

内部

中国科学技术大学

工程硕士学位论文



人脸活体检测系统的设计与实现

作者姓名： 杨 扬

学科专业： 软件工程

校内导师： 凌强 副教授 曹洋 副教授

企业导师： 谢晓华 副研究员

完成时间： 二〇一六年三月五日

Limited

University of Science and Technology of China
A dissertation for master's degree
of engineering



**The Design and
Implementation of Face
Liveness Detection System**

Author's Name: Yang Yang
Speciality: Software Engineering
Supervisor: Prof. Qiang Ling Prof. Yang Cao
Advisor: Prof. Xiaohua Xie
Finished time: March 5th, 2016

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外,论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名: _____

签字日期: _____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一,学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权,即:学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版,允许论文被查阅和借阅,可以将学位论文编入有关数据库进行检索,可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

公开 保密(____年)

作者签名: _____

签字日期: _____

摘要

人脸识别技术得益于其方便易用，并且不易丢失的特性，近年来被越来越广泛地应用，在带给广大人民便利的同时，也创造了巨大的经济利益。但是，由于人脸特征极易通过照片、视频等途径被他人获取；即其他人可以通过仿冒的人脸特征，使用他人身份绕过人脸识别系统的验证，进而威胁该系统的信息安全。因此，很有必要在将人脸特征输入到人脸识别系统之前，首先对人脸特征进行活体检测，即判断该输入人脸是否为真实的人脸。但是，由于对于人脸特征的伪造和仿冒有多种可行的方案，每种方案均有其各自的特点，如何通过尽可能少的交互动作来全面而准确地对尽可能多的仿冒人脸进行识别，是一个比较难解决的问题。

为了得到准确的检测结果，首先对几种比较主流的人脸仿冒措施进行分析，再对一些主流的人脸活体检测的算法进行整合，归纳出一套具有较强鲁棒性和适应性的检测序列，即通过检测眨眼、张嘴以及摇头三个动作的完成程度来判断目标人脸是否为真实的人脸。

通过对系统进行功能性需求分析以及非功能性需求分析，将系统按照不同的功能划分成多个模块。在系统架构设计方面，出于对提升用户体验的角度考虑，系统需要满足实时性的需求，即能够实时地反馈用户的交互动作。这对集成了多个活体检测算法的系统而言是非常困难的，因此本系统采用基于异步通信的模块化设计，从而大幅度提高了主界面的帧率，保证了实时性。在检测算法设计方面，通过将人脸检测、人眼检测、人脸特征点定位、计算前背景运动一致性等算法的组合，有效降低了算法的时间复杂度，提高了系统的稳定性和鲁棒性。本系统在软件工程思想的指导下，以敏捷开发的方式予以实现，并且在开发完成之后对系统进行了完备的测试，验证了系统的有效性、实时性、高效性、健壮性以及可维护性。

关键字：人脸活体检测，人脸特征点定位，运动一致性检测

Abstract

In recent years, face recognition technology has been more widely used due to its portability and convenience. This technology has created tremendous economic values, but it still has fatal drawback. Since the face image can be spread through the photos and video, so people can easily get other people's facial feature, that can threat to system secure. Therefore, it is necessary to verify the validity of the faces, before these faces pass through the face recognize system. There are many ways to counterfeit facial feature, so how to use less interactive actions to detect fake face as fully as possible is a challenging problem.

To obtain accurate result, we compared and integrated several mainstream face liveness detection algorithms. And then, summed up a set of detection sequence, that contain blink detection, open mouth detection and yaw detection. This detection sequence has strong robustness and adaptability.

By functional requirement analyze and non-functional requirement analyze, the system can be divided into several parts. For enhance the user experience, this system is required to make read-time feedback reaction to user's interactive action. It is very difficult for a system, which incorporate many algorithms. This system uses a modular design based on asynchronous communication and greatly enhance the main frame rate. Besides, by using the combination of face detection algorithm, eye detection algorithm, facial landmark regression algorithm and motion consensus algorithm, we greatly reduce the time complex and improve the stability and robustness. This project is directed by software engineer theory, using agile development approach during implementation. And after completion of the project, we tested the whole system to verify the effectiveness, efficacy, robustness, maintainability.

Key Words: Liveness Detection, Landmark Regression, Motion Consensus

目录

第一章 绪论	1
1.1 选题的背景和意义	1
1.1.1 选题的背景	1
1.1.2 选题的意义	1
1.2 国内外研究现状	2
1.2.1 人脸活体检测算法研究现状	2
1.2.2 人脸特征点定位算法研究现状	3
1.3 本人主要工作	4
1.4 论文组织结构	4
第二章 相关技术介绍	6
2.1 半朴素贝叶斯模型原理	6
2.2 随机蕨回归算法原理	7
2.3 显式形状回归算法原理	8
2.4 主流人脸数据库介绍	9
2.5 Node.js 介绍	10
2.6 MongoDB 介绍	10
2.7 本章小结	11
第三章 系统需求分析	12
3.1 系统角色划分	12
3.2 系统功能性需求	12
3.2.1 目标眨眼检测	13
3.2.2 目标张嘴检测	13

3.2.3 目标摇头检测	13
3.2.4 统计数据	14
3.2.5 训练模型	14
3.3 系统非功能性需求	14
3.3.1 实时性	14
3.3.2 高效性	15
3.3.3 健壮性	15
3.3.4 可维护性	15
3.4 本章小结	15
第四章 系统概要设计	16
4.1 系统整体架构	16
4.2 系统功能模块设计	17
4.3 本章小结	19
第五章 系统详细设计与实现	20
5.1 客户机端详细设计与实现	20
5.1.1 控制器模块设计与实现	21
5.1.1.1 控制器模块的类结构设计	21
5.1.1.2 控制器模块核心处理流程	23
5.1.2 数据通信模块设计与实现	24
5.1.2.1 数据通信模块的类结构设计	24
5.1.2.3 数据通信接口设计	25
5.1.2.2 数据通信模块核心处理流程	27
5.1.3 图像采集模块设计与实现	29
5.1.3.1 图像采集模块的类结构设计	29

5.1.3.2 图像采集模块核心处理流程	29
5.1.4 人脸检测模块设计与实现	31
5.1.4.1 人脸检测模块的类结构设计	31
5.1.4.2 人脸检测模块核心处理流程	32
5.1.5 人脸特征点定位模块设计与实现	33
5.1.5.1 形状索引特征的设计	33
5.1.5.2 级联回归器的设计	34
5.1.5.3 人脸特征点定位模块的类结构设计	36
5.1.5.4 人脸特征点定位模块核心处理流程	37
5.1.6 眨眼检测模块设计与实现	41
5.1.6.1 眨眼检测模块的类结构设计	41
5.1.6.2 眨眼检测模块核心处理流程	42
5.1.7 张嘴检测模块设计与实现	44
5.1.7.1 张嘴检测模块的类结构设计	44
5.1.7.2 张嘴检测模块核心处理流程	45
5.1.8 摆头检测模块设计与实现	46
5.1.8.1 摆头检测模块的类结构设计	46
5.1.8.2 摆头检测模块核心处理流程	48
5.2 服务器端详细设计与实现	51
5.2.1 模型模块设计与实现	52
5.2.2 控制器模块设计与实现	53
5.2.3 视图模块设计与实现	55
5.3 本章小结	55
第六章 系统部署与测试	56

6.1 系统部署	56
6.2 系统功能性测试	59
6.3 系统非功能性测试	60
6.4 本章小结	61
第七章 总结	62
7.1 论文总结	62
7.2 个人收获	62
7.3 改善空间	63
参考文献	64
致谢	66

第一章 绪论

1.1 选题的背景和意义

1.1.1 选题的背景

近年来，随着人脸识别技术的迅速发展，越来越多的基于人脸识别技术的应用进入了日常生活领域。从支持人脸识别的考勤系统、支持人脸识别的门禁系统等较为传统的人脸识别应用，到新近流行起来的支持“刷脸”支付的手机应用软件、支持“刷脸”取款的 ATM，人脸特征凭借着其方便、友好的交互方式，迅速取代了其他的冗余、繁琐的验证方式，成为最受关注的新型验证方式之一。但目前，绝大多数的人脸识别系统都把研发的重心放置于目标身份识别的准确性以及识别算法的效率上；却忽视了人脸识别系统的安全性问题，即输入的人脸可以是一个非真实的脸（包括高分辨率的彩色打印照片、3D 打印的人脸模型、人脸面具等），使用者以此可以仿冒他人身份通过人脸识别系统的验证，对后续的应用安全性构成了极大地安全隐患。

任何一种验证系统，都会面临仿冒攻击的威胁。应对的方式无外乎以下两种：一是通过保护验证信息，使仿冒者不易获取，二是通过验证系统本身来鉴别验证信息是否有效。前者的主要应用场景就是密钥验证系统（包括对称密钥以及非对称密钥），使用此方式来保护验证系统对密钥持有者的要求较高，且密钥本身不易被他人获取。由于人脸的特殊性，它作为日常生活中人类的最重要的身份标示，被广泛地传播，极容易被他人获取；所以对于人脸验证系统而言，通过验证系统本身来鉴别验证信息的可靠性是唯一的解决方案。

因此，用于增强人脸识别系统的安全性的脸活体检测技术受到了越来越多人的关注。人脸活体检测技术旨在通过人脸信息来验证目标的合法性，更加具体的说，即在传统的人脸识别系统之前，增加一道防火墙，所有试图进行人脸识别的输入都将首先进行人脸活体检测，通过验证的输入才能继续进行人脸识别，否则就认为是非法输入，从而拒绝访问。

1.1.2 选题的意义

信息安全的重要性和敏感性以及无需赘述。在人脸识别领域，使用者不仅仅

需要方便快捷地进行身份验证，更希望验证系统能够保证其身份的安全性，即不被仿冒者盗用。人脸活体检测系统是一个添加在人脸识别系统之前的认证模块，采用带有人脸活体检测模块的人脸识别系统有助于在保证信息真实有效并且可以使用的情况下大大提高人脸认证的效率。传统的人脸识别中，其可欺骗性会导致系统运行的结果不可靠等非预期性结果。所以在高敏感性的应用环境中，对于机器识别的结果，还需要人工进行筛查纠错，这就直接降低了人脸识别系统的实用性。因为动用人力来进行筛查纠错不仅成本高，而且效率极其低下，也容易发生错误。

本次课题就是针对这一应用环境，通过图像信息来对输入信息的合法性进行验证，旨在通过尽可能少的交互量以及时间，最大限度地提高伪造人脸的识别率（即最大限度地提高伪造人脸成功的成本）。

综上所述，人脸活体检测系统具有非常广阔的应用场景，人脸活体检测技术能够为人脸识别系统提供可靠的输入数据，使人脸识别系统的应用范围得到进一步的拓展。

1.2 国内外研究现状

1.2.1 人脸活体检测算法研究现状

随着人脸识别技术的成熟，在 2000 年左右，活体检测技术逐渐被各大领域重视。人脸活体检测技术的目标是通过一幅（组）人脸照片来判断照片中的人脸是否为真实的人脸，而非模型或者照片。

Herbert Bay 等人于 2006 年提出了一种基于 SFM (Structure from Motion) 的活体检测模型。该模型通过定位眼睛、嘴巴等显著位置来预测特征点的三维深度，从而实现判断活体的方法。Klaus Kollreider 等人于 2009 年提出了利用光流 (Optical Flow) 来分析人脸各部位的移动量，最终进行活体检测的方法。孙霖于 2010 年提出了利用人脸识别进行多模活体检测的概念。其思想大多是将多个检测源的数据进行混合，然后计算加权检测结果，来进行评判。

李翼于 2011 年采用高斯差分 (Difference of Gaussian, DoG) 特征结合稀疏低秩双线性逻辑斯回归的方法，在 NUAA 数据库上获得了很高的准确率。杨健伟于 2014 年提出了基于运动一致性的活体检测的方法，通过计算微纹理，计算脸部与背景的运动一致性。为人脸活体检测提供了新的方向。曹瑜于 2014 年

提出了局部二值特征（Local Binary Pattern, LBP）结合卡方统计量来判断活体人脸的方法，使用该方法能够有效提高检测的准确率。刘华成于 2014 年提出了使用奇异值分解（Singular Value Decomposition, SVD）与 HSV 直方图的方法用来检测活体，具有较高的鲁棒性。

由于活体检测的巨大市场，工业界也推出了很多里程碑式的产品。例如 2012 年，由北京旷视科技推出的 Face++，在 LFW 评测集上连续创造世界纪录，并且在人脸活体检测上，也有较为出色的表现。不过目前 Face++ 仅以 API 调用的形式为开发者提供服务，并没有公开其实现算法。阿里巴巴公司的手机支付宝 app 于 2015 年推出了“刷脸”登录的功能，手机支付宝采用的是交互式验证手段，需要用户进行做出眨眼、点头等动作进行验证，其识别率达到 90% 以上。

1.2.2 人脸特征点定位算法研究现状

人脸特征点定位技术的目标是通过一幅人脸图片，得到其中人脸的语义特征点位置的技术。其结果的准确性通过如公式(1.1)所示的误差函数来衡量：

$$\| \mathbf{S} - \mathbf{S}_{\text{Ground True}} \|_2 \quad (1.1)$$

其中 \mathbf{S} 是指通过预测得到的人脸特征点位置， $\mathbf{S}_{\text{Ground True}}$ 表示实际的人脸特征点位置，但由于在预测阶段， $\mathbf{S}_{\text{Ground True}}$ 是未知的，因此不能直接通过最小化误差函数来得到 \mathbf{S} 。现阶段有两类方法，分别从两个不同的角度去得到 \mathbf{S} ，因此绝大多数主流的人脸特征点定位算法都可以被分为如下两类：基于最优化算法的人脸特征点定位算法以及基于回归算法的人脸特征点定位算法。

基于最优化算法的人脸特征点定位需要再设计一个能量函数，该能量函数要求与上述误差函数相关性大，通过最优化该能量函数来减少误差。此类方法的结果误差直接取决于能量函数是否能够反映误差函数的变化以及能量函数能否被最优化。例如由 Timothy F. Cootes 等人于 2001 年提出的 AAM（Active Appearance Models）算法就是典型的基于最优化算法的人脸特征点定位算法。AAM 算法通过训练得到一个表观模型（Appearance Model）；然后在预测阶段，使用该模型最小化纹理误差（Texture Residual）来达到减少误差的目的。但是，由于表观模型的形变能力不强，所以对于训练集之外的人脸，不能做到很好的拟合；并且 AAM 算法受初始化数据的影响较大，鲁棒性不强。

基于回归的算法并不需要另外设计一个能量函数，此类算法是通过训练得

到回归器，在预测阶段直接通过输入映射出输出结果，此类算法的普遍优点是能够有效利用大容量的训练集，从而提高回归质量。David Cristinacce 等人与 2007 年提出的 Boosted Regression 框架结合 ASM（Active Shape Models）算法以及 Michel Valstar 等人与 2010 年提出的 Boosted Regression 框架结合 Graph Models 的方法中，均采取为每个特征点训练一个回归器的策略。这种策略有两个很明显的劣势：不容易区分一些特征区分度不高的特征点（比如脸颊上的两个特征点）、没有考虑特征点之间的位置关系。Piotr Doll’ar 等人于 2010 年使用所有特征点进行整体回归，并提出了级联回归器的算法框架，该方法通过贝叶斯概率模型逐步对结果进行修正。Xudong Cao 改进了级联回归器的框架，并且引入了新的语义特征，在人脸特征点定位上取得了非常好的效果。

1.3 本人主要工作

- 研究人脸活体检测系统的定位，了解相关的基本功能需求；并且通过阅读大量文献，掌握国内外有关人脸活体检测的发展现状。
- 根据系统的应用场景，明确系统角色，确定最后的项目功能性需求以及非功能性需求。
- 通过项目的需求分析，确定项目的整体架构以及相关系统功能模块的划分。
- 根据需求分析，尝试进行核心算法的实现并且通过不同算法的组合来达到项目需求，并最终确定核心算法流程。
- 细化项目的各个模块设计，抽象出各个模块之间的通信接口，按照软件工程的项目规范把所有模块实现。
- 搭建系统的测试环境，并根据系统的功能性需求以及非功能性需求进行完善的测试。

1.4 论文组织结构

本论文的内容结构安排如下：

第一章为绪论。该章论述了人脸活体检测系统的项目背景和项目意义，并且介绍了现阶段工业界和学术界有关人脸活体检测以及人脸特征点定位的最新成果。

第二章为相关技术介绍。主要介绍了本项目涉及的一系列算法原理以及实

现框架，其中包括半朴素贝叶斯模型、随机蕨回归算法、显式形状回归算法、Node.js 以及 MongoDB。

第二章 相关技术介绍

2.1 半朴素贝叶斯模型原理

半朴素贝叶斯模型（Semi-Naive Bayes Model）对朴素贝叶斯模型（Naive Bayes Model）的改进。该算法由 Geoffrey I. Webb 等人于 2005 年提出。该算法填补了朴素贝叶斯模型的两大缺陷：需要大量训练样本以及独立性假设与大部分特征不相符。假设输入的 N 维特征向量的形式为： $f = (f_1, f_2, \dots, f_N)$ ，模型的输出类别为 $C \in \{c_1, c_2, \dots, c_k\}$ 。贝叶斯模型是一个通过计算后验概率并求出最大值所属的类别，并作为输出。贝叶斯模型的形式化表示为：

$$H(f) = \operatorname{argmax}_k P(C=c_k | f_1, f_2, \dots, f_N) \quad (2.1)$$

使用贝叶斯公式可以将后验概率的形式变化为先验概率的形式：

$$H(f) = \operatorname{argmax}_k P(f_1, f_2, \dots, f_N | C_k) * P(C_k) \quad (2.2)$$

在朴素贝叶斯模型中，假设所有的特征都是互相独立的，因此上述联合概率分布等于各个特征的先验概率的乘积，即为：

$$P(f_1, f_2, \dots, f_N | C_k) = \prod_{i=1}^N P(f_i | C_k) \quad (2.3)$$

但是这个假设过于严格，在实际的应用中，该假设往往是不成立的，这就直接导致了朴素贝叶斯模型得出的结果与真实情况有较大的偏差。

半朴素贝叶斯模型取消了这个全局独立性假设，而是对特征进行分组，假设各个组内特征不独立，各个组间特征独立。假设 N 个特征被划分为 L 个组，每个组内含有 S 个特征，则上述的联合概率分布可以表示为：

$$P(f_1, f_2, \dots, f_N | C_k) = \prod_{l=1}^L P(F_l | C_k) \quad (2.4)$$

通过计算每个组内的联合概率分布，并通过独立事件的组合来求得该先验概率，再通过最大化先验概率来得到整个模型的最终输出。

半朴素贝叶斯模型通过组内容量的大小来协调算法复杂度与准确度之间的关系。S 的量级与算法准确度成正比，而 L 的量级与算法复杂度成正比。

2.2 随机蕨回归算法原理

随机蕨回归器是由 Dollár 等人在 2010 年首次提出。该算法将半朴素贝叶斯模型进行了应用到了回归器的设计上。随机蕨回归器采用了级联式回归的设计方式，由多个回归器共同来约束输出结果。更具体地说，随机蕨级联回归框架的每一级回归器都是对上一级回归器所做出的结果的一个修正。该算法不同于之前流行的大部分算法，采用参数化的约束来控制输出结果；在随机蕨回归算法中，最终的输出是所有的训练集的一种线性组合，由此保证了在不依赖参数化约束模型的情况下，依然能够控制输出结果。

以图像的特征点定位的应用为例，随机蕨回归器在特征选择上，采用的是像素差异特征。选取的是随机生成的 N 对像素对的强度差，组成原始的特征向量。其形式如下：

$$X = \{x_1, x_2, \dots, x_n\} \quad (2.5)$$

通过随机采样，将特征的数量缩减至 S 个，并且在每次迭代时生成 S 个划分阈值，将 S 个经过随机采样的特征映射成 S 个二进制位。映射规则如下：

$$\text{binary}_i = \begin{cases} 0, & \text{if } x_i < \text{threshold}_i \\ 1, & \text{if } x_i \geq \text{threshold}_i \end{cases}, \quad 1 \leq i \leq S \quad (2.6)$$

在训练时，通过将 S 个二进制位的映射，可以把 S 维特征映射为 0-2^S-1 之间的一个整数 d，将该训练集存于序号为 d 的桶中，完成第一个回归器的训练。随后用第一个回归器预测出所有训练集图片的特征点，并将该误差作为第二级回归器的回归目标，以此类推，完成所有回归器的训练。

在预测时，对于每个回归器，对输入图片进行取“像素差异特征”后，再进行随机采样，并将映射的到的序号为 d 的桶中的所有回归目标中，选取一个残差最小的回归目标，作为本次回归器的输出。形式化表示如下：

$$R_i = \operatorname{argmin}_R \sum_i d(S_i, S) \quad (2.7)$$

其中，函数 $d(S_i, S)$ 表示第一个回归目标与序号为 d 的桶中所有回归目标的残差。

在预测时，对于整个级联回归器，将输入图片依次通过所有的回归器，将每个回归器输出的结果进行累加，即为整个级联回归器最终的输出。形式化表示如下：

$$R_{\text{Final}} = R_0 + R_1 + \dots + R_S \quad (2.8)$$

随机蕨回归器的训练，预测速度都非常快，并且由于采用了级联的方式约束结果，随着级联回归器数量的增加，回归器输出的误差粒度逐渐细化，可以得到比较准确的输出结果。由于级联回归器中的每一级回归器的回归目标为上一级回归器与真实形状之间的残差，因此采用随机蕨这种较弱的回归器亦能取得较好的回归效果，并且能够极大地提高回归速度。

同样得益于级联回归的设计，经过 Dollar 等人证明，随机蕨回归器可以在每一个阶段以指数级的速度逐步收敛，因此该算法具有非常强的鲁棒性，可以在不同的输入的情况下得到稳定的输出结果。

2.3 显式形状回归算法原理

显式形状回归（Explicit Shape Regression）算法由 Xudong Cao 等人于 2014 年提出。该算法也是以随机蕨回归算法为基础，并且在全局特征选取，整体架构设计，以及初始化数据等方面进行了大量改进，使其能够非常高效准确地进行人脸特征点定位。

在全局特征选取上，Xudong Cao 提出了形状索引特征（Shape Indexed Feature）。形状索引特征相较于原始的随机蕨回归算法的随机像素差异特征，能够保持非常好的语义一致性。更具体地说，随机像素差异特征是在全局坐标系上随机地取点对，并计算像素的强度差作为特征，各个像素对的坐标是以全局坐标系下的绝对坐标来表示的，不能根据输入图像的实际情况进行改变。形状索引特征同样是从像素差异特征改进而来：首先在全局坐标系下随机选取点对，而每个点对并非以绝对坐标的形式保存，而是保存为距离最近的特征点的相对坐标；在实际计算特征之时，再通过上述方式逆向映射会实际的坐标点。因此形状索引特征的各个坐标能根据不同的输入形状进行调整，从而保证特征的语义一致性，增强算法的准确率以及鲁棒性。

在整体架构的设计上，该算法创造性地使用了 2 层回归器的设计，其结构如下：

其中，高层次的回归器 R_t 被称为外层回归器，而低层次的回归器 r_k 被称为内层回归器。与上一节提到的级联回归的思想类似，显式形状回归的每一层也采用级联式的设计：每一个回归器的输入是上一个回归器的输出，每一个回归器的

回归目标是上一个回归器的结果与真实形状的残差。但由于单个内层回归器的回归能力很有限，并不能很好地纠正上一层的误差，并且容易造成过拟合的现象。因此，Xudong Cao 采用了两层回归器的结构：外层回归器形状索引特征的初始化，内层回归器使用该形状索引特征进行回归，即同一个外层回归器下属的内层回归器均使用相同的形状索引特征，这样能够提升训练与预测的速度，并且使得结果更加准确。

此外，显式形状回归还提出了初始数据扩增来优化回归结果。由于级联式回归器的特点所致，对于每一个输入，都需要有一个初始化形状作为第一个回归器 R_0 的输入。之前的方法无外乎采用所有训练集的均值形状或者选取某一个全局残差最小的形状作为初始形状输入。这会导致训练得到的回归模型的泛化能力不足，并且由于随机蕨算法的随机性，还会导致输出结果的不稳定（对于同样的输入，回归得到差距较大的输出形状）。Xudong Cao 采用了初始化数据扩增的方法：在训练时，随机地采用多个形状作为初始形状进行回归，可以大幅度增强回归模型的泛化能力，有效避免过拟合；在预测时，将通过多个初始形状进行回归得到的结果的均值进行输出，能够提高输出结果的稳定性。

2.4 主流人脸数据库介绍

- **BioID:** BioID 数据库由 Oliver Jesorsky 等人于 2001 年提出，该数据库包含了 1521 张人脸图片，每张人脸照片采用 20 个特征点进行标注，该数据库的人脸均由相同距离的正脸在实验室环境下采集而成。
- **LFW87:** LFW87 数据库由 XX 于 20008 年创建，其中大量图片提取自 LFW (Labeled Face in the Wild) 数据库，该数据库中包含 4002 张训练图片以及 1716 张测试图片，每张人脸图片使用 87 个特征点来标注。
- **Helen:** Helen 数据库由 XX 于 2012 年创建，该数据库采用 2330 张高分辨率的人脸图片，每张人脸图片使用 194 个特征点进行标注。由于该数据库的高分辨率、多特征点的特点，使得该数据库训练出来的模型能够更好地利用图

像的细节信息，使结果更加准确。

2.5 Node.js 介绍

Node.js 是一款流行的高性能开源的网络服务与应用平台，使用 JavaScript 作为脚本语言。Node.js 的构建基础是 Google 公司的 V8 引擎（Chrome JavaScript Runtime V8），而 Google 的 V8 引擎亦是目前全球效率最高的 JavaScript 解释器之一，这使得 Node.js 拥有非常好的并发性能以及非常快的执行速度。

Node.js 能够提供一个使用事件驱动来实现异步开发的优秀解决方案。由于 Node.js 中绝大多数的 API 都是采用基于事件触发的异步调用模式设计的，非常适合进行高并发、高 IO、低运算的网络开发。

Node.js 旨在提供一种可伸缩、可扩展的高性能运行环境，以解决 WEB 服务器的性能瓶颈问题。由于传统的 WEB 服务器需要为每一个访问链接创建一个处理线程，会导致服务器只能响应十分有限的请求。而 Node.js 由于采用了单进程、单线程的事件驱动的设计方式，把每个新的请求加入事件循环（event loop）的轮询队列而非创建新线程，这个方式极大地减少了系统开销，提高了系统的处理并发连接的能力。

Node.js 还是用模块化的设计来划分不同的功能包，这一特点也在很大程度上提高了开发效率。例如 HTTP 模块内封装了大量 HTTP 以及 TCP/IP 协议栈的操作函数，采用 Node.js 可以有效降低服务器的资源消耗，提高脚本的性能。

虽然 Node.js 与 2013 年才正式发布，但是已经得到了业界的一致认可，其中国际著名的社交网站 LinkedIn、国际著名的开源项目托管网站 GitHub 和国内的阿里巴巴都把 Node.js 作为项目的重要组成部分。

2.6 MongoDB 介绍

MongoDB 是一款分布式的非关系型（NoSQL）数据库系统，该数据库系统于 2009 年发布，采用 C++ 语言实现。MongoDB 作为一款非关系型数据库，更加适合分布式应用的场景，与传统的关系型数据库相比，有如下优点：

- **高性能：**相比于关系型数据库的查询缓存（针对每一条查询语句设置缓存信息，每当查询的目标发生改写，会导致缓存失败），MongoDB 的缓存粒度更加细，每当数据改写后能够及时更新缓存，提高查询性能。此

外，由于 MongoDB 并不支持关系型数据库的事务操作，可以进一步减少操作时间，提升性能。

- 易扩展性：MongoDB 采用的是面向文档存储的数据模型，这是其非常容易在多个服务器之间进行数据分割；此外，由于 MongoDB 的各个集合之间取消了外键的约束，因此也方便数据进行扩展。
- 高灵活性：MongoDB 采用键值对的形式进行数据存储，并不要求每个表项拥有相同的字段，可以灵活地存取数据而避免频繁地增加减少字段。

2.7 本章小结

本章节从半朴素贝叶斯模型开始，具体介绍了两种应用了该模型思想的级联式回归算法：随机蕨回归算法以及显式形状回归算法。推导了半朴素贝叶斯模型的数学模型，阐明了两个回归算法的历史以及详细的算法流程。除此之外，本章节还介绍了本项目所涉及的两个开源软件，分别是 JavaScript 运行平台 Node.js 以及非关系型数据库 MongoDB。总的来说，本章节详细介绍了本项目所涉及的算法以及软件框架，为后续提出系统设计打下了良好的理论基础以及应用基础。

第三章 系统需求分析

在第一章中，已经说明过人脸活体检测系统对于信息安全的重要意义，即旨在保护人脸识别系统不受假冒的人脸的攻击。为了解决上述问题，设计了一个通过简单交互式操作进行活体检测，并且提供验证信息的可视化管理、提供 web 端口供管理员使用与操作的人脸活体检测系统。

该系统可以通过采集一系列的人脸图片（序列帧）以及其他辅助信息，判别图片中的人脸是否为真实的活体，而非仿冒攻击的手段（比如照片、视频、三维模型等）。本章对人脸活体检测系统确定了项目的设计目的，并且分别进行功能性需求分析和非功能性需求分析。

3.1 系统角色划分

表 3-1 人脸活体检测系统系统角色表

角色	职责或功能
用户	系统的用户，使用人脸活体检测系统进行验证的使用者
管理员	系统管理员，可以设置参数以及查询结果

如表 3-1 所示，本系统的系统角色有两类，分别为用户和管理员。其中用户是系统前端的使用者，也是被验证者，系统与用户交互并采集用户的图像序列，进行人脸活体检测。管理员是系统的后端管理员，主要负责系统的参数设定，并且可以集中查看整个网络中的系统的验证结果。

3.2 系统功能性需求

本系统为一个交互式人脸活体检测系统，因此核心功能为验证输入人脸的有效性，同时还需要与使用者进行简短的交互。系统与使用者的主要交互方式为图形界面指示并伴随语音提示的方式。

从验证方式上，系统可以支持眨眼检测、张嘴检测以及摇头检测。选择此三种验证方式可以在最少的交互下达到比较优秀的识别率。

系统的主要功能性需求如下：目标眨眼检测、目标张嘴检测、目标摇头检测、统计数据、训练模型。

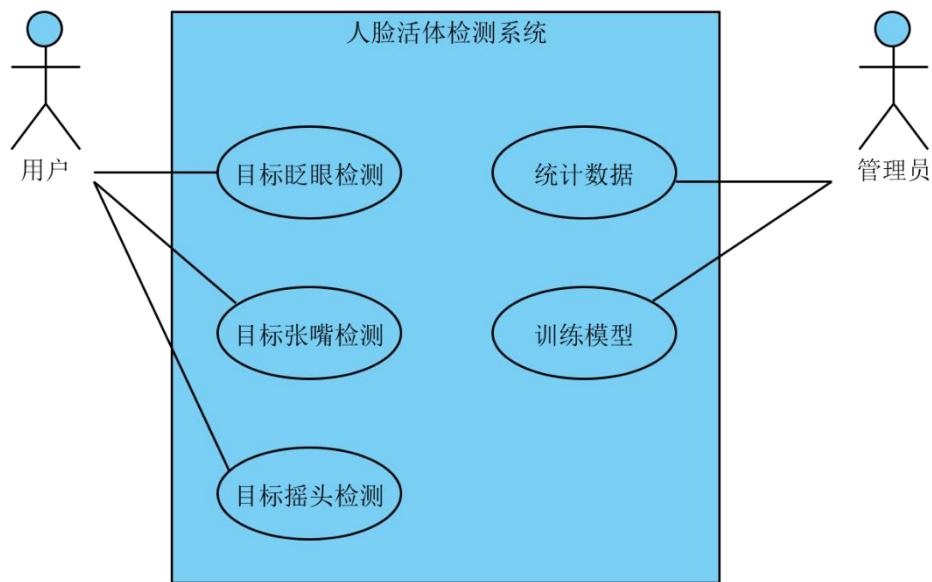


图 3-1 人脸活体检测系统用例图

3.2.1 目标眨眼检测

目标眨眼检测的用例主要对应人脸眨眼检测的功能。系统可以进行图形界面提示以及语音提示，指导用户进行眨眼动作，同时捕获序列帧。对于用户端采集的一系列序列帧，通过人脸特征点定位确定眼睛的位置，再建立状态机并检测出用户是否眨眼。在人脸眨眼检测中，用户仅需对着摄像头眨眼，即可完成验证。

3.2.2 目标张嘴检测

目标张嘴检测的用例主要对应人脸张嘴检测的功能。系统可以进行图形界面提示以及语音提示，指导用户进行张嘴动作，同时捕获序列帧。系统对于用户端采集的一系列序列帧，通过人脸特征点定位确定嘴巴的位置，能够用过阈值判断检测用户是否张嘴。在人脸张嘴检测中，用户仅需对着摄像头张嘴，即可完成验证。

3.2.3 目标摇头检测

目标摇头检测的用例主要对应人脸摇头检测的功能。系统可以进行图形界面提示以及语音提示，指导用户进行摇头动作，同时捕获序列帧。系统对于从用户端采集的一系列序列帧，通过人脸特征点定位，可以确定人脸的轮廓，再进行前景建模与背景建模，最后能够进行人脸摇头检测，通过阈值判断用户是否摇头。

在人脸摇头检测中，用户仅需跟随图形界面中的滑块摇头，即可完成验证。

3.2.4 统计数据

人脸活体检测系统为一个联网验证系统，所有用户的验证结果会存储在数据库服务器中。系统需要提供统计数据并且输入输出的接口。在数据输入端，该系统可以接受指定网络内所有的客户端所发送的验证结果并且将其保存；在数据输出端，系统可以通过 web 页面的方式向管理员提供验证数据的可视化管理以及统计归纳。

3.2.5 训练模型

训练模型的用例对应训练机器学习算法中所需要涉及的回归模型的功能。该回归模型支持离线训练，即管理员可以在管理员界面中开启训练模型的线程，当该模型训练完毕后，模型中的所有参数均已经确定，因此在使用过程中可以直接加载，而无需再次训练。

系统能够自由地让管理员用户选择训练数据的类型，包括训练集的类型、特征点的数量以及位置、回归器的架构以及级联数量等属性；并且系统可以加载对应的训练数据，并且根据管理员所设置的参数组合进行回归模型的训练。

3.3 系统非功能性需求

上一节简要介绍了系统各个主要功能性需求，本届内容为系统的非功能性需求。软件系统的非功能性需求即软件系统为了满足用户的业务需求或者使用体验而必须满足的且在功能性需求之外的特性。软件系统的非功能性需求不仅仅影响着功能性需求的完整度，同时也影响着整个软件系统的质量。软件系统的非功能性需求贯穿着整个软件生命周期的软过程，结合本项目需求，从以下三个方面进行展开叙述：实时性、健壮性以及可维护性。

3.3.1 实时性

本系统设计的初衷是一个交互式的人脸活体检测系统，因此需要与用户进行交互并及时给予反馈。因此，反馈是否及时对用户体验的影响非常大，甚至会影响用户的交互动作能否正常完成。在实际应用中，所有的用户反馈均需要达到实时响应的标准，其中主视频的帧率不能低于 12FPS。

3.3.2 高效性

本系统作为一个交互式的人脸活体检测系统，其主要应用场景为作为人脸识别系统的过滤器，且在大部分实际应用中，用户对于过于繁琐的交互动作以及长时间的等待是非常反感的。因此，在交互动作的设计上，既要考虑活体检测的成功率，也要兼顾用户体验。在实际应用中，每个交互动作的时间不能超过 3 秒，整体交互动作时间不能超过 10 秒。

3.3.3 健壮性

本系统作为一个活体检测系统，即需要直接面对各种形式的仿冒攻击。因此，健壮性是最基本的系统特性之一，因为只有保证了系统的健壮性，才能够进一步提升识别的正确率。更具体的说，系统需要应对各种形式的输入而不能崩溃，并且需要有良好的容错性。

3.3.4 可维护性

系统具有良好并且详尽的注释以及文档，因为在可以预见的未来，人脸活体检测的方式未来很有可能根据实际使用情况进行扩增，因此系统的检测功能将会越来越多，一个良好的系统架构不仅仅便于系统开发过程中的需求变更以及模块扩展，同时也方便了代码的后期维护以及重构。同时，系统还应具备跨平台的特性，保证用户在不同的操作系统下均能够部署以及使用，并且保证相同或者近似的用户体验。

3.4 本章小结

本章从系统的角色划分开始，着眼于系统的功能性需求以及非功能性需求，从两个方面对人脸活体检测系统进行了详尽的需求分析。在功能性需求方面，本章基于系统总体用例图主要阐述了系统的各大功能，并且对每个功能进行了细分描述。在非功能性需求方面，本章从用户体验以及软件工程规范的角度入手，并且对系统需要满足的各大特性进行展开描述。通过对项目进行需求分析，逐步细化项目所需要完成的目标，为后续章节提出系统的概要设计以及详细设计奠定了基础。

第四章 系统概要设计

在上一章节，已经详细介绍了人脸活体检测系统的功能性需求以及非功能性需求。本章节将从软件工程的角度提出对整个系统的概要设计，即根据系统的功能性需求以及非功能性需求，确定系统的整体框架以及模块分布。具体内容如下：

4.1 系统整体架构

本项目的开发过程采用自顶向下，逐步精化的开发模式。为了兼顾人脸活体检测系统的非功能性需求，系统将实现复杂度较高的模块置于客户机端，可以确定系统的客户机端需要承担较为复杂的功能以及较为繁重的计算量。因此，本项目采用 C/S（客户机/服务器）架构进行开发。

相比近几年流行的 B/S（浏览器/服务器）架构，虽然 B/S 架构拥有免安装、部署扩展容易、维护方便的优点；但 C/S 架构能够充分发挥客户机 PC 的处理能力，保证客户端的执行效率以及用户体验，并且在正常的检测过程中，客户机与服务器之间的通信次数以及通信量均在可控范围，因此 C/S 架构非常适合作为人脸活体检测的整体架构。

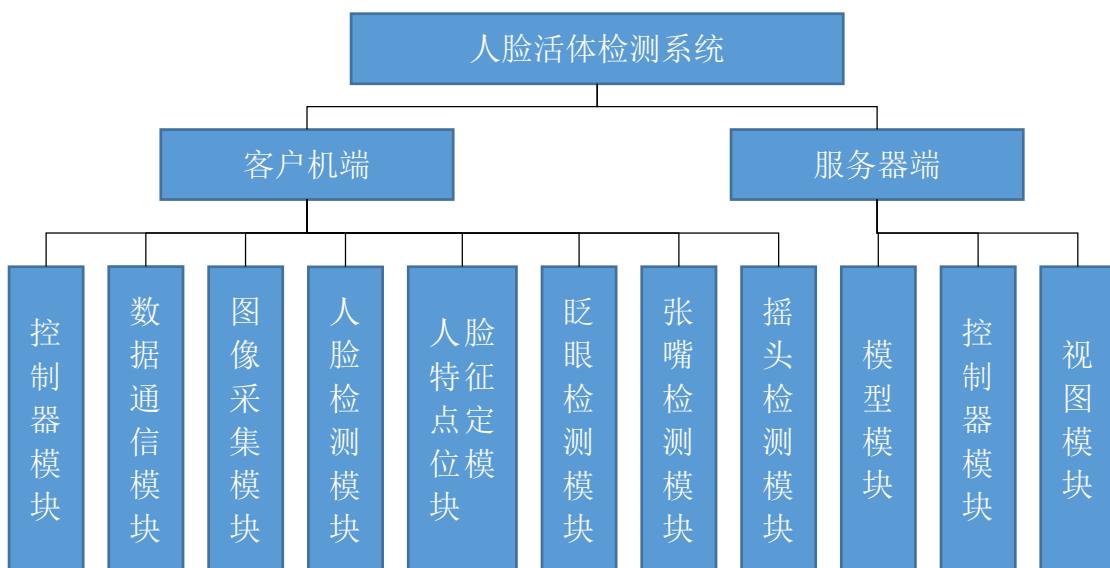


图 4-1 系统整体架构图

如图 4-1 所示，客户机运行在运行在用户的内存空间内，作为一个独立的进程运行在用户态。客户机采用模块化设计，分成 8 个模块：控制器模块、数据通信模块、图像采集模块、人脸特征点定位模块、人脸检测模块、眨眼检测模块、张嘴检测模块以及摇头检测模块。其中的核心模块为控制器模块；数据通信模块、图像采集模块、人脸检测模块以及人脸特征点定位模块为内部调用模块；眨眼检测模块、张嘴检测模块以及摇头检测模块为主要功能模块。为了保证程序执行的高效率，各个模块均采用单独线程的方式进行执行，而线程间通信以及调用通过控制器模块完成。为了满足功能模块的可扩展性，各个功能模块与控制器使用的设计模式为简单工厂模式，通过简单工厂模式隐藏各个功能模块内部的实现细节。

在服务器端，采用成熟的 web 应用开发框架：基于 Node.js 的 express 框架，该框架提供丰富的 http 快捷方法以及可以任意排列组合的 connect 中间件，可以迅速实现一个集健壮、友好等优点于一身的 web 应用。服务器端的架构采用 MVC 结构，使数据与业务逻辑相分离，简化了开发难度，提高了开发的速度，增强了系统的鲁棒性。

4.2 系统功能模块设计

系统整体架构图展示了人脸活体验验证系统的总体模块布局，本节将对各个模块之间的协作方式进行描述。

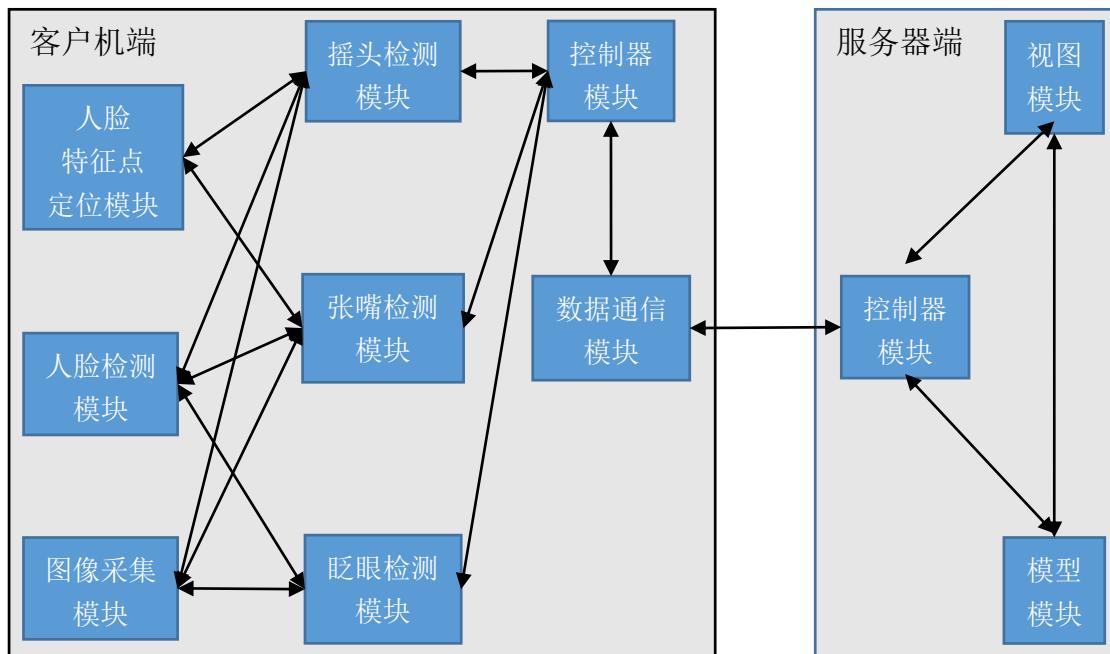


图 4-2 系统功能模块间通信示意图

客户机端的控制器模块为客户机的逻辑核心，负责控制客户机端的其他模块并且进行信息交互。控制器模块在客户机端运行之后会首先启动，采用独立线程的方式运行，逐个调用其他模块，并且根据各个模块的返回值，最终输出活体检测的结果。

数据通信模块是客户机端与服务器端通信的模块，主要功能为校验系统版本、获取人脸特征点配准训练文件、获取随机验证指令、返回验证结果以及其他附属信息等。数据通信模块在客户机端启动之时，为之提供初始化数据；在验证完毕以后，向服务器端返回验证结果。

眨眼检测模块负责完成对于输入的人脸进行眨眼检测的功能。该模块接受控制器模块的调用指令，使用独立线程的方式运行。在执行阶段，该模块调用图像采集模块获取用户输入的图像，调用人脸特征点定位模块进行定位，最后对人脸的眼部区域进行眨眼检测，并且向控制器模块返回检测结果。

张嘴检测模块负责完成对于输入的人脸进行张嘴检测的功能。该模块接受控制器模块的调用指令，使用独立线程的方式运行。在执行阶段，该模块调用图像采集模块获取用户的输入图像，再调用人脸特征点定位模块进行定位，最后使用人脸特征点定位的结果进行张嘴检测，并且向控制器模块返回检测结果。

摇头检测模块负责完成对于输入的人脸进行摇头检测的功能。该模块也是接受控制器模块的调用指令，同样使用独立线程的方式运行。在执行阶段，该模块首先通过调用图像采集模块的接口获得人脸的输入图像，并且调用人脸特征点定位模块的接口进行人脸特征点定位，最后通过人脸特征点的坐标划分图像区域，进行摇头检测，并且将结果返回给控制器模块。

人脸识别模块负责完成对人脸以及人脸的显著特征部位——眼睛进行检测。该模块接受 2 个功能模块的调用指令。在执行阶段，可以在输入的图片中搜索人脸以及眼睛的大致位置，并且返回人脸以及人眼的包围盒的参数。

人脸特征点定位模块负责完成的功能为人脸图片的特征点定位。该模块接受 3 个功能模块的调用指令。在程序执行阶段，对于输入的图片，首先进行人脸区域的粗定位，再基于粗定位的人脸区域进行人脸特征点定位，并且将定位的结果返回给调用指令的发起模块。

图像采集模块的主要功能为，将通过摄像头采集到的图像序列提交给功能模块。该模块接受 3 个模块的调用指令，并且以独立线程的方式运行。在程序执行阶段，该模块以最高帧率获取摄像头所采集的图像，并且保存；等待功能模块的调用指令，并且把图像给调用指令的发起模块。

在服务器端，控制器模块是 MVC 架构的核心模块，用以处理客户机端发来的请求并且协调模型模块以及视图模块进行响应。模型模块主要负责将服务器端的数据结构与数据库中的表项进行映射，并且封装业务逻辑接口，供控制器模块调用。视图模块由一系列模板页面组成，该模块的主要功能为，接受控制器模块的实例化参数，生成 html 页面，最后将目标页面返回给控制器模块。

4.3 本章小结

本章节提出了人脸活体检测系统的总体架构以及各个功能模块的划分。首先通过分析项目的需求，确定本项目采用 C/S 架构，并且使用系统功能模块图对于服务器端以及客户机端的功能模块进行划分。随后使用系统功能模块间通信示意图来明确各个功能模块之间的层次关系、调用顺序以及接口规范等，为后续提出系统的详细设计打下了基础。

第五章 系统详细设计与实现

在前两章提出的人脸活体检测系统的需求分析以及概要设计的基础上，本章将会对整个系统及其各个模块进行详细设计。

5.1 客户机端详细设计与实现

根据系统总体架构图的设计，进行细化设计，通过抽离出各个模块的通用部分，提出客户机端的系统整体类图，如图 5-1 所示：

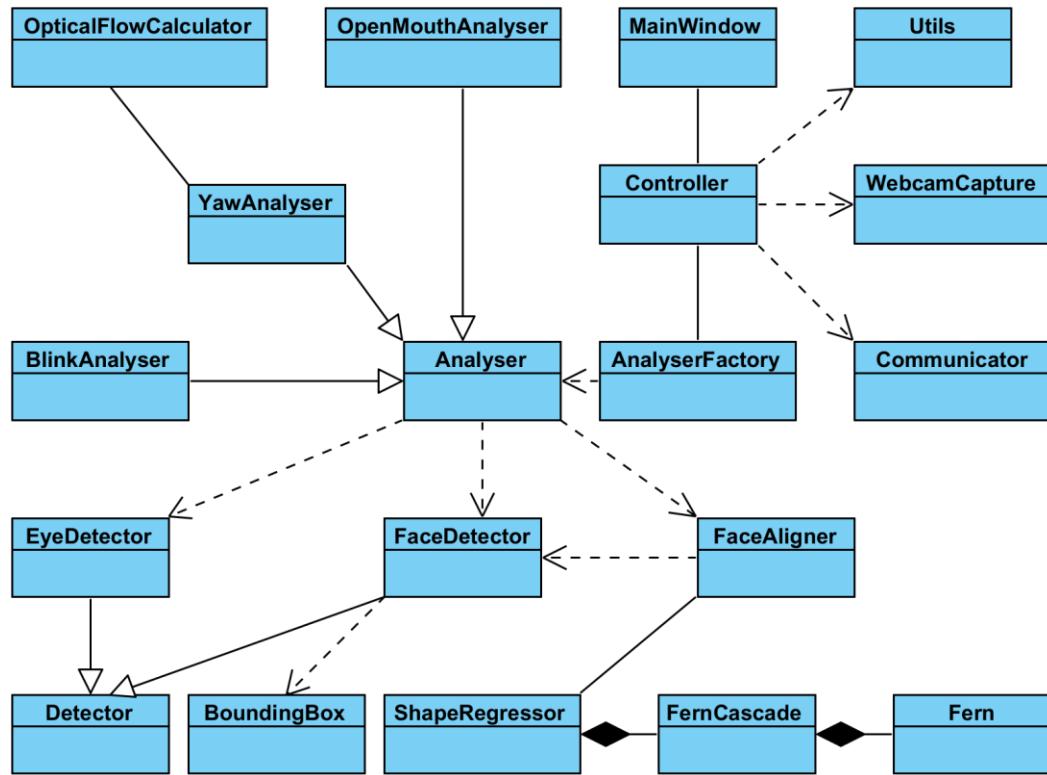


图 5-1 客户机端系统整体类图

客户机端的各个主要功能模块均采用独立线程的方式进行设计，线程间的通信采用 Qt 的信号槽（Signal and Slot）机制进行。其中控制器模块由 3 个类组成：MainWindow 类、Controller 类以及 Analyser 类；数据通信模块由 Communicator 类组成；图像采集类由 WebcamCapture 类构成；人脸特征点定位模块由 8 个类构成：BoundingBox 类、Detector 类、FaceDetector 类、EyeDetector 类、FaceAligner 类、ShapeRegressor 类、FernCascade 类以及 Fern 类。眨眼检测模块由 2 个类组

成：Analyser 类以及 BlinkAnalyser 类；张嘴检测模块由 2 个类组成：Analyser 类以及 OpenMouthAnalyser 类；摇头检测模块由 3 个类组成：Analyser 类、YawAnalyser 类以及 OpticalFlowCalculator 类。另外 Util 类作为全局工具类，包含整个项目的常用函数以及共享变量。

整体设计上，各个模块之间采用高内聚低耦合的设计理念：不同的类的实例之间的函数调用仅限于用同一模块之间，跨模块的信息传递均采用异步的信号槽机制，隐藏了各个模块内部的实现细节，提升了代码的可维护性。

由于系统对于各主要功能模块均采用独立线程方式运行，因此在设计之初就着重考虑了未来功能模块数量的扩展需求，因此引入简单工厂模式来设计控制器模块与各功能模块调用关系：通过建立一个工厂类，来封装所有的功能模块并且提供实例化接口，每个功能模块均采用一致的启动接口；当一个功能模块需要被调用之时，控制器模块通过工厂类提供的实例化接口获得功能模块的实例化对象，再调用该对象的启动接口。通过应用该设计模式，极大地降低了由于添加功能模块而导致的对于现有系统进行修改的工作量，提高了系统的可扩展性。

由于客户机端的各个模块均需要较高的执行效率，因此在实现过程中选用 C++ 结合支持原生 C++ 语言的 Qt 库；此外，选用 Qt 库开发图形化界面可以在保证界面功能性以及美观性的同时，极大程度上降低开发难度，提升开发速度。

5.1.1 控制器模块设计与实现

5.1.1.1 控制器模块的类结构设计

控制器模块为整个客户机端的核心模块，客户机端的所有其他模块均由控制器负责调度，各个模块的运行结果也汇总至此。控制器模块由 3 个类组成：MainWindow 类、Controller 类以及 Analyser 类。该模块的类图如图 5-2 所示：

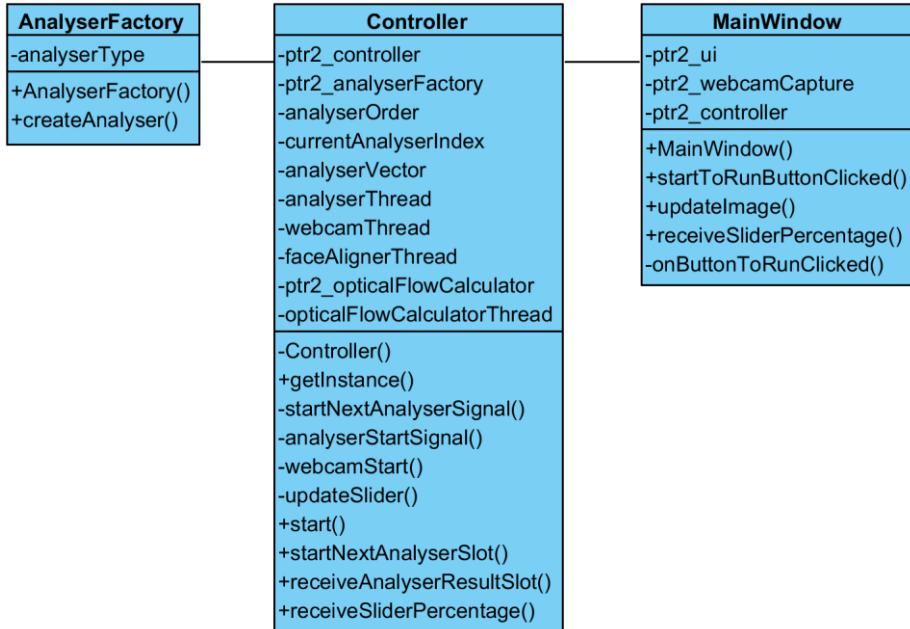


图 5-2 客户机端控制器模块类图

其中，`MainWindow` 类为与用户进行图形界面交互的主窗口类，是客户机端向用户传递信息的主要方式。`Controller` 类为控制器模块的主体，负责响应 `MainWindow` 类交互请求、调用其他功能模块的主线程函数、处理返回结果等功能。`AnalyserFactory` 类为简单工厂模式的工厂类，负责根据不同的请求，实例化对应的 `Analyser` 类。

由于 `Controller` 类采用单例模式设计，因此 `ptr2_controller` 为指向 `Controller` 类的实例化对象的指针；`ptr2_analyserFactory` 为指向生成 `Analyser` 类的实例的工厂类的指针；`analyserOrder` 为通过数据通信模块获取到的随机验证指令序列；`currentAnalyserIndex` 为当前执行的 `Analyser` 类的实例的计数；`analyserVector` 为通过 `AnalyserFactory` 所生成的实例对象的序列；`analyserThread` 为当前执行的功能模块的主线程；`webcamThread` 为图像采集模块的主线程；`faceAlignerThread` 为人脸特征点定位模块的主线程；`ptr2_opticalFlowCalculator` 为指向 `OpticalFlowCalculator` 类的指针；`opticalFlowCalculatorThread` 为异步光流计算的主线程。在接口设计中，`getInstance()`方法为 `Controller` 单例模型的实例化接口；`start()`方法为 `Controller` 主线程的入口函数；`webcamStart()`方法为图像采集模块的主线程的入口函数；`startNextAnalyserSlot()`方法为进入下一轮检测循环的接口，该方法在每个 `Analyser` 实例的主函数执行完毕后被调用；

`receiveAnalyserResultSlot()`方法为各个 Analyser 的实例的运行结果的返回接口；`receiveSliderPercentage()`方法为 Controller 与 MainWindow 中的滑块控件交互的接口。

在 MainWindow 类中，`ptr2_ui` 为指向 UI 配置结构体的指针；`ptr2_webcamCaptureinter` 为指向 WebcamCapture 类的实例化对象的指针；`ptr2_controller` 为指向 Controller 类的实例化对象的指针。在接口设计中，`startToRunButtonClicked()`方法为用户交互按钮状态改变的接口；`updateImage()`方法为更新 UI 中的主视频窗口的接口；`receiveSliderPercentage()`方法为接受 Controller 改变 UI 界面中的控制滑块的参数的接口。

在 AnalyserFactory 类中，`analyserType` 为保存着验证模块信息的序列，每一条记录对应着一个 Analyser 的实例标识；在接口设计中，`createAnalyser()`方法为类静态方法，供实例化特定 Analyser 时调用。

5.1.1.2 控制器模块核心处理流程

Controller 类于客户机端启动伊始便首先实例化。为了防止由控制器模块所控制的其他模块的大量计算所造成的“卡顿”现象，Controller 类的实例以独立线程的方式运行。其调用关系如图 5-3 所示：

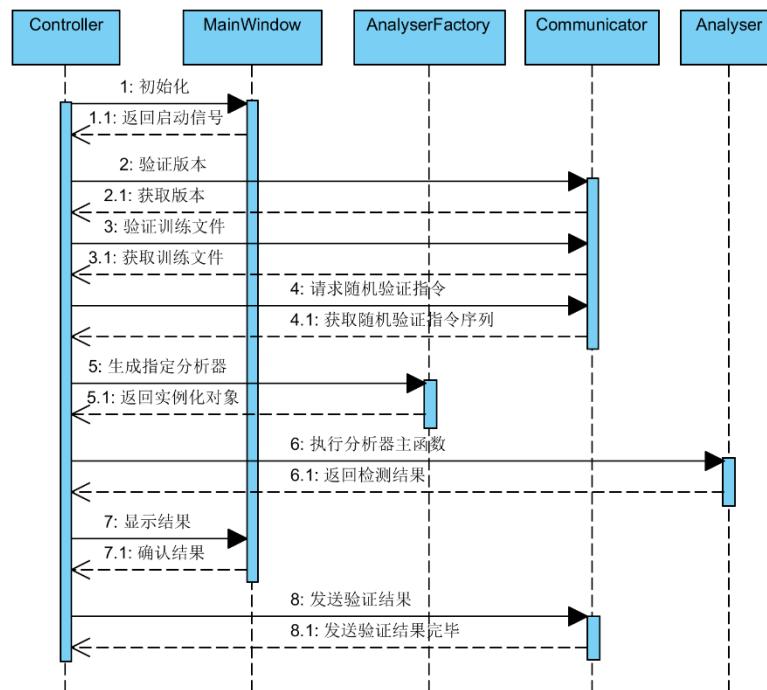


图 5-3 客户机端控制器模块时序图

如图所示，Controller 类是客户机端的入口，在 Controller 类实例化完毕后，MainWindow 类被实例化并且向 Controller 发送启动信号。随后，Controller 调用数据通信模块的 Communicator 进行版本验证、训练文件初始化以及获取随机验证指令。在数据通信模块初始化完毕之后，Controller 实例化 AnalyserFactory，获取各个分析器的实例并且执行分析器的主函数，最后通过 MainWindow 以及 Communicator 返回验证结果。其主函数活动图如图 5-4 所示：

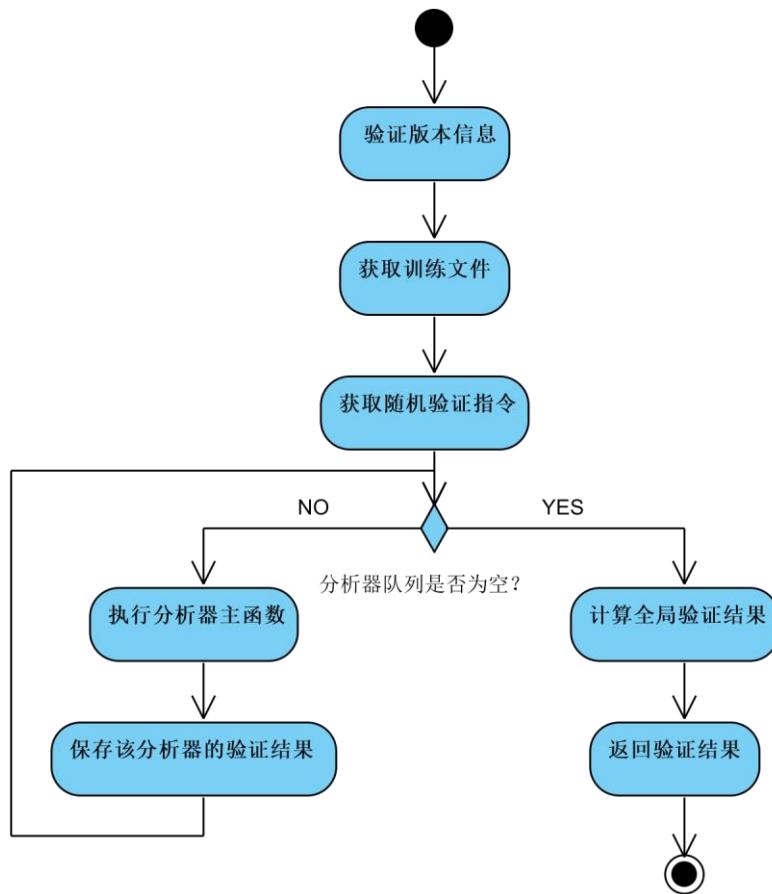


图 5-4 客户机端控制器模块主线程活动图

5.1.2 数据通信模块设计与实现

5.1.2.1 数据通信模块的类结构设计

数据通信模块是客户机端与服务器端通信的接口模块。该模块由 1 个类构成：Comminicator 类。该模块的类图如图 5-5 所示：

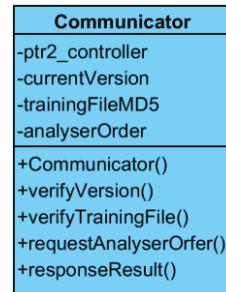


图 5-5 数据通信模块类图

在 Communicator 类中，ptr2_controller 为指向 Controller 类的实例化对象的指针；currentVersion 为目前客户机端的版本号；trainingFileMD5 为目前在客户机端的训练文件的 MD5 散列码；analyserOrder 保存着从服务器端获取的随机验证指令序列。在接口设计中，verifyVersion()方法是与服务器端进行版本验证的接口；verifyTrainingFile()方法为与服务器端进行训练文件一致性检测的接口；requestAnalyserOrder()是从服务器获取随机验证指令序列的接口。

5.1.2.3 数据通信接口设计

出于设计与实现的便捷性考虑，客户机端与服务器端之间的通信采用 Json 作为数据交换的格式。详细接口设计如下：

表 5-1 验证系统版本的 Json 格式

```

{
    "type" : "request",           //消息类型
    "requestType" : "verifyVersion" //请求类型
}

{
    "type" : "response",          //消息类型
    "responseType" : "verifyVersion", //响应类型
    "version" : 12                //版本号
}

```

表 5-2 验证人脸特征点定位的训练文件的 Json 格式

```
{  
    "type" : "request", //消息类型  
    "requestType" : "verifyTrainingFile" //请求类型  
}  
  
{  
    "type" : "response", //消息类型  
    "responseType" : "verifyTrainingFile", //响应类型  
    //服务器端训练文件的 MD5 散列码  
    "trainingFileMD5" : "1c0a9bc79c2ce0dd74373a5912e35a76"  
    //服务器端训练文件的 URL  
    "trainingFileURL" :  
        "http://172.21.8.186:8000/trainingFile"  
}
```

表 5-3 获取随机验证指令序列的 Json 格式

```
{  
    "type" : "request", //消息类型  
    "requestType" : "getAnalyserOrder" //请求类型  
}  
  
{  
    "type" : "response", //消息类型  
    "responseType" : "getAnalyserOrder", //响应类型  
    //作为服务器端标识一个请求的 ID  
    "ID" : 35,  
    "analyserCount" : 3, //随机验证指令的长度  
    //随机验证指令序列  
    "analyserList" : ["yaw","openMouth","blink"]  
}
```

表 5-4 发送验证结果的 Json 格式

```
{
    "type" : "request", //消息类型
    "requestType" : "sendResult", //请求类型
    "ID" : 35, //作为服务器端标识一个请求的 ID
    "result" : true, //验证结果
    //视频截图的 base64 编码
    "base64Image" : "data:image/png;base64,iVBQmCC....."
}
{
    "type" : "response", //消息类型
    "responseType" : "sendResult", //响应类型
    "ID" : 35, //作为服务器端标识一个请求的 ID
    "flag" : "received"
}
```

其中，在由于在验证人脸特征点定位的训练文件阶段，为了保证结果的准确性，需要人脸特征点定位算法所使用的训练文件的一致性，因此作为一种简便快速的验证方法，引入了 MD5 散列码来验证服务器端的训练文件与客户机端的训练文件的一致性。

在获取随机验证指令序列阶段，可能会出现多个客户机端同时从服务器端获取随机验证指令序列的情况，因此在 Json 消息中，设置了 ID 字段来区分不同的请求，保证服务器端可以追踪所有请求的状态。

在发送验证结果阶段，客户机端需要将一张视频截图连带验证结果一起发送至服务器端。出于统一 IO 接口以及节约服务器系统资源的角度考虑，采用 Base64 算法将图片编码后，也通过 Json 消息进行发送；服务器端在收到该 Json 消息后，通过将 Base64 的编码字段进行解码，即可得到原图像。

5.1.2.2 数据通信模块核心处理流程

Communicator 类由客户机端的控制器模块的 Controller 负责实例化，与服务器端进行数据通信。该模块的调用关系如图 5-6 所示：

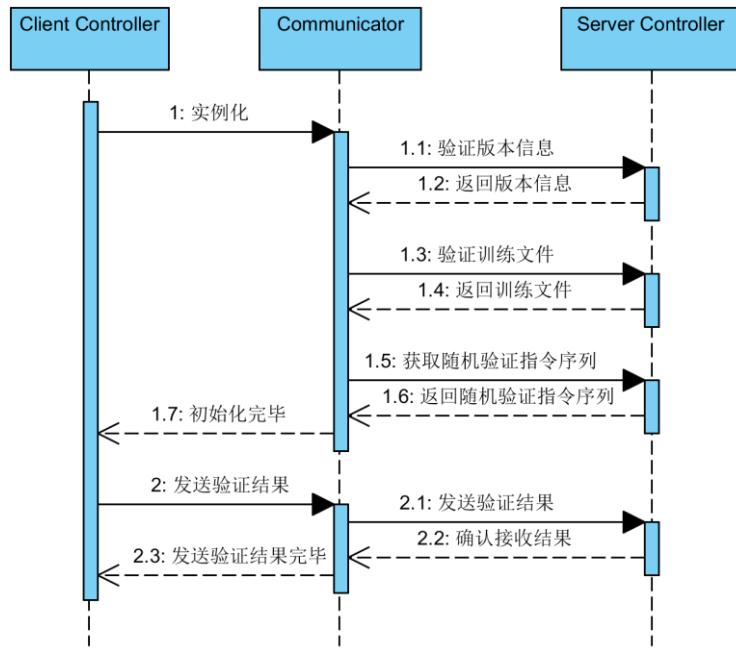


图 5-6 数据通信模块时序图

在实例化完成后的初始化阶段，该模块会与服务器端进行同步通信，并且完成系统的参数初始化，最后将控制权交回给控制器模块。在发送验证结果阶段，该模块接受从控制器模块传来的参数并将其发送至服务器端，完成数据的传输。其活动图如图 5-7 所示：

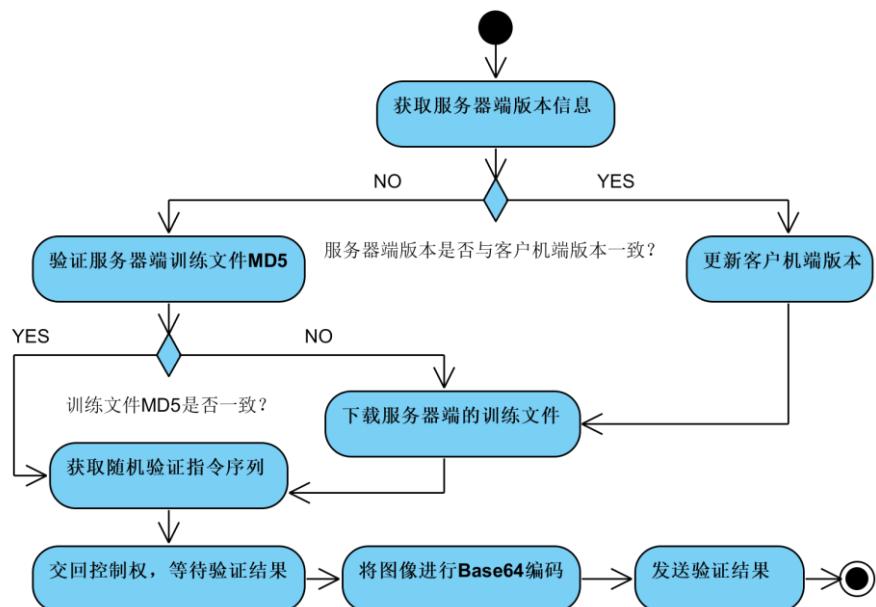


图 5-7 数据通信模块活动图

5.1.3 图像采集模块设计与实现

5.1.3.1 图像采集模块的类结构设计

图像采集模块负责从摄像头采集序列帧，并将这些序列帧保存下来。该模块提供了不同粒度的接口，供其他模块根据各自的需求进行调用。图像采集模块由 1 个类组成：WebcamCapture 类。该模块类图如图 5-8 所示：

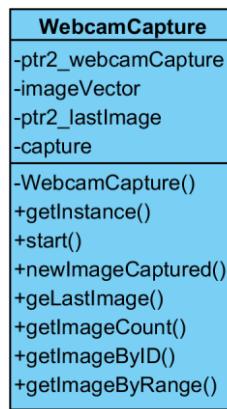


图 5-8 图像采集模块类图

在 WebcamCapture 类中，`ptr2_webcamCapture` 为指向 WebcamCapture 的实例化对象的指针；`imageVector` 为存储图像序列的数组；`ptr2_lastImage` 为指向最新图片的指针；`capture` 为 opencv 的图像采集模块的实例。在接口设计方面，`getInstance()`方法为 WebcamCapture 单例模式的实例化方法；`start()`方法为该模块的主线程入口函数；`newImageCapture()`方法为捕获到新的图片的调用接口；`getLastImage()`方法为获取最新图像的接口；`getImageCount()`方法为获取图像序列数组的容量的接口；`getImageByID()`为根据图像 ID 获取图像的接口；`getImageByRange()`方法为通过一个范围来批量获取图片的接口。

5.1.3.2 图像采集模块核心处理流程

WebcamCaptrue 类由 Controller 对象负责实例化，该类的实例化对象运行在独立线程空间中，其主线程函数为无限循环，负责完成图像的捕获以及保存。在捕获图像的同时，提供中断事件接口供其他模块在获取图像时调用。该模块的时序图如图 5-9 所示：

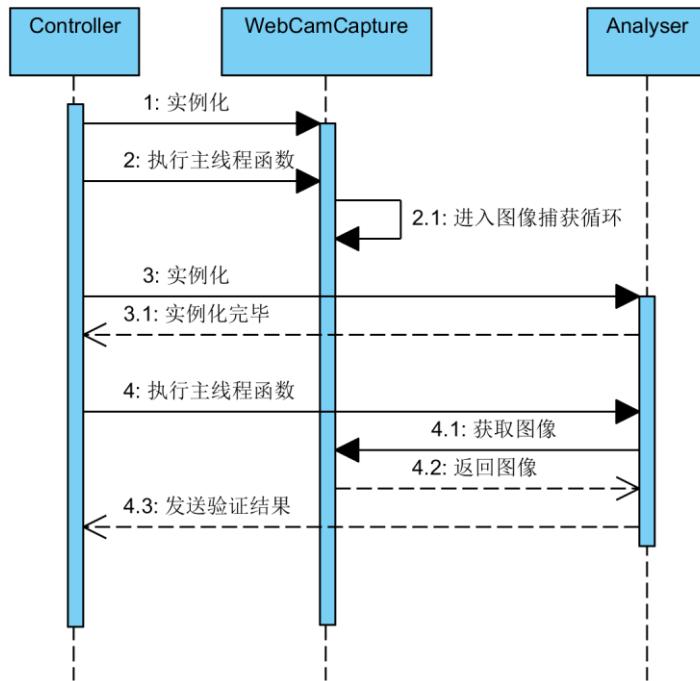


图 5-9 图像采集模块时序图

在图像采集的过程中，WebcamCapture 会持续地采集图像并且存入预先设置好的数组中，并且更新数据指针。在每次迭代间隙检查请求队列，响应其中的队列中的获取图像的请求。其活动图如图 5-10 所示：

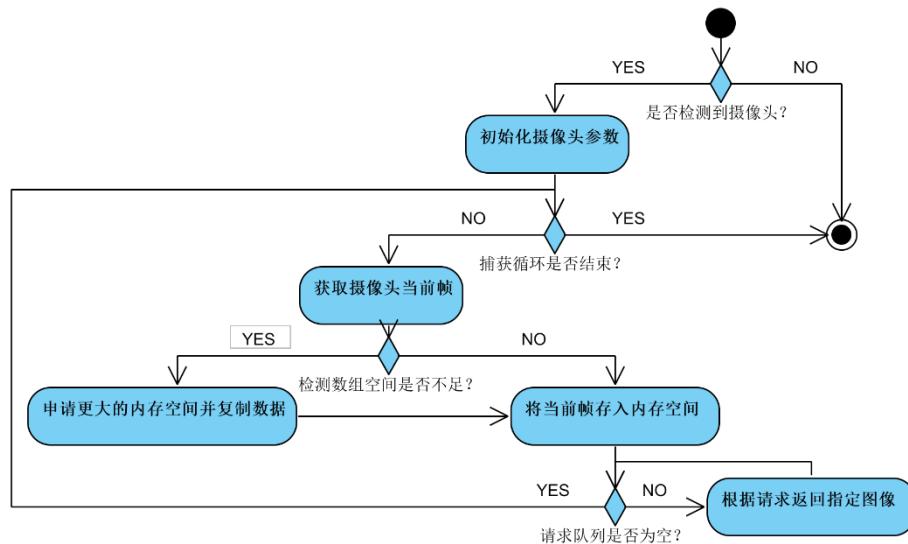


图 5-10 图像采集模块活动图

5.1.4 人脸检测模块设计与实现

该模块向眨眼检测模块、张嘴检测模块以及摇头检测模块开放调用接口，

5.1.4.1 人脸检测模块的类结构设计

人脸检测模块负责进行输入图像的人脸检测以及人眼检测的功能。该模块由 4 个类构成，其类图如图 5-11 所示：

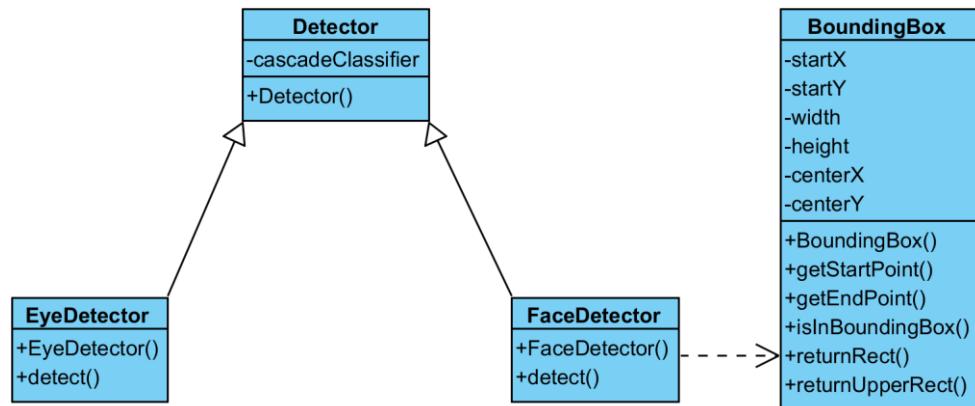


图 5-11 人脸检测模块类图

其中，**Detector** 类为抽离出检测动作的抽象类，该类作为基类，可以派生出不同的派生类以执行特定的功能。**EyeDetector** 类为 **Detector** 类的其中一个派生类，负责对于输入的图像进行人眼检测，即接受一幅图像作为输入，返回识别为人眼的位置的包围盒的序列。**FaceDetector** 类为 **Detector** 类的另一个派生类，负责对于输入图像进行人脸检测，即接受一幅图像作为输入，以包围盒的形式返回图像中人脸的位置。**BoundingBox** 类为自定义数据结构，用于表示图像中的一个包围盒。

在 **Detector** 类中，`cascadeClassifier` 为分类器所需要使用的模型文件。在接口设计中，`detect()`方法设计为纯虚函数，作为派生类的入口函数接口。

EyeDetector 类继承自 **Detector** 类，在该类中，`detect()`方法实现了 **Detector** 基类中的 `detect()`纯虚函数接口，其功能为在一幅图像的指定区域中，搜索符合人眼特征的区域，并且将所有的符合条件的区域以包围盒序列的形式返回。

FaceDetector 类也继承自 **Detector** 类，在该类中，`detect()`方法实现了 **Detector** 基类中的 `detect()`纯虚函数接口，其功能为在一幅图像中搜索最符合人脸特征的区域，并且将该区域以包围盒的形式返回。

5.1.4.2 人脸检测模块核心处理流程

人脸检测模块的主要功能为检测一幅图像中的人脸或者人眼的区域，由于人眼检测较为敏感，因此容易受到外界的光照、背景以及噪声条件的影响。因此在进行人眼检测之前，系统会先进行人脸检测，将人脸检测的结果作为人眼检测的粗定位，再对目标区域进行人眼检测，以提高人眼检测算法的稳定性。其调用关系如图 5-12 所示：

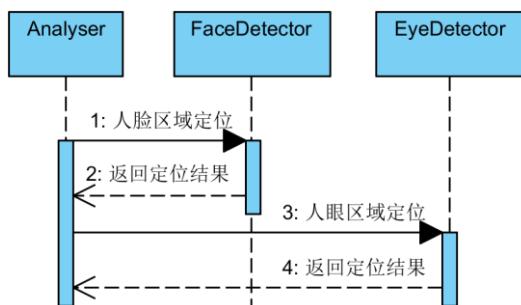


图 5-12 人脸检测模块时序图

人脸检测以及人眼检测采用的是 opencv 所提供的 Ada Boost 接口实现，选取的特征为哈尔特征。对于每一幅输入图像，Analyser 的实例化对象都会调用 FaceDetector 以及 EyeDetector 进行人脸区域以及人眼区域的检测。其活动图如图 5-13 所示：

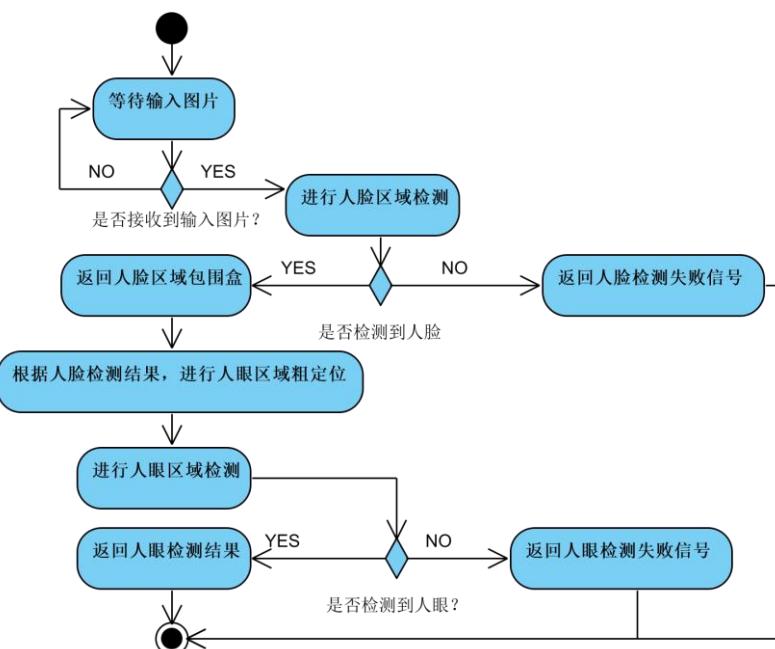


图 5-13 人脸检测模块活动图

5.1.5 人脸特征点定位模块设计与实现

人脸特征点定位模块是人脸活体检测系统的重要模块之一，该模块主要负责针对输入的人脸图片，检测其中的人脸特征点的坐标。在具体的定位算法上，该模块采用 Xudong Cao 于 2014 年提出的显式形状回归算法进行人脸特征点定位。本小节将会首先阐述回归算法所采用的特征，随后再提出级联式回归器的分层结构，最后再描述该模块的类结构设计以及核心处理流程。

5.1.5.1 形状索引特征的设计

形状索引特征（Shape Indexed Feature）是用来描述输入图片以及当前测定的特征点序列相对于真实的特征点序列的残差的一种特征。其本质是像素差异特征（Pixel Difference Feature）的一种扩展。

像素差异特征的选取规则为在图像的全局坐标系上选取 N 个像素对，并且计算各个像素对的强度差，这些强度差的序列即为。由于像素差异特征可以通过迭代的方式获取非常充足的信息，并且只占用较少的 CPU 计算资源，因此被广泛地应用于提取图像的语义特征。不过，由于像素差异特征的像素对的选取是基于全局坐标系的，即每对像素对的绝对坐标一经选取，便不会改变；对于变化比较剧烈的人脸图像而言，同一个坐标点的语义信息会有较大的不同，由此导致噪声的增加，影响像素差异特征的准确性。

形状索引特征在像素差异特征的基础上，采用相对坐标系来确定像素点对的位置：即首先在全局坐标系中生成 N 个像素对，然后对于每个像素点，分别记录与之 L2 距离最小的特征点的编号以及该像素点相对于该特征点的偏移，由此作为确定该像素点的依据。由于引进了相对于最邻近的特征点相对坐标系，有效提高了所选取的像素点的语义一致性，降低了该特征的噪声。

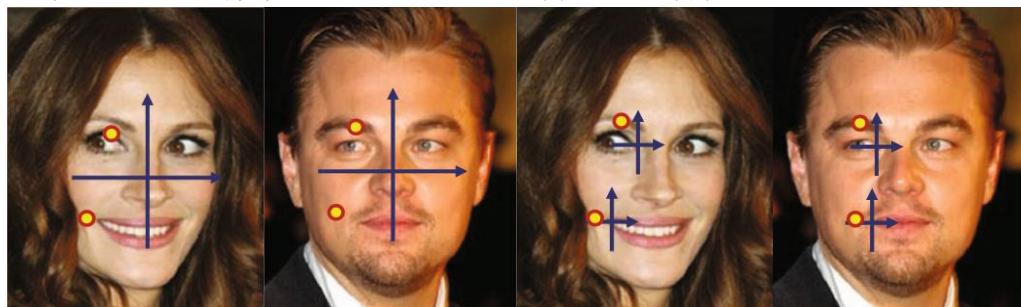


图 5-14 形状索引特征示意图

如图 5-14 所示，第一张以及第二张照片采用的是传统的像素差异特征，第三张以及第四张照片采用的是形状索引特征。可见，在像素差异特征中，同一对点在不同的输入图片中的语义信息并不一致，而在形状索引特征中，同一对点在不同的输入图片中的语义信息基本是一致的。因此，采用形状索引特征作为输入图像的描述特征可以有效提高人脸特征点定位算法对于不同的输入图片的鲁棒性以及稳定性。

5.1.5.2 级联回归器的设计

在人脸特征点定位模块中，负责调整特征点位置的算法为显式形状回归算法。该算法依赖于一个两层结构的级联式形状回归框架：在该框架中，回归器在逻辑上被分为内外两层，外层的回归器负责生成内层回归器所需要的计算形状索引特征的像素对；内层回归器为典型的随机森林回归器，接受前一个内层回归器输出的特征点位置，使用外层回归器所指定的像素对计算形状索引特征并且修正这些特征点的位置。

对于外层回归器而言，其与相邻的外层回归器形成链式结构，每个外层回归器的输出即为下一个外层回归器的输入，以此类推直至到达级联回归器的结尾。每一个外层回归器都会对前一个回归器所输出的特征点的坐标的序列进行修正，减少其误差，并且输出修正之后的特征点坐标的序列。外层回归器之间的连接关系如图 5-15 所示：

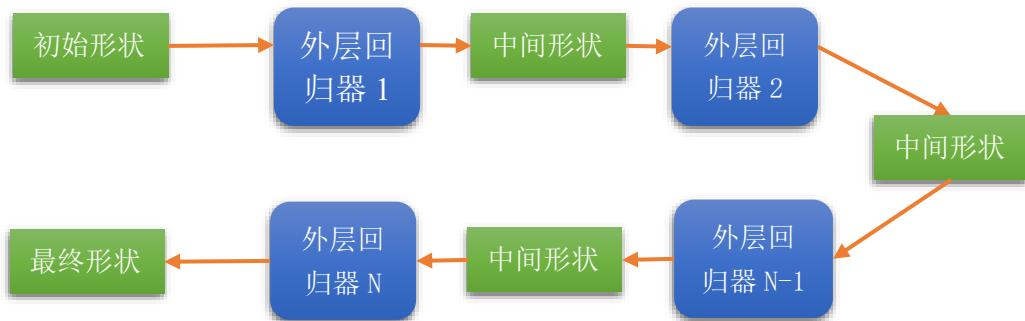


图 5-15 外层回归器连接关系图

外层回归器并不直接调整特征点的坐标的序列，而是迭代地调用内层回归器对特征点坐标的序列进行修正，最后将所有包含的内层回归器的结果汇总后，向下一个外层回归器输出。在每个外层回归器中，包含了 M 个内层回归器（在本系统中，每个外层回归器包含了 50 个内层回归器），这些内层回归器也是于相

邻的内层回归器进行链式的连接。每个内层回归器接收前一个内层回归器所输出的特征点坐标的序列以及外层回归器所定义的像素点对，通过计算形状索引特征来讲每个输入映射到一个输出集中，将该输出集的加权和作为整个内层回归器的输出并且修正特征点坐标的序列。内层回归器之间的连接关系如图 5-16 所示：

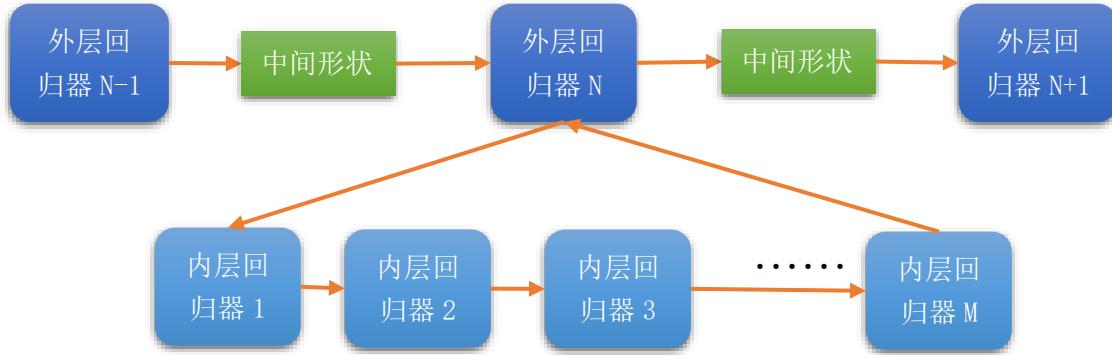


图 5-16 内层回归器连接关系图

本算法采用的基础回归器为基于形状索引特征的随机森林回归器，在训练阶段，随机森林回归器接受 3 个参数，分别为输入形状、真实形状以及形状索引特征。通过计算得到输入形状与真实形状的残差，再使用随机阈值对形状索引特征与回归器的输出集合进行映射，最后将该残差添加到该输入集合中，完成一次训练迭代。在测试阶段，随机森林回归器接收 2 个参数，分别为输入形状以及形状索引特征，通过随机阈值定位该形状索引特征对应的输出集合，将该集合中的残差的加权和与当前形状叠加，并将更新之后的形状输出，完成一次测试迭代。这种回归器的运算速度较快但是其回归能力较弱，每次迭代后，只可以对特征点坐标的序列进行小幅度的修正。因此，在本系统中采用级联式的回归框架对各个回归器进行连接。

因为每次重新生成新的形状索引特征并且计算相应的计算参数会影响该模块响应的实时性，所以在实现过程中，采用两层结构的级联式的回归框架，外层回归器负责更新形状索引特征所需要的像素点对，而内层回归器不需要重新生成这些像素对，可以直接使用外层回归器所生成的像素对计算形状索引特征并且快速输出结果。这种框架的优点为：可以在保证回归精度的情况下，尽可能地降低运算量，提升运算速度。

5.1.5.3 人脸特征点定位模块的类结构设计

人脸特征点定位模块主要负责对于输入的人脸图片，计算其各个特征点的坐标。该模块由 4 个类构成： FaceAligner 类、 FernCascade 类、 Fern 类、 ShapeRegressor 类。其类图如图 5-17 所示：

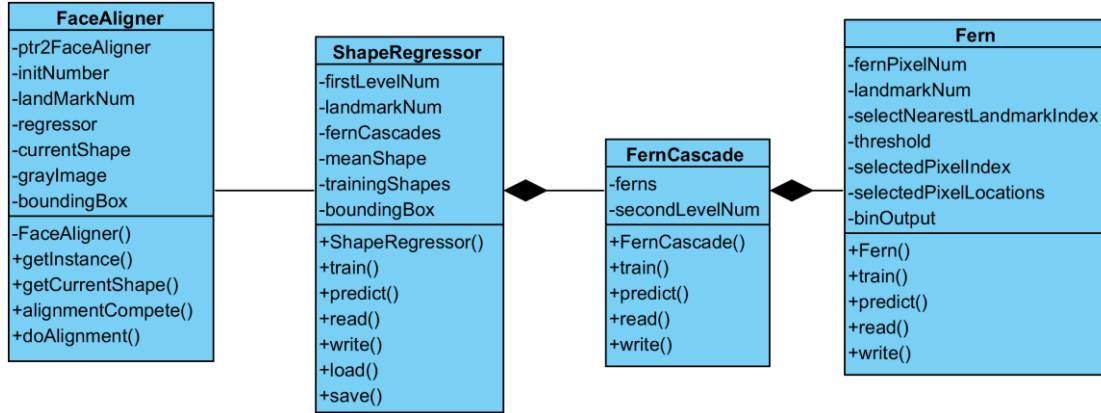


图 5-17 人脸特征点定位模块类图

其中， FaceAligner 类为人脸特征点定位模块的接口类，该类主要负责人脸特征点定位模块与其他模块的通信以及相关数据、用户设定的参数的保存。 ShapeRegression 类为级联回归器的实现，负责对输入的人脸图像进行人脸特征点定位。 FernCascade 类为级联回归器的外层框架的元素，负责生成相应的随机蕨特征，并且对上一级的回归结果进行修正。 Fern 类为级联回归器的内层框架元素，是一个典型的随机蕨回归器，负责从外层回归器获取随机蕨特征，并且对上一级的回归结果进行修正。

在 FaceAligner 类中， ptr2FaceAligner 为单例模式中指向该类的实例化对象的指针； initNumber 为进行初始化数据扩增的数量； landMarkNum 为每张训练图或测试图中，人脸特征点的数量； regressor 为 ShapeRegressor 类的实例化对象； currentShape 为当前计算生成的人脸特征点坐标的序列； grayImage 为当前正在进行人脸特征点定位的输入图像； boundingBox 为输入图像 grayImage 中的描述人脸位置的包围盒对象。在接口设计中， getInstance() 方法为单例模式的实例化接口函数； getCurrentShape() 方法为获取当前的人脸特征点坐标的序列的函数； alignmentCompete() 方法为人脸特征点定位完成的信号接口； doAlignment() 方法为执行人脸特征点定位算法的接口函数。

在 ShapeRegressor 类中， firstLevelNum 为级联回归器中外层（第一层）回归

器的数量; `landmarkNum` 为训练图或者测试图中人脸特征点的数量; `fernCascades` 为级联回归器中外层回归器的序列; `meanShape` 为所有训练图中的人脸特征点序列的均值; `trainingShaps` 为训练图中的人脸特征点序列的集合; `boundingBox` 为目前正在对人脸特征点定位的图像的描述人脸位置包围盒对象。在接口设计中, `train()`方法为训练完整的级联回归器函数; `predict()`方法为对一幅输入图像进行人脸特征点定位的函数; `read()`方法为读取训练文件的参数的函数; `write()`方法为将参数写回训练文件的函数; `load()`方法为根据训练文件的参数初始化级联回归器的函数; `save()`方法为根据当前级联回归器的状态记录参数的函数。

在 `FernCascade` 类中, `ferns` 为级联回归器中的内层(第二层)回归器的序列; `secondLevelNum` 为级联回归器中内层回归器的数量。在接口设计方面, `train()`方法为训练该外层回归器的函数; `predict()`方法为人脸特征点定位函数, 该方法接受一幅输入图像以及一组当前的特征点位置的序列, 通过随机蕨特征对当前的特征点位置的序列进行修正, 减少误差; `read()`方法为读取级联回归器的训练文件的参数并初始化 `FernCascade` 类的实例化对象; `write()`方法为将当前该类的实例化对象的参数保存并且写入训练文件。

在 `Fern` 类中, `fernPixelNum` 为选取的随机蕨特征的像素对的数量; `landmarkNum` 为人脸特征点的坐标的数量; `selectNearestLandmarkIndex` 为当前检测的该特征像素的最邻近的特征点的编号, 用以计算形状索引特征 (Shape Indexed Feature); `threshold` 为将像素差异特征 (Pixel Difference Feature) 转化为随机蕨回归器的输出类别的阈值; `selectedPixelIndex` 为当前正在计算的像素点的编号; `selectPixelLocations` 为当前正在计算的像素点的位置; `binOutput` 为当前该随机蕨回归器的输出结果的序列。在接口设计方面, `train()`方法为通过一系列的人脸训练图像以及对应的人脸特征点序列的参数来训练该随机蕨回归器的函数; `predict()`方法为从一张输入图片以及对应的人脸特征点序列中, 计算出该特征点的与真实人脸特征点序列的残差, 并且修正该特征点序列的函数; `read()`方法为读取训练文件的参数设置并且初始化该随机蕨回归器的函数; `write()`方法为保存本随机蕨回归器的所有参数至训练文件的函数。

5.1.5.4 人脸特征点定位模块核心处理流程

人脸特征点定位模块的核心算法是基于随机蕨的人脸特征点定位算法, 在本小节将分别就训练阶段以及测试阶段的算法流程进行描述。

人脸特征点定位模块为其他功能模块提供人脸特征点定位的接口，由于该模块实例化之后的对象数量多，因此初始化时，采用自顶向下的迭代初始化的方式：外层回归器先于内层回归器初始化，随后外层回归器再将其下属的内层回归器进行初始化。在训练阶段，由 FaceAligner 的实例化对象调用 shapeRegressor 的训练接口，后者开始以层级关系的顺序训练各个外层回归器以及内层回归器。在训练阶段，由 Analyser 的实例化对象调用 FaceAligner 的人脸特征点定位接口，由 ShapeRegressor 以层级关系的顺序依次调用各个外层回归器以及内层回归器，最后返回经过所有回归器修正的人脸特征点坐标的序列。该模块调用关系如图 5-18 所示：

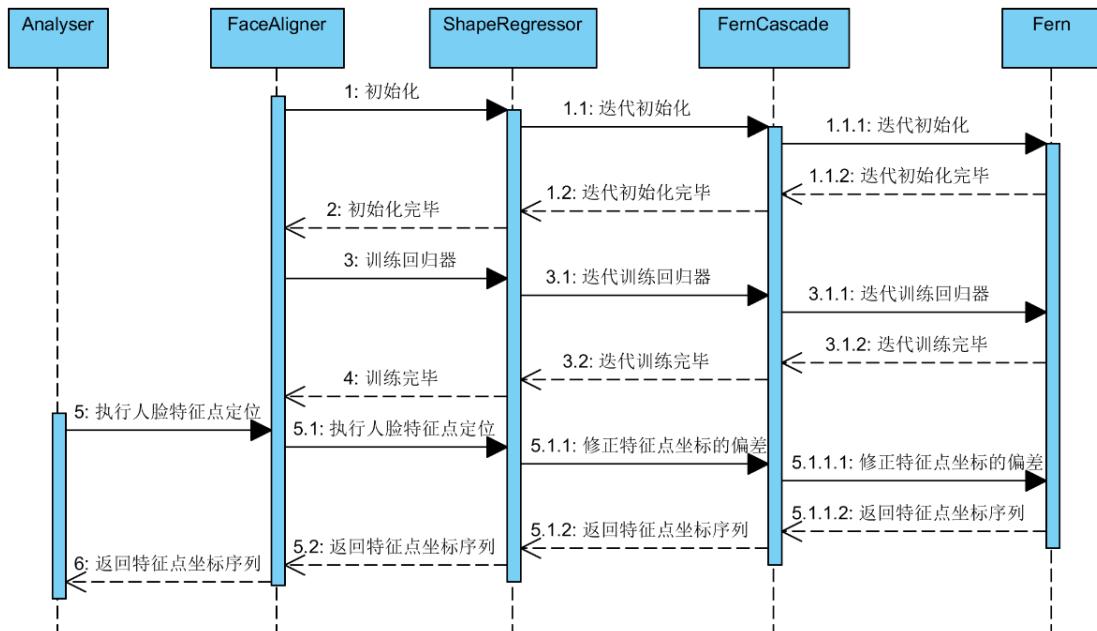


图 5-18 人脸特征点定位模块时序图

在级联回归器的训练阶段，对于每个外层回归器而言，首先对所有的训练图片进行数据扩增，随后遍历扩增后的训练数据库，取出每一张图片并生成一组用于计算形状回归特征的像素对的序列，最后把图片以及像素对的序列传给每一个内层回归器用于训练。对于每个内层回归器而言，首先使用接收到的像素对的序列计算形状索引特征，对于形状索引特征的每一个维度都使用一个随机生成的阈值进行区间划分，通过比较形状索引特征的每一个维度的值与随机阈值的关系可以将该训练图片以及对应的特征点坐标的序列映射至一个输出集合中。当训练库中所有的数据都被映射至某一个输出集合后，再计算经过数据扩增后

的训练库的形状的均值并且保存。至此，一个完整的训练过程结束。级联回归器的训练阶段活动图如图 5-19 所示：

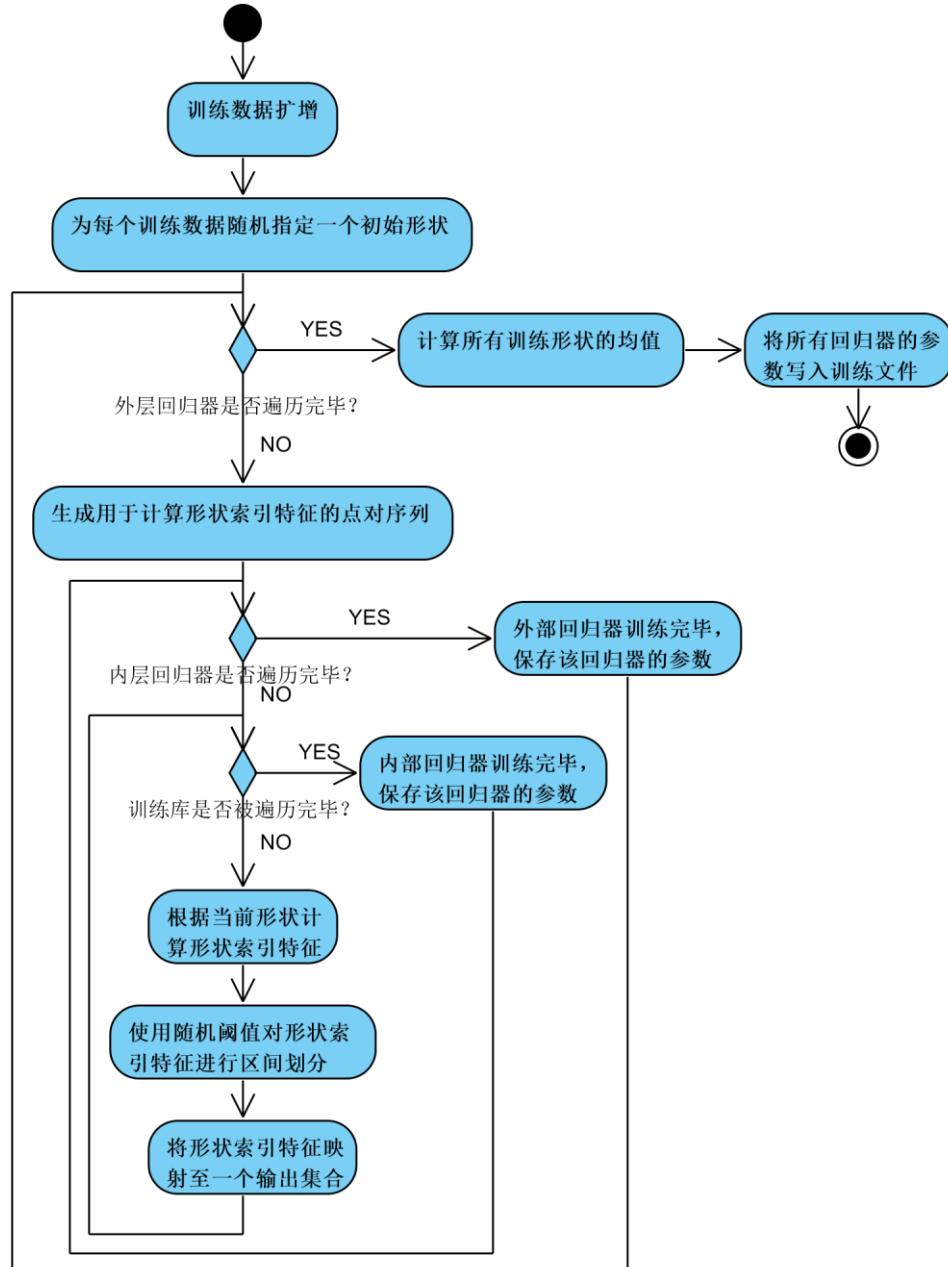


图 5-19 人脸特征点定位模块训练阶段活动图

在级联回归器的测试阶段，首先从训练文件中提取出训练数据库的形状的均值，使用该均值形状作为原始的人脸特征点坐标的序列。对于每个外层回归器而言，接收一组当前的人脸特征点坐标的序列，读取训练文件中的用于计算形状

索引特征的像素对的序列并通过下属的各个内层回归器不断修正当前形状中的偏差，最后输出的形状将作为下一个外层回归器的形状输入。对于每个内层回归器而言，首先接收用于计算形状索引特征的像素对的序列并且计算形状索引特征，然后使用训练文件中的划分阈值将形状索引特征映射只一个输出集合中，最后计算输出集合中的形状残差加权和，并且使用该加权和去修正当前的输入形状。当该输入图像经过所有的回归器修正后，当前的人脸特征点坐标的序列即为最终的输出结果。级联回归器的测试阶段活动图如图 5-20 所示：

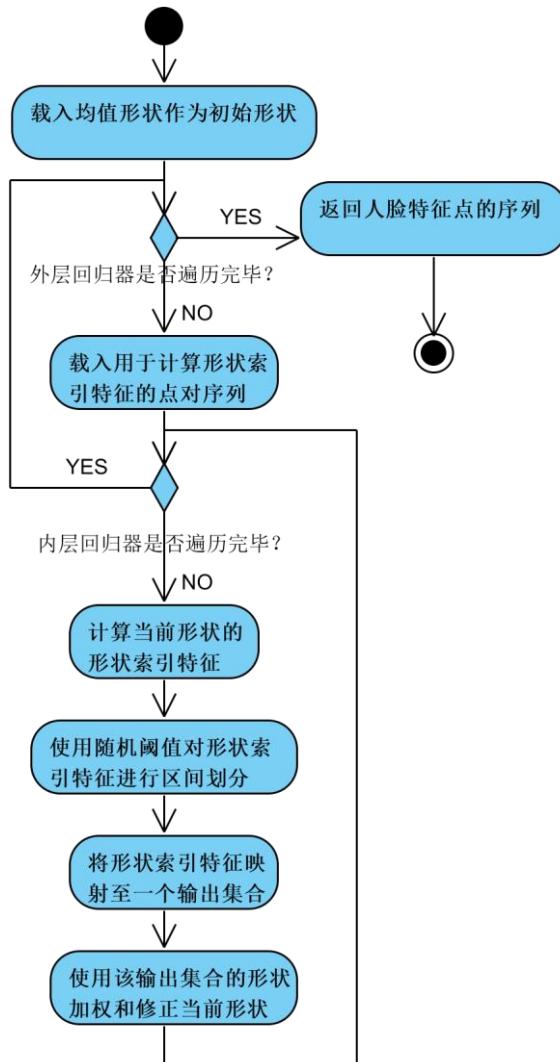


图 5-20 人脸特征点定位模块测试阶段活动图

5.1.6 眨眼检测模块设计与实现

5.1.6.1 眨眼检测模块的类结构设计

眨眼检测模块为客户机端的具体检测模块之一，该模块负责检测用户是否进行有效的眨眼动作。眨眼检测模块由2个类组成：Analyser类以及BlinkAnalyser类，该模块类图如图5-21所示：

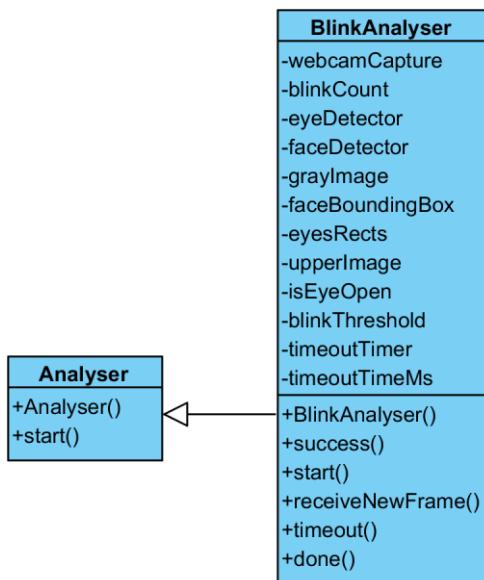


图 5-21 眨眼检测模块类图

其中，Analyser类为系统的具体检测模块的抽象基类，可以派生出各个具体的检测类。BlinkAnalyser类继承自Analyser基类，负责检测用户是否按照系统指示进行了眨眼动作。

在Analyser类中，start()方法是纯虚接口，为各个具体的检测模块的入口函数接口。

在BlinkAnalyser类中，webcamCapture为图像采集模块的实例化对象，用于获取用户的输入图像；blinkCount为当前检测到的眨眼次数；eyeDetector为指向EyeDetector类的实例化对象的指针，用于检测图像中眼睛的位置；faceDetector为只想faceDetector类的实例化对象的指针，用于检测图像中人脸的位置；grayImage为当前从图像采集模块获取到的用户输入图像；faceBoundingBox为标识图像中人脸位置的包围盒；eyesRects为标识图像中人眼位置的包围盒序列；upperImage为经过裁剪的输入图像，用于增强检测精度；isEyeOpen为标识眼睛

是否睁开的标志位； blinkThreshold 为预设的表示检测成功的眨眼次数阈值； timeoutTimer 为超时计时器，用于控制模块的超时逻辑的触发； timeoutTimeMs 为超时计时器的触发阈值。在接口设计方面，success()方法为眨眼检测通过的函数接口； start()方法为眨眼检测模块主函数入口； receiveNewFrame() 为从图像采集模块获取到新图像的槽函数()； timeout()方法为超时响应函数； done() 为检测结束的信号函数，在眨眼检测模块检测结束后调用，用于发送检测结果。

5.1.6.2 眨眼检测模块核心处理流程

BlinkAnalyser 类由 Controller 对象负责实例化，随后 Controller 对象将 BlinkAnalyser 对象加入 WebcamCapture 的事件循环，再调用 BlinkAnalyser 的入口函数；当收到 BlinkAnalyser 的检测结果以及检测结束的信号后，将 BlinkAnalyser 从 WebcamCapture 的事件循环移除。该模块的时序图如图 5-22 所示：

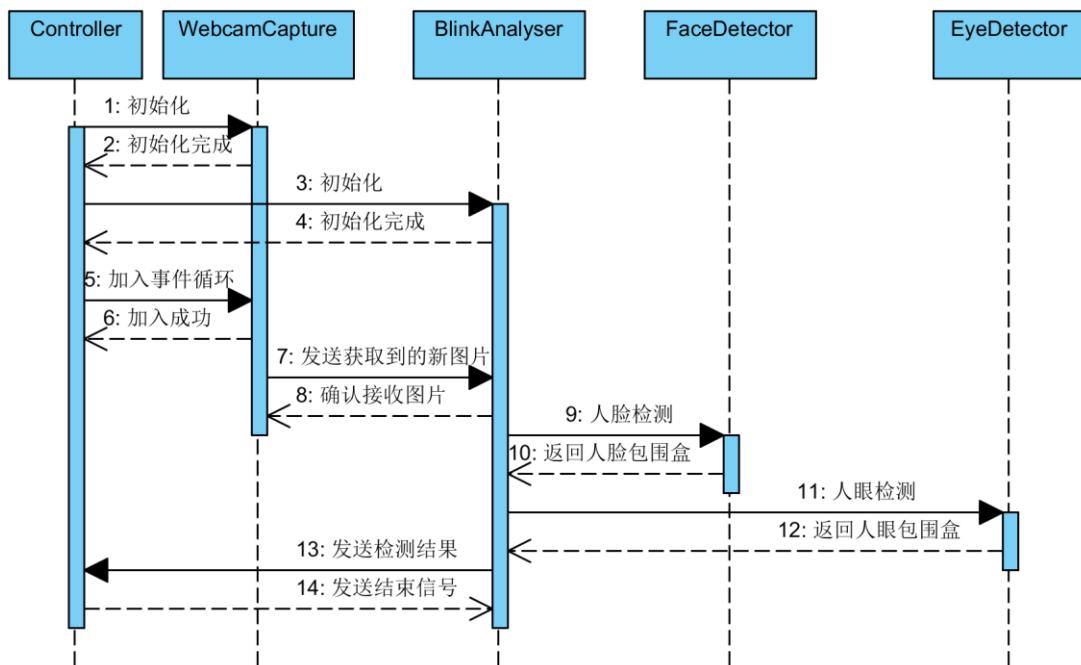


图 5-22 眨眼检测模块时序图

在眨眼检测的过程中，会调用 OpenCV 的哈尔特征分类器进行人脸检测以及人眼检测，由于人眼检测所使用的检测窗口较小，较容易受到图像中的噪声干扰，因此在进行人眼检测前，首先进行人眼位置的粗定位：即通过人脸检测确定

人脸的大致位置，再通过人类眼部位置的先验知识，使用人脸包围盒的上半部分进行人眼检测。在人眼检测之前首先进行人眼位置粗定位可以有效提高人眼检测的正确率，降低图像中的噪声对于检测结果的影响。

检测眨眼次数的功能是通过设计一个状态机实现的：在接收到图像采集模块所发送的新图片后，就人眼检测，通过比对与前一帧的状态差来判断用户是否进行了眨眼动作，如果相邻的两帧的人眼检测状态发生了从 false 到 true 的跳变，则记录用户进行了一次眨眼动作。

当眨眼次数达到预先设定的阈值时，或者在触发超时状态后，都会结束程序的事件循环并且退出。眨眼检测模块的活动图如图 5-23 所示：

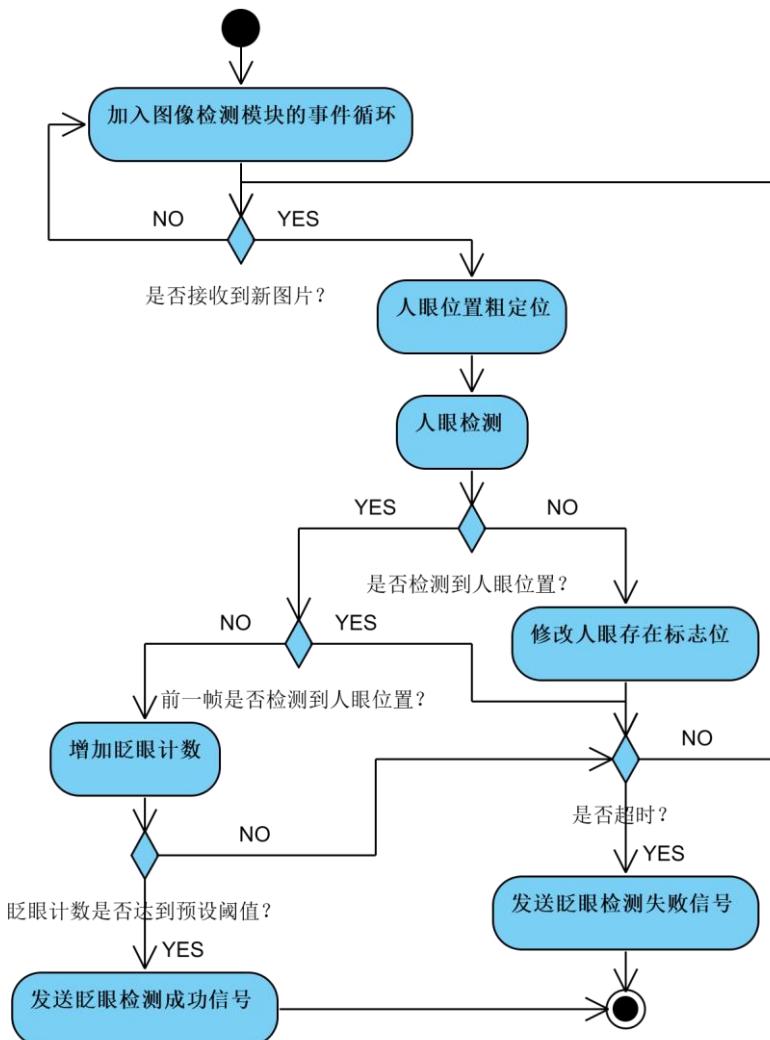


图 5-23 眨眼检测模块活动图

5.1.7 张嘴检测模块设计与实现

5.1.7.1 张嘴检测模块的类结构设计

张嘴检测模块也为客户机端具有具体检测功能的模块之一，该模块负责检测用户是否按照指示进行了有效的张嘴的动作。张嘴检测模块由 2 个类组成：Analyser 类以及 OpenMouthAnalyser 类，该模块的类图如图 5-24 所示：

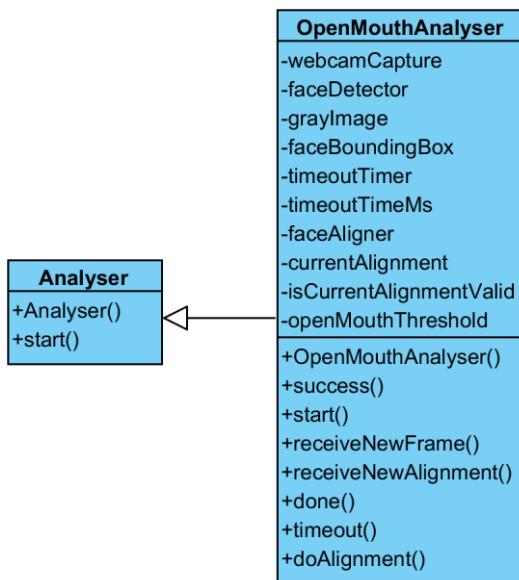


图 5-24 张嘴检测模块类图

其中，Analyser 类为系统的具体检测模块的抽象基类，可以派生出各个具体的检测类。OpenMouthAnalyser 类继承自 Analyser 基类，负责监测用户是否按照系统的引导进行了张嘴动作。

在 Analyser 类中，`start()`方法是纯虚接口，为各个具体的检测模块的入口函数接口。

在 OpenMouthAnalyser 类中，`webcamCapture` 为图像采集模块的实例化对象；`faceDetector` 为人脸检测模块的实例化对象；`grayImage` 为当前检测的用户输入图片；`faceBoundingBox` 为标识图片中人脸位置的包围盒对象；`timeoutTimer` 为超时计时器，负责在超时后激活对应的处理函数；`timeoutTimeMs` 为预设的超时计时器的触发阈值；`faceAligner` 为人脸特征点检测模块的实例化对象，`currentAlignment` 存储着人脸特征点检测模块所返回的人脸特征点坐标的序列；`isCurrentAlignmentValid` 为表示人脸特征点定位结果有效性的标志位；

`openMouthThreshold` 为预设的张嘴程度的阈值。在接口设计方面，`success()`方法为张嘴检测成功的响应函数；`start()`方法为张嘴检测模块的入口函数；`receiveNewFrame()`方法为从图像采集模块获取到新图像的响应函数；`receiveNewAlignment()`方法为从人脸特征点定位模块接收到人脸特征点坐标序列的响应函数；`done()`方法为张嘴检测结束时所调用的函数；`timeout()`方法为超时计时器触发超时逻辑的响应函数；`doAlignment()`方法为调用人脸特征点定位模块的接口函数。

5.1.7.2 张嘴检测模块核心处理流程

`OpenMouthAnalyser` 类由 `Controller` 对象通过工厂类进行实例化，在完成实例化后，随即将该对象加入图像采集模块的事件响应队列。当 `OpenMouthAnalyser` 的实例化对象接收到由图像采集模块所发送的新图像后，即调用人脸检测模块以及人脸特征点定位模块的相关函数，通过人脸特征点坐标的序列来判断用户是否完成了有效的张嘴动作。该模块的时序图如图 5-25 所示：

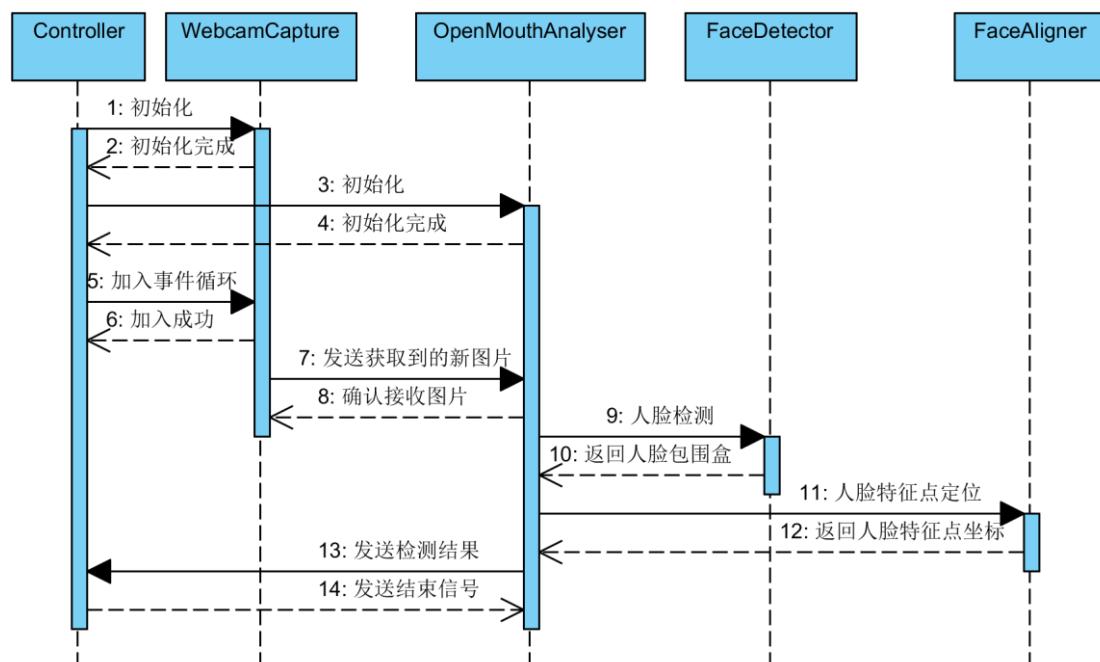


图 5-25 张嘴检测模块时序图

在张嘴检测的过程中，该模块会首先调用 OpenCV 提供的哈尔特征分类器进行人脸检测，随后使用人脸特征点定位模块确定各个人脸特征点的位置，并且

通过比对嘴部的特征点的位置关系来衡量用户的张嘴动作是否达到预设要求。由于用户距离摄像头的距离具有不确定性，因此张嘴检测模块采用特征点之间的相对距离来计算用户的张嘴程度，而非图像中的绝对距离：实际实现中，所采用的相对距离为嘴部张开的距离与嘴唇厚度的比值。该模块进行张嘴检测的活动图如图 5-26 所示：

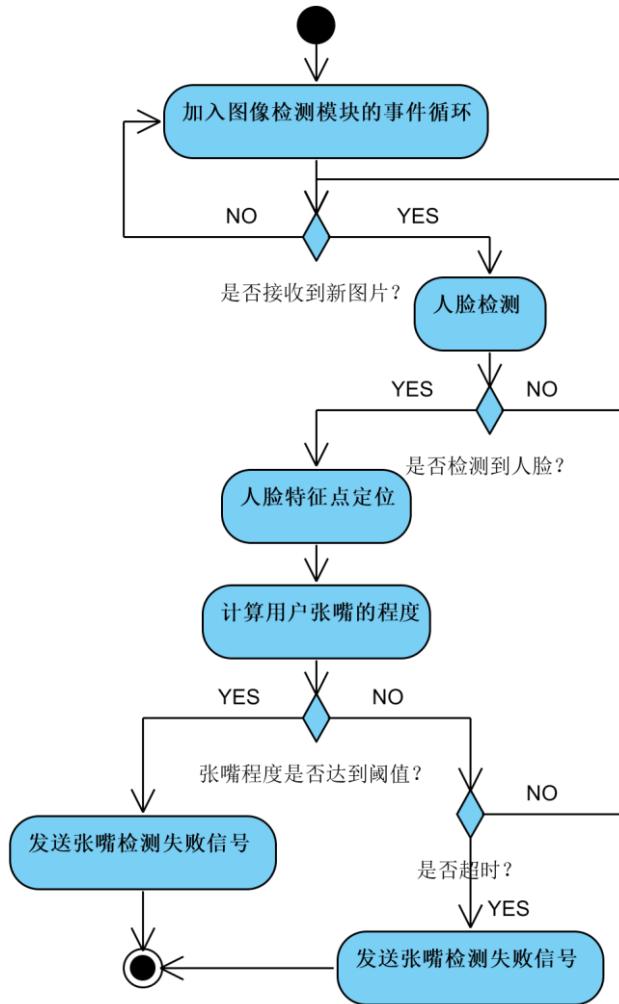


图 5-26 张嘴检测模块活动图

5.1.8 摆头检测模块设计与实现

5.1.8.1 摆头检测模块的类结构设计

揆头检测模块为客户机端的具体检测模块之一，该模块负责检测用户是否按照系统指示进行了揆头的动作。揆头检测模块由 3 个类组成：Analyser 类、

YawAnalyser 类以及 OpticalFlowCalculator 类。该模块的类图如图 5-27 所示：



图 5-27 摆头检测模块类图

其中, Analyser 类为人脸活体检测系统中具有具体检测功能的类的抽象基类, 系统中所有的检测类均可以由该类派生而来。YawAnalyser 类继承自 Analyser 基类, 负责检测用户是否按照系统提示进行了摇头动作。OpticalFlowCalculator 类为光流计算类, 该类设计为单例模式, 负责为各个具体的检测类提供光流计算的接口。

在 Analyser 类中, `start()`方法是纯虚接口, 为各个具体的检测模块的入口函数接口。

在 YawAnalyser 类中, `totalProgressTimeMs` 为预设的摇头检测流程的总耗时;

progressTimer 为摇头检测流程的超时计时器； updateSliderTimer 为更新客户机端图形化界面的指示滑块的位置的计时器； webcamCapture 为图像采集模块的实例化对象； faceDetector 为人脸检测模块的实例化对象； faceAligner 为人脸特征点定位模块的实例化对象； grayImage 为当前正在检测输入图像； faceBoundingBox 为标识图像中人脸位置的包围盒对象； opticalFlowCalculator 为光流计算类的实例化对象； isCurrentAlignmentValid 为表示当前保存的人脸特征点坐标的序列是否有效的标志位； isOpticalFlowCalculatorBusy 为表示光流计算器空闲的标志位； zoneMap 为图像分区图，用于将输入图像自适应地划分成脸部、左侧背景区以及右侧背景区； sliderPhase 为当前图形化界面滑块的位置； isProgressTimeout 为标识摇头检测流程是否超时的标志位； faceNormalVector 图像中人脸区域的光流的模的序列； leftBackgroundNormalVector 为左侧背景的光流的模的序列； rightBackgroundNormalVector 为右侧背景的光流的模的序列； facePhaseVector 为图像中人脸区域的光流的方向的序列； leftPhaseVector 为左侧背景的光流的方向的序列； rightPhaseVector 为右侧背景的光流的方向的序列； currentAlignment 为当前输入图片中的人脸的特征点坐标的序列； currentOpticalFlow 为当前的光流计算结果。在接口设计方面， separateNormAndPhase()方法为计算光流的模以及方向的函数； calculateZoneMap()方法为生成图片的分区图的函数，用于自适应地生成左侧背景区域以及右侧背景区域； start()方法为摇头检测模块的入口函数； receiveNewFrame()方法为从图像采集模块接收到新的用户输入图像的响应函数； receiveNewAlignment()方法为从人脸特征点定位模块接收到人脸特征点坐标的序列的响应函数； receiveNewOpticalFlow()方法为从光流计算器获取到光流的计算结果的响应函数； updateSliderTimeout()方法为更新图形化界面的滑块位置的响应函数； done()方法为摇头检测流程运行完毕后调用的函数； doAlignment()方法为调用人脸特征点定位模块进行人脸特征点定位的函数接口； calculateOpticalFlow()方法为调用光流计算器进行光流计算的函数接口。

5.1.8.2 摆头检测模块核心处理流程

YawAnalyser 类也是由 Controller 对象通过工厂类进行实例化，并且在创建对象后将其加入图像采集模块的事件响应队列。摇头检测的流程如下：首先通过语音指令以及图形化界面的滑块对用户的摇头动作进行引导，同时捕捉客户机端的所有图片并且计算相邻的两帧之间的稠密光流，最后分别计算脸部、左侧背

景以及右侧背景的光流的模和相位角，通过分析脸部与背景的运动一致性来判断本次摇头动作是否有效。摇头检测模块的时序图如图 5-28 所示：

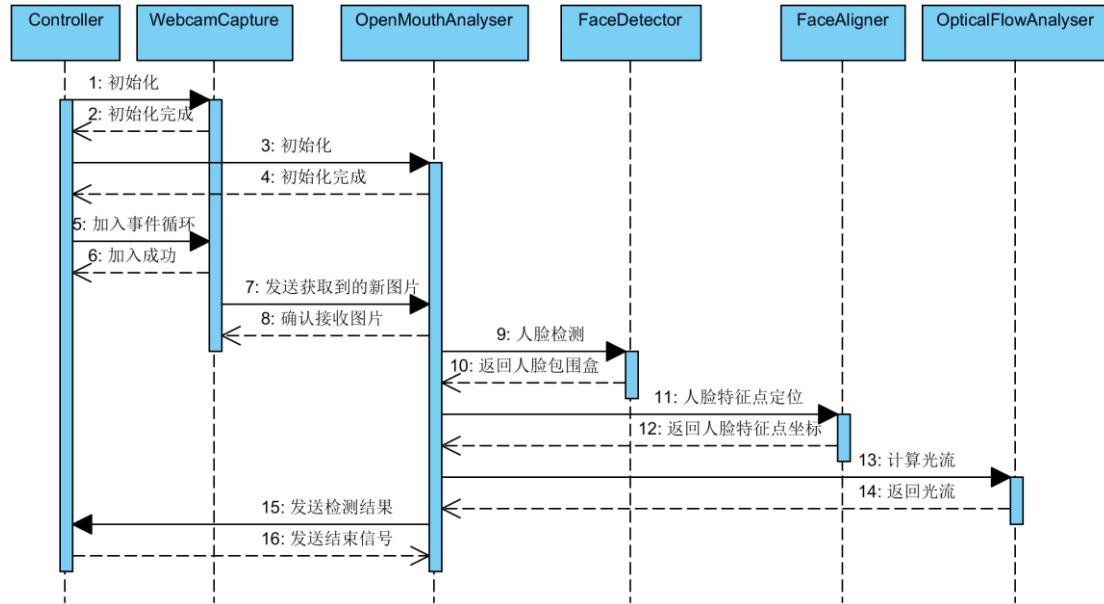


图 5-28 摆头检测模块时序图

在摇头检测的过程中，对于捕获的每一张图片，该模块首先会调用人脸检测模块进行人脸的粗定位，其次再通过调用人脸特征点定位模块对人脸的轮廓线进行定位，由人脸的轮廓线与边缘的关系计算得出图像的分区图，其中划定了本次采样的左侧背景区域以及右侧背景区域，最后使用光流计算各个区域之间的运动一致性。

在每次图像分区图的计算中，首先需要获取人脸位置的包围盒以及人脸特征点坐标的序列，根据人脸特征点坐标的序列可以确定人脸的轮廓，结合人脸位置的包围盒，将两者的交集设置为人脸区域；随后再通过人脸位置的包围盒在图片中的相对位置，选取人脸左侧与人脸右侧的中心区域，分别划定为左侧背景区以及右侧背景区。图像分区图计算示意图如图 5-29 所示：



图 5-29 图像分区图示意图

在运动一致性的分析中，首先在捕获图像的同时，记录下每一帧的各个区域的光流的模以及相位角，在捕获结束后，分别计算脸部与左侧背景、脸部与右侧背景的运动一致性。更具体地说，通过计算光流的模以及相位角的皮尔逊相关系数，并比较其与预设阈值的关系来做出是否有效的判断。该算法活动图如图 5-30 所示：

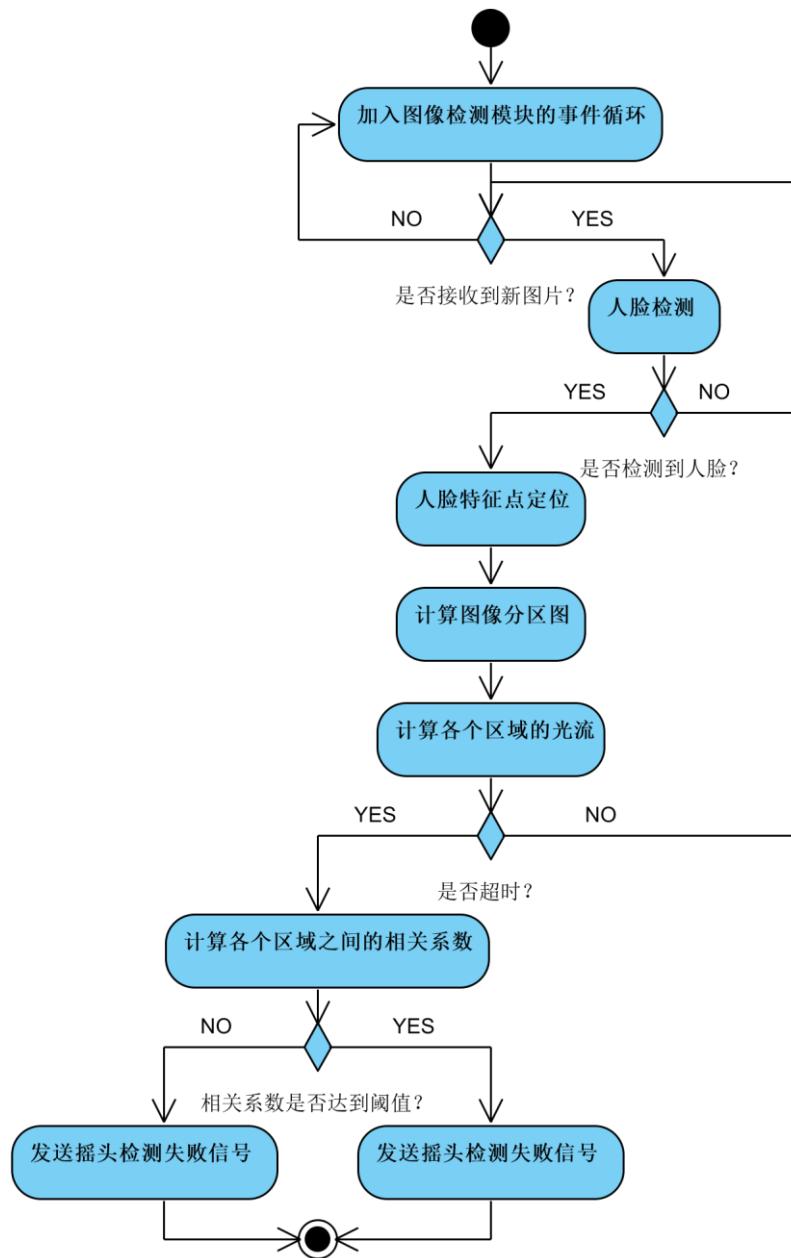


图 5-30 摆头检测模块活动图

5.2 服务器端详细设计与实现

人脸活体检测系统的服务器端主要负责网络内所有客户端的信息收集与显示、随机验证指令的生成与发送、验证结果的保存的功能。由于本系统的服务器端承载的任务较轻，计算量不大，因此采用轻量级的服务器脚本语言 Node.js 进行开发。Node.js 拥有高效率、轻量化、支持异步 IO 等特性，特别适合开发 IO

密集型而 CPU 计算量较轻的项目。使用 Node.js 可以极大地降低服务器端的软硬件要求，使项目的适用范围更加广泛。

在框架设计方面，由于涉及到完整的视图管理以及数据库技术，因此采用基于 Node.js 的 MVC 框架 Express 来开发服务器端。本节就 MVC 的三个模块的设计进行展开说明。

5.2.1 模型模块设计与实现

模型模块在 MVC 三层架构中又称为数据持久层，主要负责人脸活体检测系统与数据库系统的通信，将系统中数据的“增删改查”的操作同步至数据库系统中。

人脸活体检测系统采用非关系型数据库 MongoDB 作为数据库系统，为了简化开发流程，本系统选用 Mongoose 作为系统的数据持久层组件。基于 Mongoose 开发数据持久层需要为每一个数据库集合（Collection）创建一个对应的模式（Schema），在本系统中，设计有 2 个模式，分别对应着数据库中的 2 个集合：分别为 AdminUser、AnalyserResult。

AdminUser 模式所对应的数据库集合负责存储管理员用户的相关信息，包括管理员用户的用户名以及密码。AnalyserResult 模式所对应的数据库集合负责存储人脸活体检测结果的相关信息，包括随即指令 ID、随机指令内容、验证结果以及最后一帧的截图的保存路径。

AdminUser 模式与数据库集合的对应关系如表 5-5 所示：

表 5-5 AdminUser 模式关联字段

```
var UserSchema = new mongoose.Schema({  
    username : String,      //管理员用户名字段  
    password : String       //管理员密码字段  
});
```

AnalyserResult 模式与数据库集合的对应关系如表 5-6 所示：

表 5-6 AnalyserResult 模式关联字段

```
var ResultSchema = new mongoose.Schema({
    ID : Number,           //检测 ID 字段
    timeStamp : String,     //时间戳字段
    IPAddress : String,     //客户机端 IP 地址字段
    randomOrder : String,   //随机指令字段
    result : Number,        //检测结果字段
    imgPath : String        //截图保存路径字段
});
```

5.2.2 控制器模块设计与实现

控制器模块在 MVC 三层架构中又称为业务逻辑层，主要负责人脸活体检测系统的服务器端的请求拦截以及路由功能。在本系统中，服务器端控制器模块具体负责管理员登录请求的处理、管理员控制台页面请求的处理、客户机端获取随机指令序列的请求处理以及客户机端返回验证结果的请求处理。其中，管理员登录请求以及管理员控制台页面请求为服务器端与浏览器之间的通信行为，而随机指令序列的获取的请求以及返回验证结果的请求为服务器端与客户机端之间的通信行为。

服务器端与客户机端的通信的活动图如图 5-31 所示：

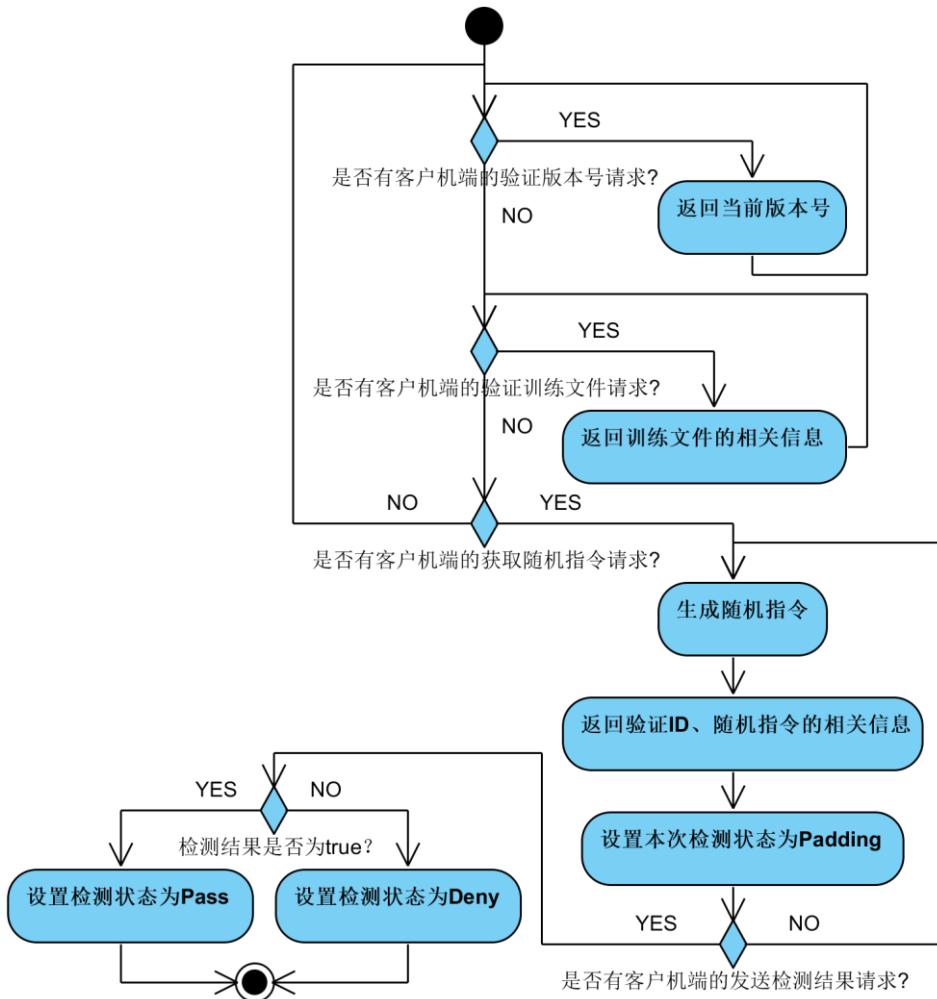


图 5-31 服务器端与客户机端的通信过程活动图

在服务器成功运行之后，就开始等待客户机端发送的请求。在服务器收到客户机的验证版本号的请求后，会返回当前系统的版本号供客户机端进行校验；随后，服务器会收到客户机端发来的验证训练文件的请求，服务器会将训练文件的散列码以及下载地址返回给客户机，供客户机进行验证或者更新训练文件。

客户机在进行每次人脸活体检测前，都会向服务器发送获取随机指令的请求，在服务器收到该请求后，即开启一次检测流程：首先生成随机的指令序列，并且将验证 ID 以及随机指令发送回客户机，同时在数据库中新增一个检测条目，并将其检测状态设置为 Padding；之后，服务器端等待接收客户机发来的检测结果，如若检测结果为通过，则将数据库中该条目所对应的检测状态设置为 Pass，反之，则将该条目所对应的检测状态设置为 Deny。

5.2.3 视图模块设计与实现

视图模块在 MVC 三层架构中又称为表示层，主要负责人脸活体检测系统的服务器端的页面渲染以及页面展示的功能。本系统的前端页面主要由以下几个页面组成：管理员登录页面、管理员控制台页面、验证结果查询页面。视图模块的每一个页面均封装为一个页面模版，向控制器模块暴露实例化接口，接受控制器模块的实例化参数并渲染页面，最后将结果页面发送给客户机端。

在管理员登录页面中，主要控件为一个表单。通过触发表单的提交事件来将表单内容传递到服务器端，完成管理员登录的功能实现。

在管理员控制台页面中，页面框架采用左右分栏设计。左侧为系统导航栏，右侧的上部为系统主要信息显示部分，该部分会向控制器模块暴露 4 个接口，分别接收系统版本号、当前客户机连接数量、总检测次数以及总检测失败次数。右侧的下部为系统的负载示意图，该部分向控制器模块提供一个数据初始化的接口，通过接受一个数组来标识系统的历史总负载以及总体验证通过率的数据。

在验证结果查询页面中，页面框架采用左右分栏设计。

5.3 本章小结

本章主要阐述了人脸活体检测系统的详细设计及其实现，从系统的总体设计触发，自顶向下地逐步精化，通过对客户机端的 8 个模块以及服务器端的 3 个模块进行细化设计，并运用 UML 图从软件工程的角度对系统的类结构设计、各模块间调用关系以及各模块的核心处理流程进行了详尽的描述。在接下来的一章中，将着重描述系统的部署以及测试相关的内容。

第六章 系统部署与测试

本章将会阐述人脸活体检测系统的部署以及测试相关的工作。首先，通过部署流程将人脸活体检测系统在测试环境顺利运行，然后对系统进行测试，其中测试内容分成两大块，一是针对系统的各个功能性用例进行功能性测试，即通过不同类型的输入检验人脸活体检测系统的功能是否完整；二是针对本系统的非功能性需求进行非功能性测试。

6.1 系统部署

人脸活体检测系统采用 C/S 结构，因此需要对服务器端以及客户机端进行分别部署。在服务器端，所采用的系统环境如表 6-1 所示：

表 6-1 服务器端部署软件环境

类型	内容
操作系统	Ubuntu Linux 15.10
Javascript 运行环境	Node 5.3.0
Web 开发框架	Express 4.13.4
数据库系统	MongoDB 3.2
数据库建模工具	Mongoose 4.3.7

在客户机端，所采用的系统环境如表 6-2 所示：

表 6-2 客户机端部署软件环境

类型	内容
操作系统	Ubuntu Linux 15.10
C++ 应用程序开发框架	Qt 5.5.1
计算机视觉库	OpenCV 3.1

本系统客户机端的主界面如图 6-1 所示，其中，区域 A 为在摇头检测阶段引导用户摇头的引导滑块，当进入摇头检测流程时，该滑块会指引用户摇头，用户仅需跟着滑块摇头即可完成摇头检测；区域 B 为摄像头的实时捕捉图像，将图像信息以及相关的检测信息反馈给用户，方便用户调整位置以及姿态；区域 C 为相关参数、随机指令序列、检测结果的显示区域，在最上方为系统的实时帧率，

分别反映当前主界面的帧率、人脸特征点定位的帧率、光流计算的帧率，中部为随机指令序列的显示区域，当客户机端从服务器端获取到随机指令序列后，就会将该指令序列显示在该区域，下部为检测结果的显示区域，分别由 PASS、PADDING、DENY 表示检测通过、正在检测中以及检测失败三个状态。

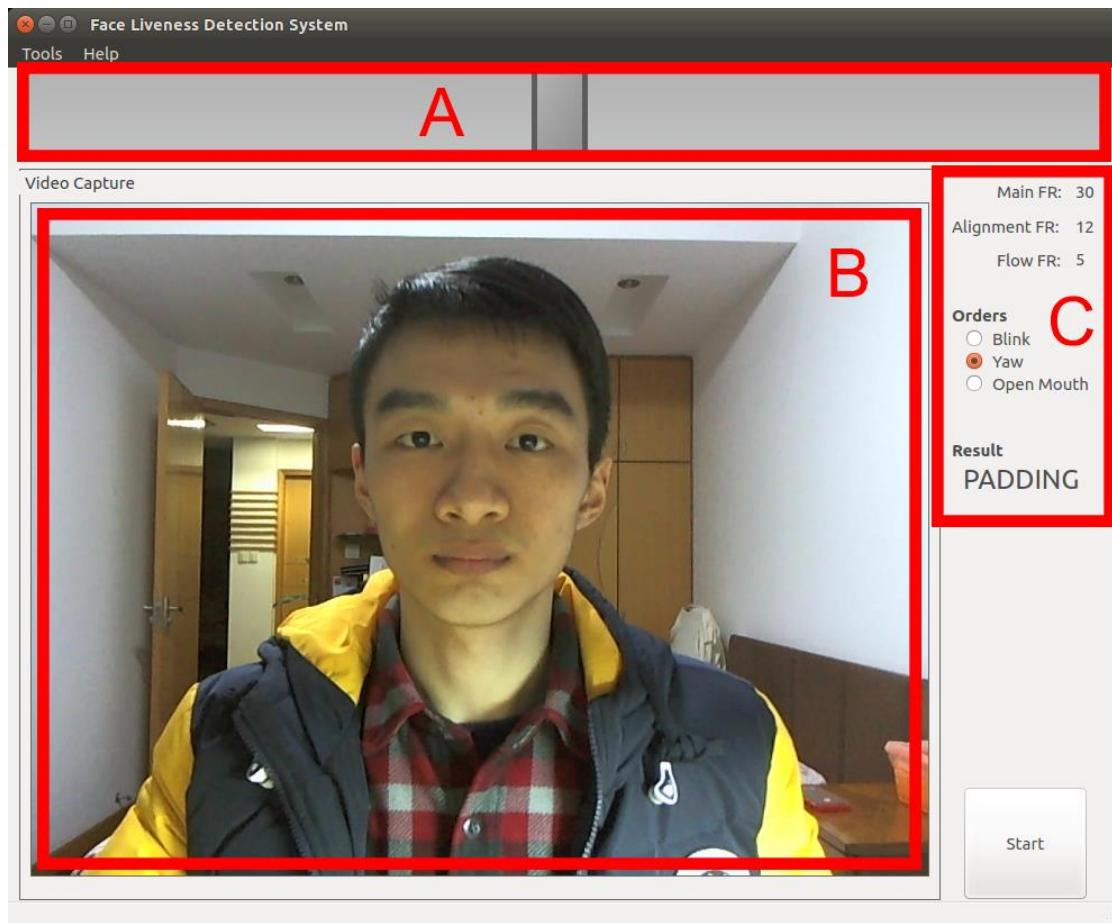


图 6-1 人脸活体检测系统客户机端主界面

本系统的服务器端可以通过 Node.js 应用部署的方法部署于 Windows、Linux 以及 Mac 系统上，本次测试以 Linux 系统为例。服务器端成功运行后，可以通过浏览器监控系统的运行情况。本系统服务器端的主界面如图所示，在服务器端的主界面中，主要显示了当前系统版本号、当前与服务器端连接的客户机端的数量、总检测计数、检测失败计数四个指标以及统计每日检测中通过与失败数量的图表。

第六章 系统部署与测试

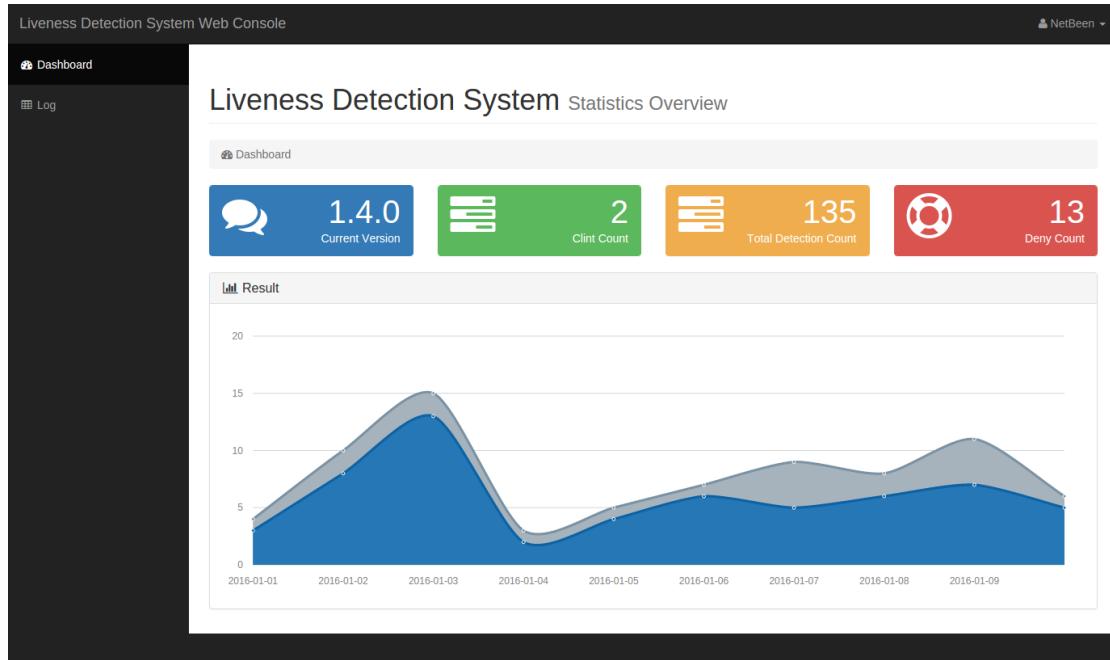
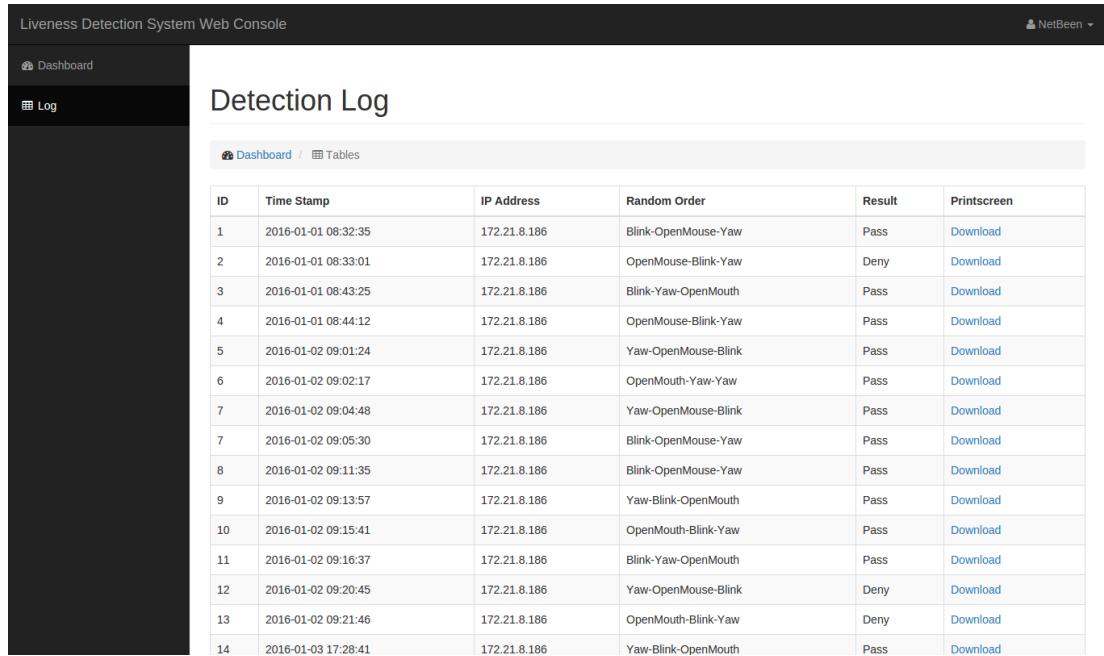


图 6-2 人脸活体检测系统服务器端主界面

本系统服务器端的日志界面如图 6-3 所示，在服务器端的日志界面中可以查询到每一次系统进行验证的详细信息，包括验证流程的编号、时间戳、客户机端的 IP 地址、本次检测的随机指令以及本次检测的结果。对于每一个检测条目，还可以下载最后一帧的截图，方便对检测结果进行人工校验。



The screenshot shows a web-based log viewer for a liveness detection system. The interface includes a sidebar with 'Dashboard' and 'Log' options, and a main content area titled 'Detection Log'. Below the title is a breadcrumb navigation bar with 'Dashboard / Tables'. The main content is a table with the following columns: ID, Time Stamp, IP Address, Random Order, Result, and Printscren. The table contains 14 rows of log entries, each detailing a specific detection attempt with its timestamp, IP address, sequence of detected features (e.g., Blink-OpenMouse-Yaw), result (Pass or Deny), and a 'Download' link for printscreens.

ID	Time Stamp	IP Address	Random Order	Result	Printscreen
1	2016-01-01 08:32:35	172.21.8.186	Blink-OpenMouse-Yaw	Pass	Download
2	2016-01-01 08:33:01	172.21.8.186	OpenMouse-Blink-Yaw	Deny	Download
3	2016-01-01 08:43:25	172.21.8.186	Blink-Yaw-OpenMouth	Pass	Download
4	2016-01-01 08:44:12	172.21.8.186	OpenMouse-Blink-Yaw	Pass	Download
5	2016-01-02 09:01:24	172.21.8.186	Yaw-OpenMouse-Blink	Pass	Download
6	2016-01-02 09:02:17	172.21.8.186	OpenMouth-Yaw-Yaw	Pass	Download
7	2016-01-02 09:04:48	172.21.8.186	Yaw-OpenMouse-Blink	Pass	Download
7	2016-01-02 09:05:30	172.21.8.186	Blink-OpenMouse-Yaw	Pass	Download
8	2016-01-02 09:11:35	172.21.8.186	Blink-OpenMouse-Yaw	Pass	Download
9	2016-01-02 09:13:57	172.21.8.186	Yaw-Blink-OpenMouth	Pass	Download
10	2016-01-02 09:15:41	172.21.8.186	OpenMouth-Blink-Yaw	Pass	Download
11	2016-01-02 09:16:37	172.21.8.186	Blink-Yaw-OpenMouth	Pass	Download
12	2016-01-02 09:20:45	172.21.8.186	Yaw-OpenMouse-Blink	Deny	Download
13	2016-01-02 09:21:46	172.21.8.186	OpenMouth-Blink-Yaw	Deny	Download
14	2016-01-03 17:28:41	172.21.8.186	Yaw-Blink-OpenMouth	Pass	Download

图 6-3 人脸活体检测系统服务器端日志界面

6.2 系统功能性测试

本节主要是针对之前设计与实现的人脸活体检测系统的检测算法进行真实数据测试，测试的主要目的是验证本系统是否可以对用户输入的图像序列进行多角度的分析并且给出合理正确的检测结果。

测试的方法如下：测试者通过使用不同的仿冒手段对人脸活体检测系统进行攻击，查看本系统应对仿冒者的无效输入时，是否能够正确地检测并做出判断。由于本系统有三个检测模块，并且其执行顺序由服务器随机生成，因此在进行功能性测试时，将各个检测模块单独作为独立的检测单元进行测试，即彼此的测试结果均不会对其他检测单元造成影响。具体的检测结果如表 6-3 所示：

表 6-3 各检测模块功能性测试结果

仿冒人脸输入方式	眨眼检测	张嘴检测	摇头检测
手持照片	未通过	未通过	未通过
三维石膏模型	未通过	未通过	通过
高分辨率视频	通过	通过	通过
高分辨率面具	通过	未通过	通过

如表所示，在各个检测模块独立运行的模式下，仅有高分辨率视频可以顺利通过验证，其他方式的非法输入均被系统拒绝。该项测试证明，系统的检测模块在绝大多数情况下对于真实人脸以及非法人脸进行识别。由于高分辨率视频具有极强的仿真度，因此需要结合服务器端生成的随机验证指令进行识别，由于服务器端返回的验证指令的顺序是随机的，而视频中的人脸动作的录制顺序是一致的，由此来区分人脸与高分辨率视频。

通过系统的功能性测试，可以验证人脸活体检测系统可以对非法的人脸输入进行检测，达到该系统的功能性需求。

6.3 系统非功能性测试

本节主要针对人脸活体检测系统的非功能性需求进行测试，本系统的非功能性需求主要由实时性、高效性、健壮性以及可维护性构成。

实时性方面，主要用过测试系统的每秒帧率，来判断能否给用户提供一个良好的交互式体验。通过对各个模块的帧率进行统计，其结果如表 6-4 所示：

表 6-4 各检测模块运行帧率

模块名称	平均运行帧率
眨眼检测模块	26 FPS
张嘴检测模块	12 FPS
摇头检测模块	5 FPS

正如表所展示的帧率，很难达到人类肉眼视觉残留的速率（约为 12 FPS 以上），因此本系统在设计架构之时采用异步通信的方式作为模块间的调用，使得用户主界面的帧率可以保持在 28 FPS 到 30 FPS 之间，满足实时性的需求。

高效性方面，主要通过测试系统的所有检测流程是否能够在较短的时间内完成。通过对各个模块的检测时间的统计，其结果如表 6-5 示所：

表 6-5 各检测模块运行时长

模块名称	平均运行时间
眨眼检测模块	1.6 秒
张嘴检测模块	0.9 秒
摇头检测模块	6 秒

本系统的活体检测的耗时由各个模块的运行时间以及系统的数据通信时长所构成，总时间可以控制在 10 秒之内，满足高效性的需求。

健壮性方面，主要通过测试在非常规的运行方式下，系统是否能够稳定运行。通过对系统以非常规的运行方式来模拟用户的实际使用情况，并记录系统的反馈情况，其结果如表 6-6 所示：

表 6-6 健壮性测试反馈结果

非常规运行方式	系统反馈结果
开启检测流程后，未捕捉到人脸	计时器超时后判定检测失败
眨眼检测环节，未检测到眼睛	计时器超时后判定检测是吧
检测流程未结束，再次点击开始按钮	重新获取验证指令，重新开始检测流程
获取随机指令超时	提示用户重新点击开始按钮
返回检测结果超时	超过重试次数则提示“提交失败”
检测流程执行中，丢失人脸	忽略丢失人脸的序列帧

从表中的测试结果可以看出，系统在应对非常规输入的均可以采用合理的方式进行处理，并不会因为客户机端的输入异常或者客户机端与服务器端的通信异常而导致系统出现闪退、卡死等严重情况，从而保证系统的稳定运行。

可维护性方面，系统在设计之初即采用模块化设计，根据功能需求为每个模块设计通信接口，由此能够有效降低系统模块之间的耦合度，极大程度上降低系统维护的难度。此外，本系统所采用的所有依赖库均为支持跨平台特性的标准库（Qt、OpenCV、Node.js 以及 MongoDB），因此保证了本系统具有良好的跨平台性能，本系统的客户机端已经在 Windows 平台以及 Linux 平台成功部署。

6.4 本章小结

本章主要对人脸活体检测系统进行了功能性测试以及非功能性测试。从测试结果可以看出，本系统对于非真实的人脸具有较强的辨识能力，可以通过简单的交互式验证手段对从客户机端采集的用户图像进行人脸活体检测，从而判断该人脸输入是否真实合法。此外，本系统还满足系统的非功能性需求，包括实时性、高效性、健壮性以及可维护性，这些特性表示系统不仅具有良好的人脸活体检测能力，还具有较为友好的用户交互以及较强的容错能力。

第七章 总结

7.1 论文总结

为了保障人脸识别系统的安全，在最大程度上排除非法的伪造人脸输入人脸识别系统所导致的安全隐患；同时又需要兼顾用户的使用体验，使用尽可能少的交互方式以达到最大化地检测非法输入的人脸。本文设计了一个基于 C/S 架构的人脸活体检测系统。

本系统设计的目的是通过较少的交互式验证的方式，最大程度上增加攻击者使用仿冒人脸攻击人脸识别系统的成本以及难度。项目的特色以及创新点为通过对真假人脸特点的分析设计出一套快捷简易的验证方式，可以在短时间（约为 10 秒）内完成真假人脸的鉴别，从而最大程度上保障了人脸识别系统的安全。

7.2 个人收获

在人脸活体检测系统的设计与实现的过程中，我完成的主要工作如下：

1. 参与了人脸活体检测系统的需求分析以及功能模块划分的工作，确定了系统的几种检测方式、交互强度以及用户体验方面的需求。
2. 通过阅读《Face Alignment by Explicit Shape Regression》及其参考文献和相关资料，整理出该算法的实现框架并且使用 C++ 实现。
3. 通过阅读《面向人脸识别的人脸活体检测方法研究》及其参考文献和相关资料，采用该文章中通过分析前背景之间的相关度来区别照片与人脸的思路，实现了通过计算前背景的稠密光流的相关系数来进行活体验证的功能模块。
4. 完成人脸活体检测系统的客户机端以及服务器端的所有模块的设计与开发。
5. 完成人脸活体检测系统的单元测试、功能测试以及性能测试。
6. 通过分析性能测试的结果，总结项目中对时间复杂度影响较大的几个模块，对其进行参数优化以及输入数据预处理，以达到提高系统的运行速度，提高用户体验。

在系统设计与实现以及论文写作的过程中，老师们给我很多指导，我收获很大，具体学习到的经验以及知识点如下：

1. 在企业导师谢晓华副研究员的帮助以及指导下，阅读了大量计算机视觉方向的国内外重要文献，尤其是计算机视觉领域的顶级会议上所发表的文章，使

我对计算机视觉的起源、发展历史、重要里程碑以及著名的算法有了全面而具体的认识，从而进一步认识到人脸活体验证的重要意义，这为我后续设计并且实现本系统打下了坚实的基础。

2. 本系统的客户机端是在 Qt 5.5.1、OpenCV 3.1 环境下开发完成的，服务器端是在 Node.js 5.3.0、MongoDB 3.2 环境下开发完成的。在前期的学习阶段，我对人脸特征点定位算法有了较为全面的认识；在系统实现的过程中，又提升了我使用上述工具链的熟练程度。本项目涉及服务器后台程序、web 前端界面以及桌面端软件的编写和调试，全面而深入地提升了我的编码能力。
3. 本项目从最初的模块设计到最后的编码实现都是由我本人完成的，在此过程中，我将软件工程的理念进行了实际应用，并且取得了不错的效果。这对未来继续从事软件开发工作具有非常重要的意义，不仅让我掌握了软件工程的开发流程，并且培养了良好的编程习惯以及编程风格。

7.3 改善空间

虽然人脸活体检测系统能够使用较为轻量级的交互来达到验证人脸的真实性的目的，不过该系统本身仍然有着非常大的改善空间。具体待改善的方面如下：

1. 检测模块的数量还有所欠缺。由于本系统采用随机验证指令来应对高清视频的回放攻击，因此检测模块的数量的增加会导致系统的安全性呈指数式增加，所以未来还可以继续增加其他的验证手段。
2. 引入非交互式验证机制。由于交互式验证机制的特点，所以需要用户配合本系统进行哪个，这会对用户造成一定麻烦，因此今后可以将交互式验证方式改为非交互式验证方式，进一步提高用户体验。

参考文献

- [1]. Geoffrey I. Webb, Janice R. Boughton, Zhihai Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. Machine Learning(ML), 58, 5–24, 2005 提出半朴素贝叶斯定理
- [2]. Piotr Doll’ar, Peter Welinder, Pietro Perona. Cascaded Pose Regression. IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2010 提出随机蕨回归器
- [3]. Mustafa Özuysal, Michael Calonder, Vincent Lepetit, Pascal Fua. Fast keypoint recognition using random ferns. IEEE Transactions on Pattern Analysis & Machine Intelligence(PAMI), 2010
首次提出 random fern, 提出随机蕨分类器
- [4]. 蒋秀鹏. 基于 NodeJS 的数字标牌系统的设计与实现[硕士学位论文]. 天津: 南开大学, 2014 参考了一些 Node.js 的概念
- [5]. 王越. 基于 nodejs 的微博系统的设计与实现[硕士学位论文]. 成都: 电子科技大学, 2014
参考了一下 nodejs 以及 mongodb 的概念
- [6]. Xudong Cao, Yichen Wei, Fang Wen, Jian Sun. Face Alignment by Explicit Shape Regression. International Journal of Computer Vision(IJCV) 107:177–190, 2014 提出了 ESR 算法
- [7]. Oliver Jesorsky, Klaus J. Kirchberg, Robert W. Frischholz. Robust Face Detection Using the Hausdorff Distance. Third International Conference on Audio- and Video-based Biometric Person Authentication, 2001 提出了 BioID 数据库
- [8]. Peter N. Belhumeur, David W. Jacobs, David J. Kriegman, Neeraj Kumar. Localizing Parts of Faces Using a Consensus of Exemplars. IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2011 提出了 LFPW 数据库
- [9]. Lin Liang, Rong Xiao, Fang Wen, Jian Sun. Face Alignment Via Component-Based Discriminative Search. European Conference on Computer Vision(ECCV), 2008 提出 LFW87 数据库
- [10]. Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev. Interactive facial feature localization. European Conference on Computer Vision(ECCV), 2012 提出 Helen 数据库
- [11]. 全义明, 黄蔚, 李戴维. 基于 MongoDB 的信息集成系统的设计与实现. 信息技术: 1009-2552(2015)02-0125-05, 2015 参考了 MongoDB 的简介
- [12]. Iain Matthews, Simon Baker. Active Appearance Models Revisited. International Journal of Computer Vision(IJCV), 60(2), 135–164, 2004 AAM 参考文献 1
- [13]. Patrick Sauer, Tim Cootes, Chris Taylor. Accurate Regression Procedures for Active Appearance Models. British Machine Vision Conference(BMVC), 2011 AAM 参考文献 2
- [14]. Jason Saragih, Roland Goecke. A Nonlinear Discriminative Approach to AAM Fitting. International Conference on Computer Vision(ICCV), 2007 AAM 参考文献 3
- [15]. Timothy F. Cootes, Gareth J. Edwards, Christopher J. Taylor. Active Appearance Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(6), 681–685, 2001 AAM 参考文献 4
- [16]. David Cristinacce, Tim Cootes. Boosted Regression Active Shape Models. British Machine

参考文献

- Vision Conference(BMVC), 2007 基于回归的传统算法 1
- [17].Michel Valstar, Brais Martinez, Xavier Binefa. Facial Point Detection using Boosted Regression and Graph Models. IEEE Conference on Computer Vision and Pattern Recognition, 2010 基于回归的传统算法 2
- [18].Herbert Bay, Tinne Tuytelaars, Luc Van Gool. SURF: Speeded Up Robust Features. European Conference on Computer Vision(ECCV), 2006 提出了一种基于 SFM (Structure from Motion) 的活体检测模型
- [19].Klaus Kollreider, Hartwig Fronthaler, Josef Bigun. Non-intrusive liveness detection by face images. European Conference on Computer Vision(ECCV), 2009 利用光流来分析人脸各部位的移动量，最终进行活体检测
- [20].孙霖. 人脸识别中的活体检测技术研究[博士学位论文]. 杭州: 浙江大学, 2010 提出了利用人脸识别进行多模活体检测的概念
- [21].杨健伟. 面向人脸识别的人脸活体检测方法研究[硕士学位论文]. 北京: 北京邮电大学, 2014
- [22].曹瑜. 活体人脸检测技术研究[硕士学位论文]. 北京: 北京工业大学, 2014
- [23].刘华成, 人脸活体检测关键技术研究[硕士学位论文]. 宁波: 宁波大学, 2014

致谢

转眼间，在中科大的三年求学历程即将走到终点。回顾走过的这三年校园时光，丰富多彩，历历在目。对我个人而言，不仅仅在学识水平上有了质的提升，而且还结识了很多优秀的老师和同学，与他们的交往是我人生中一段非常宝贵的经历。为此，我非常感谢中国科学技术大学提供这个高层次的平台，正是由于这些老师和同学们的帮助，才能让我在软件工程领域打下坚实的基础。与此同时，我也要感谢中国科学院深圳先进技术研究院为我提供了一个难得的实习机会，在实习期间，我感受到了实验室里充满激情的工作环境，学到了很多在学校无法学习到的技能。

在完成本次毕业设计以及论文撰写期间，我的校内导师凌强副教授、曹洋副教授以及我的企业导师谢晓华副研究员对我的帮助非常大。曹洋老师从开题报告开始就在百忙之中抽空对我的项目进行了详尽而又明确的指导，他严谨的工作态度深深感染了我，从曹洋老师身上，我不仅得到了项目研发方向的指导，还养成了求实的学术习惯。与此同时，本篇论文的创作也离不开谢晓华副研究员的倾力指导，大到项目设计思路，小到具体公式的推演，谢老师都会耐心地引导我，给我指出正确的方向。在为期一年半的实习过程让我的软件开发技术得到了新的提升，也教会了我很多为人处事的道理，这都与老师们的辛勤教导分不开。在此，我要向三位老师致以最诚挚的谢意，谢谢你们！

此外，我也要感谢张炜、申江涛、范晓晨、马林、吴博剑等同学，他们曾经在项目遇到瓶颈的时候或给予我指导和鼓励，或与我一起攻坚克难，项目最后的完整实现离不开他们的帮助。在此，我要向他们道声谢谢！

最后，我还要感谢我的父母，在项目实现以及论文撰写的过程中，他们一直默默地支持着我，每当项目遇到瓶颈，他们都能给出中肯的意见，并且给予无私的鼓励，帮助我成功完成该篇论文，谢谢！