人机检测，无人机在新世纪的今天已经是一个越来越热门的话题，各种无人机产品走进平民百姓家。无人机的价格不断走低，让更多的人体验到了无人机带来的乐趣与便利，也同时让许多的高空作业变得更加安全。然而无人机犹如一把双刃剑，既有其有利之处，也有其弊端。无人机的过度泛滥给人民群众的隐私带来了极大的威胁，也给航空航天业这种需要极度安全保障的工业带来了极大的安全挑战，所以这时候一个能基于视频信息的无人机检测系统就成了十分迫切的需求，这样一个系统能部署在人群密集的居民区，也可以部署在民航机场这样的场所维护人民的隐私和生命安全。

1. 系统运行环境

本系统的客户端设计运行在Android手机平台上，服务器端设计运行在Windows

平台上。

1. 系统实施环境

硬件：本系统是实时检测并且定位无人机的，所以对系统的传感器和系统处理流程以及软硬件衔接有着比较高的要求，图4.1是基于视频信息的无人机检测系统的硬件结构，包括手机摄像头网络，图像处理系统，数据传输系统，数据处理系统，可视化显示定位结果。



图4.1 系统硬件结构图

系统的工作流程是一个大坐标系内的手机摄像头网络先采集拍摄到的图像，然后图像处理程序检测图像，如果图像中没有无人机则返回零，如果图像中有无人机存在则返回无人机在图像中的坐标点位置，手机客户端将图像处理得到的数据通过数据传输系统传给pc端的服务端，服务端对数据进行处理，得到无人机在此大坐标系的坐标点，并且通过可视化工具可视化显示定位结果。

软件：由硬件得到采集到的图像，运行我们的图像处理算法，得到无人机在图像中的图像坐标点，再由网络传输程序把这个数据点传送至服务端，由服务端的定位程序得到定位到的坐标，并由可视化程序显示出来，图4.2是该系统的软件结构图



图4.2 基于视频信息的无人机检测系统的软件结构图

通过软件环境和硬件环境的相互衔接，系统能进行采集图形，得到像素坐标，并最终得到无人机坐标，最后显示出来。通过完整的软件和硬件环境，系统才能完美运行，其中软件和硬件结构是相对而言的，系统的实施就是在这样的软硬件环境下实施的。

### 4.1.2 系统总体设计

基于视频信息的无人机定位是近年来新出现的问题，无人机是近几年开始普及的，opencv也在近几年才开始蓬勃发展。而我们设计的基于视频信息的无人机定位系统即能做到基于视频信息的无人机的检测，而且能更进一步达到能在一个坐标系的范围内定位无人机的坐标位置。前面我们说过整体的系统处理流程，我们首先在一个固定的坐标系内随意部署几部Android设备，具体几部设备能做到精确定位，我们在后面的仿真测试章节会讨论。我们利用这几部手机设备拍摄到的图片，利用Android客户端的图像处理程序检测无人机是否在图片中，如果图片中没有无人机，则返回零，有无人机则返回无人机在图片中的像素点坐标。

客户端将这些数据传输回服务端，现在服务端怎么根据这些像素点坐标得到具体无人机坐标呢。这里我们使用了建表查表的方法，在进行检测之前。我们先使用一个用于建表的无人机，在坐标系下飞行。建表的无人机被系统部署好的手机设备所拍摄到的图像得到的数据传回服务端并且存回数据库，同时存入的还有无人机的gps数据。这里要求每一个时间点所有传感器采集的一帧经过图像处理后得到的数据对应这一时刻的无人机的gps数据，并且把这些数据存入数据库作为数据表的一行。但是要实现每个传感器的数据在时间上对齐并不是一个简单的任务，由于手机摄像头的帧率基本都在30fps以上，假设我们的每个摄像头的帧率都是30fps，这样每秒每个摄像头都要传回30个数据点，这样每帧的时间间隔只有大约3ms左右。由于各个传感器的系统时间不可能对齐到毫秒级，所以我们使用时间戳的方法来对齐每一帧的方法也是不太现实的。我们必须利用其它的方法来对齐每一帧的数据。这里我们可以想到的两种方法：

1. 借助外部事件

这种方法是借助外部做出能被我们的手机传感器所捕捉到的特征动作，例如是一个大的响声，或者是一道亮光。如果是一声大的声响，这样就会被我们坐标系下所有的手机的麦克风捕捉到，由于我们选取的坐标系范围不是很大，所以这个声音传播到所有传感器的时间可以忽略不计，这样我们可以以这个被麦克风所采集到的声音作为我们的特征，所有采集到这个响声的起始时刻作为我们的对齐时刻，每个传感器采集到的数据从这一时刻开始对齐。

1. 无人机做出特征动作

这种方法是让无人机做出一个特征动作，例如一个直角转弯或者是一次突然加速。假如是一个直角转弯的话，我们所有拍到这个动作的传感器所产生的数据坐标集就会有一个明显的特征，我们可以通过一系列的算法得到这个直角拐弯所在的具体时刻，这个时刻就会作为我们的对齐时刻。突然的加速也是同样的道理，无人机做出突然加速的动作，所有拍到这段动作的传感器所产生的数据坐标集也会有一个明显的特征，可能是两个坐标之间的间隔可能会突然增大，这样的话我们会根据这个特征所在的时刻作为我们的对齐时刻，每个传感器采集到的数据从这一时刻开始对齐。

无人机经过特殊的飞行轨迹，建成数据表，接下来进入检测阶段，服务端发出命令，传感器开始捕获图像，图像经过处理之后得到的数据传回服务端，服务端用采集到的数据与数据表中的数据进行比对，选取差距最小那一行的gps数据作为定位得到的无人机定位坐标。

## 4.2 详细设计

### 4.2.1 检测系统流程图

整个系统的大致流程如图4.3所示，该流程图是基于实际开发时的系统，其中建表查表的过程我们在后面的仿真实验中实现。



图4.3 系统总体流程图

### 4.2.2 客户端图像处理部分的设计

客户端的图像处理部分我们使用了模板匹配算法，模板匹配的主要特点我们已经在第二章进行了详细的介绍。这里由于模板匹配方法的计算量很大，所以其识别效果不甚理想，所以我们又在pc端实现了基于Adaboost的级联分类器的方法来对图像进行识别。Adaboost的级联分类器方法的识别速度较模板匹配方法的识别速度有一个质的提高。训练级联分类器的部分我们在第五章有了大篇幅的介绍，这里我们主要介绍一下级联分类器方法检测目标物体的过程，整个过程的流程如图4.4所示



图4.4 级联分类器检测过程

我们训练得到的级联分类器包含有20级的强分类器，如果我们需要检测的目标图像通过全部20级的强分类器才会判断检测到目标物体，如果没有通过任意一级的强分类器的检测就会判断为没有检测到目标物体，以上是Adaboost级联分类器的大致检测过程。

### 4.2.3 网络通信部分的设计

网络通信部分我们应用到了Java多线程的内容，还应用到了Java socket通信的内容，我们在客户端和服务端都应用到了Java多线程部分的内容，在客户端中我们在主线程中新建一个线程来与服务端进行通信，服务端新建多个线程来监听不同的端口，接收客户端发送过来的数据信息。图4.5给出了网络通信部分的流程图，这里首先服务端先新建serversocket然后绑定端口号，然后服务端监听端口，直至收到客户端的连接请求。连接建立之后客户端服务端就能相互接收发送数据了。

这里客户端单向将检测信息传送给服务器，服务器端对应每一个客户端都有一个接收队列，每个队列对应接收一个客户端中的信息，接收队列为主类中的公共属性，每个线程接收到一条检测信息，都会调用主类中的对应队列进行一次入队操作，主函数在检测过程中会将队列中的数据进行出队操作，然后进行查表操作，得到对应坐标点的位置，网络通信模块需要打开Android的wifi权限，同样在Android客户端也需要使用多线程的方法，在检测的过程中，启用一个网络通信线程，将检测得到的信息通过网络通信类发送给服务器。若不使用多线程的方法，由于检测部分和数据发送部分不是同步操作，所以应用程序就会崩溃，所以这里在服务器端和客户端都要使用多线程的方法。



图4.5 网络通信部分流程图

### 4.2.4 检测部分的设计

检测部分的仿真实验我们需要应用到针孔成像模型。而在实体设计部分，我们事先需要无人机在坐标系的指定高度按预定路线飞行，坐标系下的摄像头捕获到图像并处理后得到的数据传送给服务器，服务器进行建表操作，完成建表操作之后我们需要进行无人机的检测了，无人机的检测跟建表过程大同小异，前面的部分基本相同，不同之处是我们不需要将无人机的gps坐标信息传给服务端，只需要把坐标系下手机检测到的数据传回给服务端，服务端只需要将客户端传回的数据进行查表操作，查表操作的流程图如图4.6所示，建表过程中每个手机检测传感器将检测得到的数据存成结构体传给服务端，这个结构体中有三个数据，第一个数据为标记是否检测到无人机目标，如果检测到则第一个数据置1，第二个和第三个数据分别存入检测到的无人机在图像中的x像素坐标和y坐标。这里如果图像中没有检测到无人机我们就置x和y均为-1，并置第一个数据为0。初始建表的表结构如表4.1所示。在表4.1中每一列代表着摄像头标号，每一行代表着每一帧，这样每行与每列相交的地方代表这列标号的摄像机在这一帧下采集的数据，前面说过采集到的数据是一个多元的数据，包含有三个属性。

表4.1 服务器端的表结构

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | spot\_1 | spot\_2 | spot\_3 | spot\_4 | spot\_5 | spot\_6 | … | Loc |
| ID | A1 | A2 | A3 | A4 | A5 | A6 | … | L1 |
| ID | B1 | B2 | B3 | B4 | B5 | B6 | … | L2 |
| … | … | … | … | … | … | … | … | … |



图4.6 查表定位流程图

## 4.3 本章小结

本章讲述了本系统的详细设计和概要设计，先简要的介绍了本系统的总体设计，然后详细地介绍了本系统的重要部分的功能流程和主要设计思想，为后面的总体实现打下了坚实的基础。

5 系统实现

## 5.1 图像处理的级联分类器方法的实现

本系统在Android客户端的图像处理系统中只实现了模板匹配的方法，这里的级联分类器方法是在pc端实现的。由于模板匹配方法的运算量太大，匹配速度不甚理想，所以这里实现级联分类器的方法作为一个补充。

### 5.1.1 级联分类器的训练

级联分类器是基于Adaboost算法的，而Adaboost算法是针对boosting算法做出的改进算法，图5.1是Adaboost算法的流程图。要训练一个级联分类器，首先要准备一个正样本集还有一个负样本集。正样本集需要包含的是你要识别的特征的图像，图5.2是



图5.1 Adaboost算法的流程图

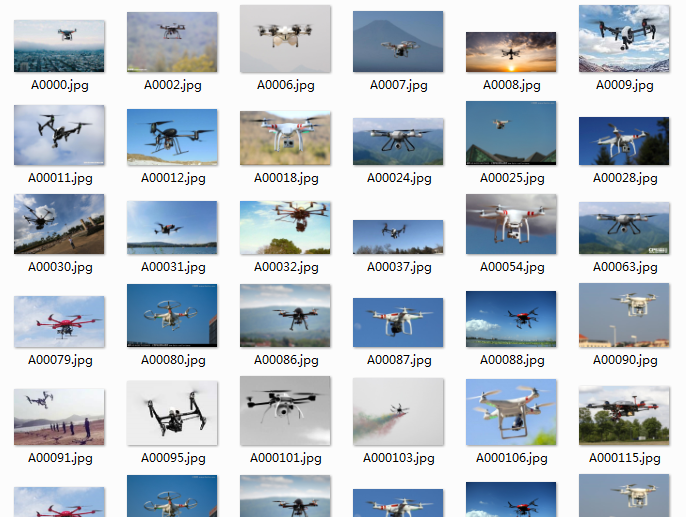


图5.2 正样本集

我们训练能检测无人机的级联分类器的正样本集的截图，由于我们需要的只是包含无人机的那一块的像素点，所以我们必须截取这一块无人机所在的像素区域，这里我们用到了opencv自带的抠图程序opencv\_createsamples程序，opencv\_createsamples程序在使用之前需要配置参数，以下是opencv\_createsamples程序需要配置的参数及参数说明：

（1） <-info> 这里指的是存放正样本数据目录位置相同的描述文件路径，可用txt、dat等数据格式表示，每一行的内容如：sample.jpg nums left\_x left\_y width height。例如我们设置参数为1.jpg 1 0 0 20 20就表示在1.jpg的正样本中存在。

（2） <-img> 如果需要把一张图片扭曲变形成多张图片，就需要填写这个参数，否则不需要填，填写这个参数前面的info参数的就会无效。

（3） <-vec> 用于生成正样本数据的文件路径，内容应该填写为文件路径的格式。

（4） <-bg> 这里填写跟存放负样本数据位置相同的描述文件的路径，这里的描述文件可以是txt或者是dat格式的文件，每一行的内容为负样本文件的文件名，后缀为jpg格式的。

（5） <-maxidev> 这个参数是一个用于生成前景灰度值的参数，实际样本的灰度值会根据这个参数然后结合随机数产生不同的灰度值。

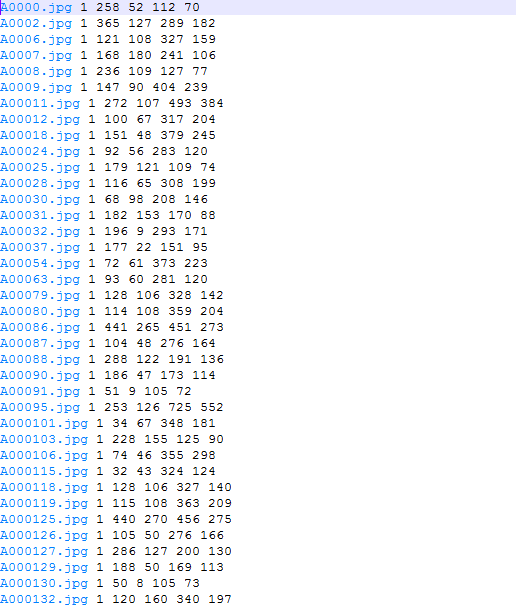


图5.3 pos.dat的内容

（6） <-maxxangle> 这个参数是对样本图片的x轴方向扭曲的最大弧度，即图片水平方向的旋转弧度。

（7） <-maxyangle> 这个参数是对样本图片的y轴方向的扭曲的最大弧度，即图片垂直方向上的旋转弧度。

（8） <-maxzangle> 这个参数是对样本图片的z轴方向的扭曲的最大弧度，即图片垂直于水平方向上的旋转弧度。

（9） <-show> 这个参数可以选择在样本创建期间是否需要通过imgshow显示出每一个生成的样本图片出来。

（10） <-w> 这个参数用于创建样本图片的宽度。

（11） <-h> 这个参数用于创建样本图片的高度。

这里我们有541个正样本数据，设置-w的参数为20，-h的参数为20，新建一个createsamples的批处理文件，代码如下:

opencv\_createsamples.exe -info pos\pos.dat -vec pos\pos.vec -num 541 -w 20 -h 20

pause

得到pos.vec文件，代表着我们的正样本数据的处理工作完成。得到pos.vec文件之后，我们正式开始进行级联分类器的训练工作。OpenCV自带两个级联分类器的训练程序，分别是opencv\_haartraining和opencv\_traincascade其中opencv\_haartraining只能训练基于haar特征的级联分类器而且训练效果明显比opencv\_traincascade的效果差，所以我们选择opencv\_traincascade方法来训练我们的级联分类器。opencv\_traincascade方法的参数配置如下：

（1） <-data> 这里需要给出训练得到的级联分类器的存储目录。

（2） <-vec> 根据opencv\_createsamples方法得到的vec后缀的正样本文件文件。

（3） <-bg> 负样本说明文件，主要包含了负样本文件的包含目录以及负样本的文件名。

（4） <-numpos> 每级训练所用到的正样本数，应选择总正样本数的90%为佳。

（5） <-numNeg> 每级分类器所用到的负样本的数目，可以大于-bg文件指定的负样本数目。

（6） <-numStages> 训练的级联分类器的级数，也就是强分类器的个数。这里我们选择训练20个强分类器。

我们的训练参数如下：

opencv\_traincascade.exe -data data -vec pos\pos.vec -bg neg.dat -numPos 500 -numNeg 1696 -numStages 20 -w 20 -h 20 -minHitRate 0.9999 -maxFalseAlarmRate 0.5 -mode ALL

pause

这里我们设置正样本集的个数为500，由于正样本集的数量必须要小于我们采集的图像的数量，这里我们设置正样本集的数量为我们采集样本数的四分之三，截取图像的大小都是20，训练得到级联分类器。

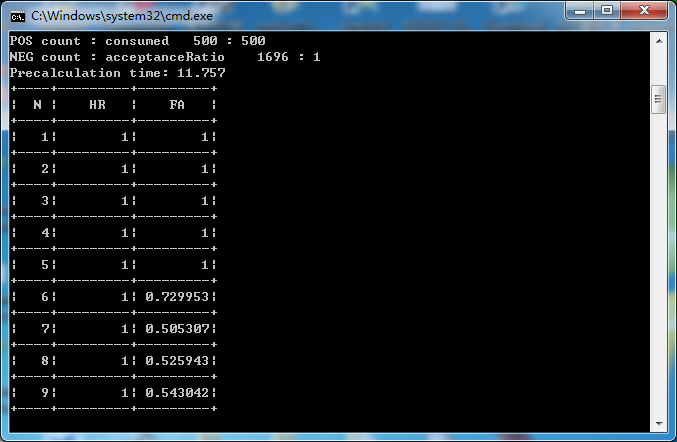


图5.4 级联分类器的训练过程截图

Haar特征的级联分类器训练过程所需时间较LBP特征的训练过程所需时间要稍久一些，这里的Haar特征的训练过程需要大约30分钟。得到了图5.5所示的级联分类器和20个强分类器。

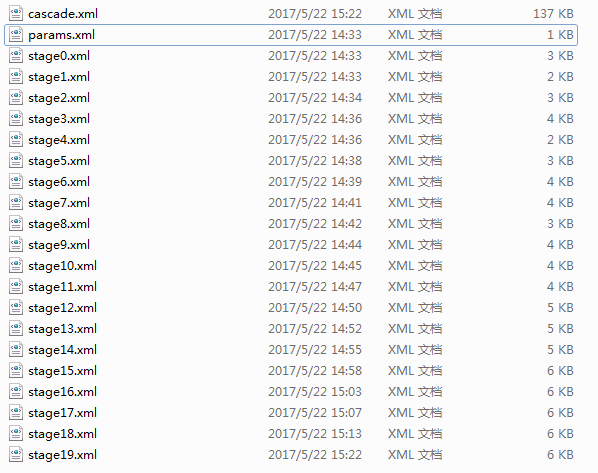


图5.5 级联分类器和强分类器文件

### 5.1.2 目标检测

级联分类器检测目标的流程图如图5.6所示，检测的过程我们采用c++实现，首先我们先定义一个cascadeclassifier对象，然后导入我们自己训练出来的cascade.xml文件，这部分的代码如下：

string Cascade\_mode = "cascade.xml"; //已训练好的分类器

CascadeClassifier Mycascade;

if (!Mycascade.load(Cascade\_mode)) { printf("[error] 无法加载级联分类器 文件！\n"); return -1; }

导入了训练得到的xml文件我们就可以进行目标的检测了。检测目标的过程大致是，先导入需要检测的目标图像，然后将图像转化成灰度图像，然后检测并用绿色的矩形方框标定出目标在图像中的位置。检测并且显示部分的代码如下：

Mat detectAndDisplay(Mat frame)

{

std::vector<Rect> pedestrain;

Mat frame\_gray(frame.size(), CV\_8U);

cvtColor(frame, frame\_gray, CV\_BGR2GRAY);

equalizeHist(frame\_gray, frame\_gray);

Mycascade.detectMultiScale(frame\_gray, pedestrain, 1.1, 2, 0, Size(64, 128));

for (int i = 0; i < pedestrain.size(); i++) {

rectangle(frame,

pedestrain[i],

Scalar(0, 255, 0),

1);

}

imshow(window\_name, frame);

return frame;

}

检测的效果如图5.6所示，无人机所在的像素区域都用粉色圆圈标注了出来，这个视频中的无人机在天上进行巡航操作，无人机的姿态由远离摄像头到靠近摄像头，我们训练得到的级联分类器都能准确的检测到视频中的无人机。



图5.6 检测效果图

## 5.2 图像处理的模板匹配方法的实现

这部分是在Android客户端实现的，这部分工作的流程是首先调用手机摄像头捕捉到检测目标的模板，然后调用手机摄像头捕捉到的每一帧图像都会跟这个模板相匹配，这里我们首先设置摄像头的参数为调用前置摄像头，具体代码部分如下：

<org.opencv.android.JavaCameraView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:visibility="gone"

android:id="@+id/HellpOpenCvView"

opencv:show\_fps="true"

opencv:camera\_id="front" />

这里我们设置此空间的可视性为隐藏，然后设置show\_fps为true这样将会在屏幕的左上方显示图像处理的fps参数，最后设置camera\_id参数为front。这里camera\_id有两个参数，分别为front和back，这里我们选择front参数，表示我们将使用前置摄像头来捕捉图像。

第二步我们会在手机摄像头的中心区域来捕捉我们要识别的对象模板，这部分的代码实现如下：

public void onCameraViewStarted(int width, int height) {

//Initialize Mats which will be used in onCameraFrame

toBeDetectedMat = new Mat();

nullMat = new Mat();

cameraStartTime = System.currentTimeMillis();

//Capture area is 200\*200

captureRectPt1 = new Point(width/2 - 100, height/2 - 100);

captureRectPt2 = new Point(width/2 + 100, height/2 + 100);

captureRect = new Rect(captureRectPt1, captureRectPt2);

resultMatRows = height - 200 + 1;

resultMatCols = width - 200 + 1;

Log.i(TAG, "onCameraViewStarted:" + width + "\*" + height);

}

这里传入的参数会根据手机屏幕的像素面积得出，weight和height的坐标就是手机屏幕的像素中心点，然后这个方法会初始化相机，并且标示出我们要检测区域的位置，然后就是检测部分了。检测部分的代码如下：

public Mat onCameraFrame(CvCameraViewFrame inputFrame) {

mRgba = inputFrame.rgba();

if (System.currentTimeMillis() - cameraStartTime < 5000) {

Core.rectangle(mRgba, captureRectPt1, captureRectPt2, RECT\_COLOR\_RED, 3);

} else {

if (!srcCaptured) {

mRgba.submat(captureRect).copyTo(toBeDetectedMat);

Mat toBeDetectedMatRGBA = new Mat();

Imgproc.cvtColor(toBeDetectedMat, toBeDetectedMatRGBA, Imgproc.COLOR\_BGRA2RGBA);

File path = Environment

.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES);

String filename = "captured.png";

File file = new File(path, filename);

filename = file.toString();

Highgui.imwrite(filename, toBeDetectedMatRGBA);

detectResult = new Mat(resultMatRows, resultMatCols, CvType.CV\_32FC1);

srcCaptured = true;

} else {

detectResult.create(resultMatRows, resultMatCols, CvType.CV\_32FC1);

int matchMethod = Imgproc.TM\_SQDIFF\_NORMED;

Imgproc.matchTemplate(mRgba, toBeDetectedMat, detectResult,

matchMethod);

Core.normalize(detectResult, detectResult, 0, 1, Core.NORM\_MINMAX, -1, nullMat);

Point matchLoc;

MinMaxLocResult minMaxLocResult = Core.minMaxLoc(detectResult, nullMat)

if (matchMethod == Imgproc.TM\_SQDIFF

|| matchMethod == Imgproc.TM\_SQDIFF\_NORMED) {

matchLoc = minMaxLocResult.minLoc;

Log.i(TAG, "detected min value:" + minMaxLocResult.minVal);

} else {

matchLoc = minMaxLocResult.maxLoc;

}

// Start draw rectangle

Point pt2 = new Point(matchLoc.x + toBeDetectedMat.cols(),

matchLoc.y + toBeDetectedMat.rows());  
这个方法中我们首先启动相机，然后把需要检测的目标放在模板采集的红色框中放置5s,然后把采集到的模板信息存储到toBeDetectedMat这个矩阵中，5s之后相机中心区域的红色方框就变成了绿色方框，绿色方框标示出待检测区域中的需检测目标所在位置，这里我们采取模板匹配算法中的TM\_SQDIFF\_NORMED计算方法，计算得到的结果存储在matchLoc这个点中，然后根据这个点画出绿色的检测结果方框。

## 5.3 客户端服务端数据传输的实现

客户端的数据传输部分和服务端的数据传输部分使用的方法大致相同，客户端和服务端都重写了Runnable接口中的run方法，都是新建一个线程来处理客户端和服务器的通信，客户端的主要代码如下：

public void run() {

s = new Socket();

try {

s.connect(new InetSocketAddress("192.168.1.101", 3000), 5000);

os = s.getOutputStream();

os.write((msg.obj.toString() + "\r\n").getBytes("gbk"));

这里是clientThread类的主要方法，将客户端的数据发送给服务器，然后主函数调用clientThread类中的方法，实现将实时得到的坐标点数据传送给服务端，这部分的代码如下：

double x1= (matchLoc.x+pt2.x)/2;

double y1= (matchLoc.y+pt2.y)/2;

try {

Message msg = new Message();

msg.what = 0x345;

String str = x1 + "," + y1;

Log.e(TAG, str );

msg.obj = str;

clientThread.revHandler.sendMessage(msg);

}

catch (Exception e)

{}

这里的x1，y1指的是检测到目标方框的中心点的像素坐标。然后服务端只需要创建一个监听指定端口线程即可。

## 5.4 本章小结

本章主要实现了级联分类器方法的训练过程和检测过程，然后实现了Android客户端的模板匹配方法。最后实现了将客户端的检测数据传送服务端的过程，至此我们已经实现了本系统中的大多数内容，定位无人机位置的算法将在下一章的仿真实验中具体分析。

# 6 仿真结果及实验

## 6.1 仿真算法

这里我们先假设我们事先知道无人机的高度坐标，然后建还包含有表。得到一张包含每个传感器检测到的无人机数据的表，又由于建表时无人机的坐标信息可以通过建表用无人机的坐标信息得到，所以这张表中还有无人机的坐标信息。每个传感器的信息可以由无人机的信息得到，这里我们可以通过针孔成像模型由无人机的坐标信息计算得到每个传感器中无人机的像素点位置，这里就仿真了建表过程。

这里涉及到三个坐标系，这三个坐标系分别是世界坐标系、摄像机坐标系、和图像坐标系。这里的世界坐标系我们指的是无人机相对于摄像机的坐标系位置，所以摄像机的位置就是原点位置了，图像坐标系是指摄像机投影到手机上的像素点的坐标。我们可以根据公式6.1由已知的无人机坐标和摄像机坐标得到图像坐标系的坐标。  
M

这里的ax和ay还有u0和v0都是摄像机的内参，Zc等于Zw。Xw和Yw都是世界坐标系的坐标，u和v代表图像坐标系的坐标。这样我们就可以根据世界坐标系的坐标得到图像坐标系的坐标，以下是python代码的转换函数的代码实现：

def get\_pos\_one(x,y):

# R=mat([[1,0,0],[0,1,0],[0,0,1]]);

# T=mat([[xc],[yc],[zc]]);

M=mat([[1,0,0,xc],[0,1,0,yc],[0,0,1,zc],[0,0,0,1]]);

N=mat([[ax,0,u0,0],[0,ay,v0,0],[0,0,1,0]]);

XW=mat([[x],[y],[zw],[1]]);

A=N\*M\*XW;

# print A;

a1=sum(A[0,:]);

a2=sum(A[1,:]);

return a1/zw,a2/zw;

得到的a1/zw和a2/zw分别代表图像坐标系的横纵坐标，以下部分代码是相机内参的定义：

f=12\*10e-4; #焦距12mm

Sx=1024; #图像分辨率1024\*768

Sy=768;

ax=f\*Sx;

ay=f\*Sy;

u0=Sx/2;

v0=Sy/2;

这里我们定义相机传感器的图像分辨率是1024\*768，然后手机摄像头的焦距定义为12mm，这样我们得到了坐标转换的函数。

得到坐标转换函数之后我们就要开始建表过程，这里我们先定义总坐标系中的摄像头的坐标位置，这里我们先定义8个摄像头的情况。8个摄像头的摄像角度经测量都为90度，又由于这里我们假定无人机的高度为40m，则摄像头的可视区域为一个圆锥体，这个圆锥体的底面为一个半径为40m的圆，图6.1为8个摄像头在40米高度覆盖的范围视图：

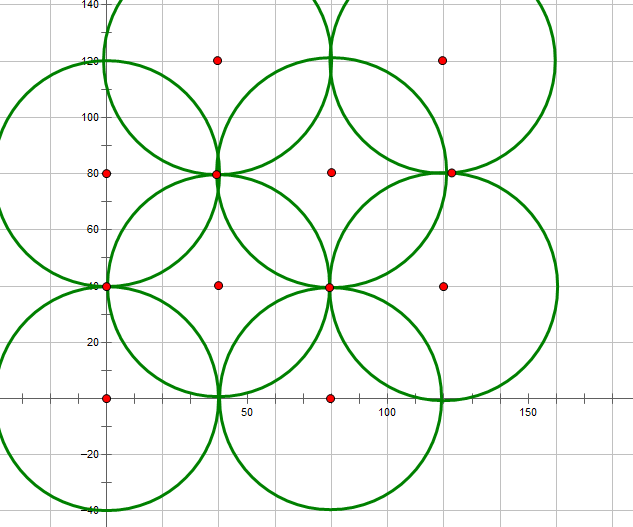


图6.1 模拟无人机在坐标系中的位置

这里我们定义总坐标系的范围为120\*120，这样如图所示，8个摄像头能完全覆盖整个坐标系。我们在120\*120的坐标系中随机生成5000个坐标点，然后定义一个结构体，此结构体含有三个参数，分别是i，x，y。首先判断坐标点是否在这8个摄像头的监视范围之内，例如若坐标点L1在摄像头C1的监视范围之内则定义L1和C1对应的结构体的i为1，并利用针孔成像的函数算出此坐标点在摄像头C1中的图像坐标点x，y并传递给结构体。如果没有在监控范围内检测到坐标点L1，则i定义为0，x，y都定义为-1。分别在每个摄像头的坐标下计算出结构体的参数，然后把结构体存入表中，这样我们可以得到一张5000行9列的数据表。5000行代表5000个坐标点，前8列代表八个摄像头，第九列存入坐标点的总坐标系的坐标。具体实现代码如下：

spot4=array([[0,0],[0,80],[40,120],[40,40],[80,0],[80,80],[120,40],[120,120]]);

class item:

def \_init\_(self):

self.i=0;

self.x=0;

self.y=0;

def get\_pos(x,y):

for i in range(0,8):

spot\_x=spot4[i][0];

spot\_y=spot4[i][1];

if(sqrt((x-spot\_x)\*(x-spot\_x)+(y-spot\_y)\*(y-spot\_y))<=40):

m1,m2=get\_pos\_one(spot\_x-x,spot\_y-y);

a=item();

a.i=1;

a.x=m2;

a.y=m1;

spotlist[i].append(a);

else:

a=item();

a.i=0;

a.x=-1;

a.y=-1;

spotlist[i].append(a);

a1=item2();

a1.x=x;

a1.y=y;

spotlist[8].append(a1);

这里的spot4存放的是摄像头在总坐标系的坐标，这里我们用坐标点的坐标减去摄像头的坐标就可以得到坐标点相对于摄像头的坐标spotlist是一个二维数组，存放的是建表得到的表数据。

表建好之后我们就需要进行查表操作，查表的过程我们已经知道了检测目标在每个摄像头位置的检测信息，然后查表得到检测目标在总坐标系下的坐标，这里我们先随机生成100个坐标点，然后根据get\_pos\_two函数得到100个坐标点对应的摄像头的检测信息，然后根据得到的检测信息进行查表操作，然后利用查表得到坐标点与随机生成的坐标点相比较，计算出建表查表法的误差。以下是查表部分的函数：

for i in range(0,50000):

flag=1;

for j in range(0,24):

if(b[j].i!=spotlist[j][i].i):

flag=0;

if(flag):#当a.i相同时才进行a.x和a.y的比较

sum=0;

for j in range(0,24):

sum+=(spotlist[j][i].x-b[j].x)\*(spotlist[j][i].x-b[j].x)+(spotlist[j][i].y-b[j].y)\*(spotlist[j][i].y-b[j].y)

if(min>sqrt(sum)):

min=sqrt(sum);

x=spotlist[24][i].x;

y=spotlist[24][i].y;

先比较a.i是否相同，如果a.i相同再在a.i相同的数据点中选择a.x和a.y的差值最小的那一行数据对应的坐标点作为检测结果。

表5.1 误差对照表

|  |  |  |
| --- | --- | --- |
|  | 随机生成的坐标点 | 查表得到的坐标点 |
| 检测点1 175.898，113.123 75.985，113.685 | | |
| 检测点2 | 98.715，61.946 | 98.784，60.381 |
| 检测点3 | 80.28，61.985 | 80.887，62.271 |
| 检测点4 | 45.395，3.444 | 46.539，4.229 |
| …… | …… | …… |
| 检测点100 | 80.887，54.939 | 81.492，54.364 |

这里我们得到了误差对照表，计算得到误差值为0.0288145585097。经过多次实验，结果显示示在系统使用8个检测摄像头。

且表内数据充足的情况下建表查表法的误差基本在5%以下，且平均误差在2%左右，建表查表法的定位效果较为理想。

## 6.2 误差分析

### 6.2.1 建表数据量对误差的影响

这里我们设定摄像头的个数为8的情况下，探索初始建表的数据量对查表误差的影响。描点做出的折线图如图6.2所示，

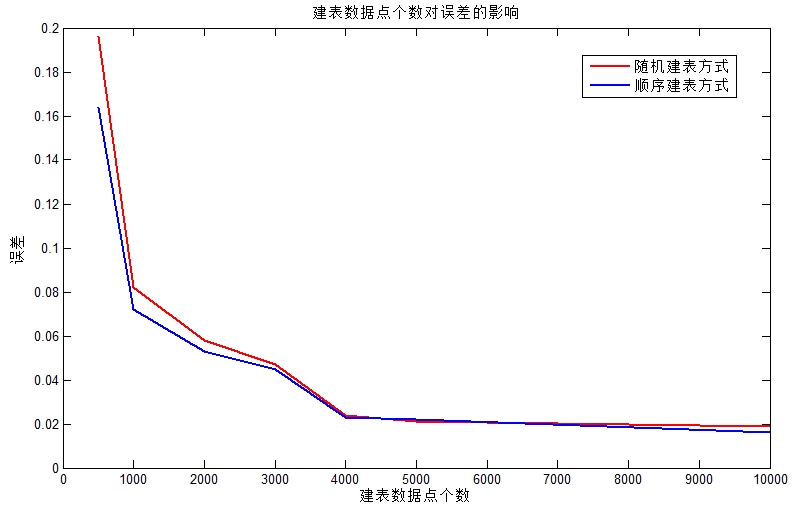


图6.2 误差折线图

由图像特征我们可以观察到，数据个数在由500个增加到1000个时误差出现了一次大幅度的减小，由0.19左右减小到0.08左右，数据个数由1000增加到5000时，误差的减小幅度却没有数据个数由500增加到1000时那么明显，但还是在小幅度下降。当数据量从5000增加到10000时，误差值却基本变化不大，表明数据的个数在5000的时候已经基本饱和，再增加数据个数对系统的检测精度的提升不大，同样也可以从图中得出，在数据点个数较小时顺序建表方式的误差要优于随机建表方式，但随着数据规模的扩大，两种方式得到的误差基本一致。

### 6.2.2 坐标系下摄像头数量对误差的影响

由于摄像头的个数不同，建库时所需的数据点个数也不同，所以我们这里相同数据规模下的误差来比较我们的检测精度，这里我们选择摄像头设备的个数分别为4个、6个、8个、10个、12个和24个，选取8个摄像头情况下数据规模为1000由于我们增加了摄像头设备的数量，所以我们需要对应增加的比例来增加数据点的个数。得到的结果如图6.3所示。

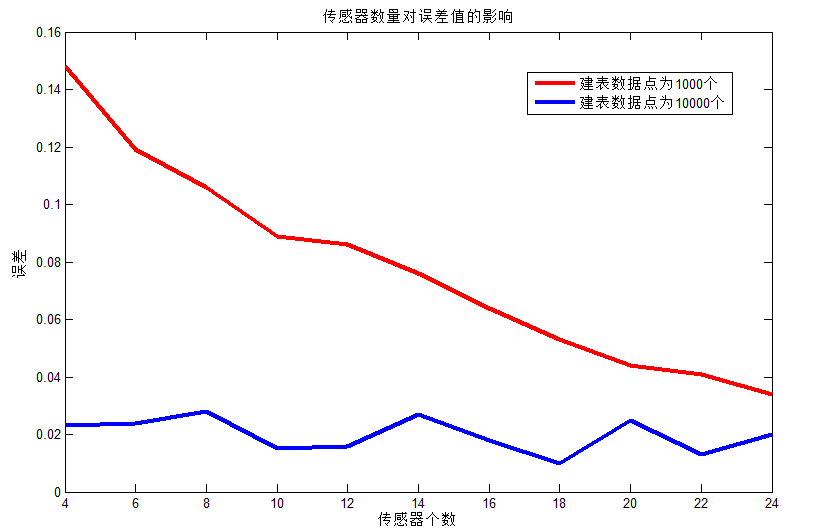


图6.3 误差折线图

由图中数据我们可以观察到，随着摄像头数量的增加，定位的误差在不断地减小，这里我们可以推断出在建表规模较小的情况下摄像头数量的增加可以使建表查表方法的检测精确度得到较大的提升，但是当建表数据规模达到一定程度时，传感器数量对实验误差的影响不大。

## 6.3 本章小结

本章我们模拟了建表查表法的大致过程，并在用python语言搭建了仿真的环境，利用我们搭建的环境我们对建表查表法的误差进行了实验分析，实验结果表明，建表查表法对目标检测的两个主要影响误差的因素分别是建表的数据规模和坐标系中部署的手机摄像头的个数，并且对于这两个因素而言检测结果都是与这两个因素的改变成反比的，建表的数据规模越大，误差越小，反之检测精度越高。摄像头数目越大，检测精度越高，误差越小。所以我们在部署实验环境时，如果我们需要搭建一个具有一定检测精度的系统，我们必须要在建表规模和摄像头数目之间做出取舍。如果我们需要较高的检测精度又想控制成本，减少摄像头的数目，那么我们需要扩大建表时的数据规模。如果我们需要较短的建表时间，同样检测精度不能降低，那我们就需要增加手机摄像头的部署数量。

# 结 论

本文从目前研究比较火热的图像识别的领域入手，开发了基于视频信息的无人机检测系统。最初是受到一篇图像运动物体追踪的启发，通过大量地阅读参考文献，经过许多次的仿真实验，最终开发出了这个基于视频信息的无人机检测系统。并且最终不仅能检测无人机，而且能定位无人机在一个坐标系中具体的二维平面坐标，具体的研究过程如下：

（1） 分析图像识别的核心思想，大量阅读前人的文献，了解了各种各样的图像识别的算法，并根据我们具体的系统要求，以及实现的难易程度选择最适合我们的算法进行研究。最终选择了模板匹配的方法和基于adaboost的级联分类器的方法。

（2） 学习相关技术，包括Android客户端编程，网络通信等系统所需的核心技术，为后期系统的具体实现打下铺垫。

（3） 结合模板匹配算法和级联分类器的方法实现系统的图像识别部分，最终实现整个系统。

（4） 学习Python语言，搭建本系统建表查表部分的仿真实验系统，为定位方法的可行性打下了理论的基础。

本文研究的基于视频信息的无人机检测系统，系统的核心图像识别的算法采用了模板匹配方法和基于adaboost的级联分类器方法，但是具体实现和核心的定位部分都是自己的独到见解。

本文通过此次的研究和学习完成了这个系统，但是这个系统还有很多需要完善的地方。比如图像识别的精度可以进一步的提高，核心的定位算法建表查表法查表过程的时间复杂度可以进一步的优化。

参 考 文 献

[1] 谭熊, 余旭初, 刘景正. 无人机视频数据定位处理系统的设计与实现[J]. 测绘通报, 2011,14(4):26-28.

[2] 田丰, 闫建国, 曹莹慧. 无人机视频中的目标定位研究[J]. 计算机测量与控制, 2010, 18(4):900-901.

[3] Zhang L, Lu H, Du D, et al. Sparse Hashing Tracking[J]. IEEE Transactions on Image Processing Society, 2016, 25(2):840-849.

[4] Li X, Shen C, Shi Q, et al. Non-sparse Linear Representations for Visual Tracking with Online Reservoir Metric Learning[J]. 2012, 157(10):1760-1767.

[5] Li H, Li Y, Porikli F. DeepTrack: Learning Discriminative Feature Representations Online for Robust Visual Tracking[J]. IEEE Transactions on Image Processing of the IEEE Signal Processing Society, 2016, 25(4):1834-1848.

[6] 万海清. 基于Android的运动检测视频监控应用的设计与实现[D].武汉： 华中科技大学, 2015.

[7] 俞晶晶. 基于多级分类器的人脸检测系统研究[D].北京：北京邮电大学, 2011.

[8] 王艳红. 基于OpenCV的运动目标检测与跟踪算法的研究[D].杭州： 杭州电子科技大学, 2013.

[9] 邵雯娟. 基于Socket的手机图像采集和传输系统的设计与实现[J]. 无线互联科技, 2013,16(11):48-49.

[10] 唐应成. 复杂场景下的视频对象跟踪技术研究[D].扬州：扬州大学, 2008.

[11] 赵振刚. 基于Haar-LBP特征和FDR-AdaBoost模型的眼睛定位[D].石家庄：河北师范大学, 2013.

[12] 陈龙. 基于DM642视频跟踪的无人机群控单元研究[D].南京：南京理工大学, 2015.

# 致 谢

本篇论文是在我的指导老师金乃高老师的细心指导下完成的，从选题到论文完成这段时间之中，金老师一直都在给我进行细致的教导，和不懈的支持，希望能借此机会对金老师表达最由衷的感谢。

感谢身边的同学，他们一路支持我，鼓励我，从他们的身上我学到了许多东西，正是由于他们的支持和鼓励，我才能顺利完成论文，这种协作创新精神和同学们之间的友情让我终生难忘。

感谢那些帮助我的其他老师们，还有感谢和我同窗四年的同学们，你们让我感觉到了家的温暖，因为有了你们这篇论文才得以顺利完成。感谢陪我度过大学时光的每一个人，有了你们大学的生活变得更加精彩。

最后感谢我的父母，有了你们的言传身教，才会有今天的我，这里想你们致以最崇高的敬意。