

**

**HUNAN UNIVERSITY**

本科生毕业设计(论文)

|  |  |  |
| --- | --- | --- |
| **设计论文题目：** | | 人脸检测与识别在ROS中的研 |
|  |  | 究与应用 |
|  | 学生姓名： | 石方玉 |
|  | 学生学号： | 201326010412 |
|  | 专业班级： | 软件1304班 |
|  | 学院名称： | 信息科学与工程学院 |
|  | 指导老师： | 肖雄仁 |
|  | 学院院长： | 李肯立 |

2017 年 05 月 15 日

**湖 南 大 学**

**毕业设计（论文）原创性声明**

本人郑重声明：所呈交的设计（论文）是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

学生签名： 日期：20 年 月 日

**毕业设计（论文）版权使用授权书**

本毕业设计（论文）作者完全了解学校有关保留、使用设计（论文）的规定，同意学校保留并向国家有关部门或机构送交设计（论文）的复印件和电子版，允许设计（论文）被查阅和借阅。本人授权湖南大学可以将本设计（论文）的全部或部分内容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和汇编本设计（论文）。

本设计（论文）属于

1、保 密 ，在 年解密后适用本授权书。

2、不保密 。

（请在以上相应方框内打“√”）

学生签名： 日期：20 年 月 日

导师签名： 日期：20 年 月 日

人脸检测与识别在ROS中的研究与应用

作者：石方玉

导师：肖雄仁

# 摘 要

本文通过对人脸检测/识别和ROS系统的研究，开发出了一套实时的人脸检测与识别系统。

用户可通过网页端访问，也可以通过本地客户端访问，使用系统功能。除实时的人脸检测/学习/识别功能外，用户还可以导入人脸图片，上传至系统进行学习处理。用户也可以对某一个学习过的人的人脸进行继续学习以增加识别率。也可以删除某个学习过的人的人脸数据。另外，用户还可以控制检测/识别/学习的处理频率。

本系统后台统一采用Python语言开发，其中人脸检测/学习/识别三大核心模块基于opencv、dlib、torch等第三方框架。B/S和C/S架构的系统都运行于ROS系统之上，采用ROS中的Publisher/Subscriber模式，进行图像等数据的通信和传输。本文开发的系统基于B/S和C/S结构的都有实现。

B/S结构的系统后台与网页端之间建立WebSocket连接，采用JSON格式进行数据传输，网页前端使用Bootstrap编写界面，使用jQuery编写网页端逻辑代码。

C/S结构的客户端使用跨平台的PySide GUI框架（Qt在Python语言上的绑定）编写图形界面，采用PySide框架里自带的通信机制Signal/Slot（信号和槽）进行数据传输。

关键词：人脸检测；人脸学习；人脸识别；ROS；

**The Research and Application of Face Detection/Recognition in Robot Operating System**

Author: Fangyu Shi

Tutor: Xiongren Xiao

# Abstract

In this paper, a real-time face detection and recognition system is developed by studying the face detection, recognition and ROS system.

Users can use system functions by accessing through the web or the local client. In addition to real-time face detection / learning / recognition function, users are able to import face images, upload to the system for face learning. Users can also continue learning the face of a person who has been learned to increase the recognition rate. Users can delete the face data of a person who has been learned too. In addition, users can control the processing rate of face detection / learning / recognition.

The backend of this system is developed by using Python programming language, in which face detection / learning / recognition of the three core modules are based on opencv, dlib, torch and other third-party framework. Both B/S and C/S architecture systems are running on the ROS system, using ROS’s built-in mode: Publisher / Subscriber mode for image and other data’s communication and transmission. The system developed in this paper is based on B/S and C/S architecture.

B/S architecture of the system establishes a WebSocket connection between the backend and the Web site, using JSON format for data transmission, web front-end use Bootstrap write user interface, and the use of jQuery framework is prepared for web-side logical code.

C/S architecture of the client using PySide, a cross-platform GUI framework (Qt framework’s binding on the Python language) to write the graphics user interface, using the PySide framework’s built-in communication mechanism, Signal/Slot for data transmission.

**Key Words:** **Face Detection; Face Learning; Face Recognition; ROS**

# 插图索引

[图3. 1系统用例图 7](#_Toc483836305)

[图3. 2 B/S架构流程图 9](#_Toc483836306)

[图3. 3 C/S架构业务流程图 10](#_Toc483836307)

[图4. 1 B/S架构系统设计图 12](#_Toc483836311)

[图4. 2 C/S架构系统设计图 12](#_Toc483836312)

[图5. 1核心模块总设计图 13](#_Toc483836395)

[图5. 2连接模块用户界面 21](#_Toc483836396)

[图5. 3主页模块页面设计 22](#_Toc483836397)

[图5. 4导入图片学习模块页面设计 27](#_Toc483836398)

[图5. 5 Qt客户端模块界面设计 28](#_Toc483836399)

[图6. 1人脸检测功能测试截图 35](#_Toc483909935)

[图6. 2人脸学习功能测试截图 36](#_Toc483909936)

[图6. 3人脸识别功能测试截图（该人脸已被学习） 36](#_Toc483909937)

[图6. 4人脸识别功能测试截图（该人脸已被学习） 37](#_Toc483909938)

[图6. 5人脸识别功能测试截图（该人脸未被学习） 37](#_Toc483909939)

[图6. 6导入图片学习功能测试截图 38](#_Toc483909940)

[图6. 7导入图片学习功能测试截图（显示damme已经被学习过） 38](#_Toc483909941)

[图6. 8导入图片学习功能测试截图（人脸识别结果） 39](#_Toc483909942)

[图6. 9人脸检测功能测试截图 39](#_Toc483909943)

[图6. 10人脸学习功能测试截图 40](#_Toc483909944)

[图6. 11人脸识别功能测试截图 40](#_Toc483909945)

# 附表索引

[表3. 1 系统用例列表 9](#_Toc483836428)

[表5. 1接口getAllFaceBoxes 15](#_Toc483836509)

[表5. 2接口getLargestFaceBox 16](#_Toc483836510)

[表5. 3接口initData 16](#_Toc483836511)

[表5. 4接口saveData 17](#_Toc483836512)

[表5. 5接口trainFace 17](#_Toc483836513)

[表5. 6接口recognizeFace 18](#_Toc483836514)

[表5. 7接口updateRecognizer 18](#_Toc483836515)

[表5. 8接口getCamera 19](#_Toc483836516)

[表5. 9接口imgPub 19](#_Toc483836517)

[表5. 10接口saveImg 19](#_Toc483836518)

[表5. 11接口callback 20](#_Toc483836519)

[表5. 12接口bytes2NpImg 20](#_Toc483836520)

[表5. 13接口npImg2Bytes 20](#_Toc483836521)

[表5. 14接口window.String.prototype.format 21](#_Toc483836522)

[表5. 15接口startServer 22](#_Toc483836523)

[表5. 16类MyServerProtocol 22](#_Toc483836524)

[表5. 17类接口MyServerProtocol.onMessage 22](#_Toc483836525)

[表5. 18消息体MSG规范 23](#_Toc483836526)

[表5. 19接口validateIP 23](#_Toc483836527)

[表5. 20接口createWebSocket 24](#_Toc483836528)

[表5. 21接口decorateSocket 25](#_Toc483836529)

[表5. 22接口requestOnce 25](#_Toc483836530)

[表5. 23接口requestLoop 25](#_Toc483836531)

[表5. 24接口reuqestFixedLoop 25](#_Toc483836532)

[表5. 25接口handleMsg 26](#_Toc483836533)

[表5. 26接口detect 26](#_Toc483836534)

[表5. 27接口train 26](#_Toc483836535)

[表5. 28接口validateNameId 26](#_Toc483836536)

[表5. 29接口stopTrain 27](#_Toc483836537)

[表5. 30接口recognize 27](#_Toc483836538)

[表5. 31接口doNothing 27](#_Toc483836539)

[表5. 32接口delPerson 27](#_Toc483836540)

[表5. 33接口fitPersonsTable 28](#_Toc483836541)

[表5. 34接口createCanvasCtx 28](#_Toc483836542)

[表5. 35接口clearCanvas 28](#_Toc483836543)

[表5. 36接口createFileReader 29](#_Toc483836544)

[表5. 37接口read 29](#_Toc483836545)

[表5. 38类MainWidget 31](#_Toc483836546)

[表5. 39类成员字段MainWidget.imgSig 31](#_Toc483836547)

[表5. 40类成员方法MainWidget.detectSlot 31](#_Toc483836548)

[表5. 41类成员方法MainWidget.trainSlot 31](#_Toc483836549)

[表5. 42类成员方法MainWidget.recognizeSlot 32](#_Toc483836550)

[表5. 43类成员方法MainWidget.imgSlot 32](#_Toc483836551)

[表5. 44类成员方法MainWidget. personChangeSlot 32](#_Toc483836552)

[表5. 45类成员方法MainWidget. fitComboBox 32](#_Toc483836553)

[表5. 46类成员方法MainWidget. emitData 33](#_Toc483836554)

[表5. 47类成员方法MainWidget. handleMsg 33](#_Toc483836555)

[表5. 48类成员方法MainWidget. validateNameId 33](#_Toc483836556)

[表6. 1 B/S架构系统测试用例 34](#_Toc483836557)

[表6. 2 C/S架构系统测试用例设计 36](#_Toc483836558)

[表6. 3测试bug记录 37](#_Toc483836559)

**目 录**

[摘 要 I](#_Toc483844891)

[Abstract II](#_Toc483844892)

[插图索引 III](#_Toc483844893)

[附表索引 IV](#_Toc483844894)

[1. 绪论 1](#_Toc483844895)

[1.1课题背景及目的 1](#_Toc483844896)

[1.2 国内外研究状况 1](#_Toc483844897)

[1.3 本文所做工作 2](#_Toc483844898)

[1.4 论文构成 3](#_Toc483844899)

[2. 关键技术和工具 4](#_Toc483844900)

[3. 系统需求分析 7](#_Toc483844901)

[3.1功能需求分析 7](#_Toc483844902)

[3.2 用例需求分析 7](#_Toc483844903)

[3.3 业务流程分析 8](#_Toc483844904)

[4. 系统总体设计 11](#_Toc483844906)

[4.1 B/S架构系统设计 11](#_Toc483844907)

[4.2 C/S架构系统设计 12](#_Toc483844908)

[5. 系统详细设计与实现 13](#_Toc483844909)

[5.1 公共模块详细设计与实现 13](#_Toc483844910)

[5.1.1 人脸检测/学习/识别模块详细设计与实现 13](#_Toc483844911)

[5.1.2 ROS通信模块详细设计 16](#_Toc483844912)

[5.1.3 工具类模块详细设计 18](#_Toc483844913)

[5.2 B/S架构系统详细设计与实现 19](#_Toc483844914)

[5.2.1 WebSocket服务器模块详细设计与实现 19](#_Toc483844915)

[5.2.2 前端呈现模块详细设计与实现 21](#_Toc483844916)

[5.3 C/S架构系统详细设计与实现 27](#_Toc483844917)

[5.3.1 Qt客户端模块详细设计与实现 27](#_Toc483844918)

[6. 系统测试 32](#_Toc483844919)

[6.1 测试背景 32](#_Toc483844920)

[6.2 测试用例设计 32](#_Toc483844921)

[6.3 测试分析 35](#_Toc483844922)

[7. 总结和展望 36](#_Toc483844923)

[附录 39](#_Toc483844924)

[附录A 消息体规范 39](#_Toc483844925)

# 绪论

## 1.1课题背景及目的

针对生物特征进行识别随着信息技术的发展，被广泛地应用到各个工作、生活场景。各种场景下都需要对涉及到安全、个人隐私和财产的操作进行身份认证，以确保为本人操作[1]。人脸识别技术由于其具有直接、友好、方便、易于为用户接受以及设备造价较低等优点，具有广阔的应用前景[2]。

本课题旨在在ROS下实现人脸的检测、识别与跟踪功能，提高ROS机器人身份认证的便捷性、实时性、准确性以及安全性。

## 1.2 国内外研究状况

随着现在人们生活越来越科技化，古老的身份认证方式无法完全保证信息安全，所以研究替代的解决方案显得非常重要。基于生物特征的识别技术是目前广泛研究的一个课题。无论是在人们的日常生活和工作场景下，或是在公安系统、国家海关、军事方面等安全性要求更高的场景下，它应用前景都很广泛。此外，基于生物特征的识别技术还具有稳定性高、效率高的特点。它利用人类个体间独一无二，可以彼此区分的特征对人身份进行验证，如指纹、虹膜、人脸等识别技术 [1]。

其中，人脸识别技术以人脸图像作为关注点，通过对人脸区分彼此个体间的特征信息抽取出来，然后把在人脸库中搜索，与之进行比对，给出最终验证结果。人脸识别现在是基于生物特征的识别领域的宠儿。其优势主要在于，识别流程对验证人没有侵犯性，是非接触式的，验证人不会有所抵触 [3]。

然而，人脸识别面临各种困难，比如面部毛发的存在，照明和阴影的变化，以及可能的角度，缩放，和尺寸偏差等。因此，理想的人脸检测系统应试图减轻这些问题对识别效果的影响，同时提高检测的准确率并降低误报率。而在选择合适的检测率和误报率以及两者之间的平衡需要权衡用户的需求以及应用程序域[4]。

总而言之，目前在人脸识别领域未能解决的问题包括：精确、稳定的特征匹配；在特征采集的条件变化后，依然稳定强大的的人脸识别算法；一般化识别算法和自适应的学习问题；光照变化对人脸识别的影响问题；针对只识别动态人脸的算法；人脸采集设备分辨率低的问题；化妆、易容，佩戴墨镜、帽子、口罩等会干扰识别的问题。另外需要注意的是：上述问题大多都是组合性的[5]。

人脸识别算法比较复杂，如果想将理论上的算法转化为实际可应用的系统， 还需要花费大量研发时间和经费。因此，为了很好地解决算法研究上的问题，常常需要结合计算机视觉库，其中比较著名的有国外的OpenCV[6]。成型的人脸识别软件有国内的face++、云从科技、天诚盛业，国外的SkyBiometry、Betaface、Luxand。人脸识别正确率都不错。不过他们都是商业化产品，是收费的，并且肯定是不开源的。目前CMU的Openface是比较新的基于机器学习的开源人脸识别系统，集成了OpenCV和Torch（一个开源的机器学习框架），使用Python和Lua语言编写，人脸识别正确率不错。

本课题所基于的系统为ROS（机器人操作系统，Robot Operating System），是专为机器人软件开发所设计出来的一套电脑操作系统架构。ROS的运行架构是一种使用ROS通信模块实现模块间P2P的松耦合的网络连接的处理架构，它执行若干种类型的通讯，包括基于服务的同步RPC（远程过程调用）通讯、基于Topic的异步数据流通讯，还有参数服务器上的数据存储[11]。

作为一个开源软件，ROS满足了广大开发者的需求，得到了广泛的应用。不少研究人员基于ROS开发出了许多室内建图、导航避障等第三方软件包，使得ROS平台的生态更加丰富。截至目前， ROS核心程序的第六版已经发布，每年也会更新一到两个版本。如今已有多达95种机器人是基于ROS二次开发的，其范围涵盖移动机器人、机械手、类人机器人、无人车和无人飞行器等热点领域，并且ROS对大部分传感器都有比较好的支持。基于ROS开发的应用程序甚至有2000多种，涵盖了硬件驱动、模拟仿真、运动规划、运动控制、环境感知等各个方面。ROS已经逐步成为机器人研发领域的通用性软件平台[12]。

## 1.3 本文所做工作

本文通过对人脸检测/识别和ROS系统的研究，使用Python语言开发出了一套实时的人脸检测与识别系统。

首先本文对人脸检测/学习/识别进行了研究，安装了Python环境，机器学习框架torch、科学计算库numpy和scilearn，学习了相关的开源的人脸检测/学习/识别框架Openface，安装了该框架的运行环境。通过阅读框架源码，本文结合开发环境构建了一套自己的人脸检测/学习/识别模块。

然后本文对ROS系统进行了研究，安装了ROS的运行和开发环境，并安装了图像处理库opencv，了解了ROS系统的架构，通信机制和opencv的接口使用规则。构建了ROS节点通信模块，成功地从摄像头采集数据，并通过ROS的发布者/订阅者模式（Publisher/Subscriber）将数据传输到指定位置。

最后，本文将人脸检测/学习/识别模块、ROS通信模块集成，开发出了一套实时的人脸检测与识别系统，有B/S架构和C/S架构的两种实现。用户可通过网页端访问，也可以通过本地客户端访问，使用系统功能。除实时的人脸检测/识别功能外，用户还可以导入人脸图片，上传至系统进行学习处理。用户也可以对某一个学习过的人的人脸进行继续学习以增加识别率。也可以删除某个学习过的人的人脸数据。另外，用户还可以控制检测/识别/学习的处理频率。

## 1.4 论文构成

第一章描述本课题背景及目的、国内外研究状况、本文所做工作以及论文构成。

第二章介绍本系统使用到的关键技术和工具。

第三章进行系统需求分析。

第四章介绍系统总体设计。

第五章介绍系统详细设计与实现。

第六章介绍系统测试情况。

第七章总结本文、遇到的问题以本文的展望。

# 关键技术和工具

1. Python

本系统使用Python语言开发系统后台，Python是一种广泛使用的高阶编程语言。作为一门解释型语言，Python的设计理念之一是强调代码的可读性（特别是使用空格缩进来分隔代码块而不是花括号或关键字）。Python自身具有丰富和强大的库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。

1. Git

使用Git从远程代码库下载代码。Git是一个开源的分布式版本控制系统，可以高效地处理项目的版本管理

1. Ubuntu

Ubuntu是一个以桌面应用为主的开源GNU/Linux操作系统，Ubuntu是基于Debian GNU/Linux，支持x86、amd64（即x64）和ppc架构。Ubuntu的目标在于为一般用户提供一个最新的、同时又相当稳定的主要由自由软件构建而成的操作系统。Ubuntu具有庞大的社区力量，用户可以方便地从社区获得帮助。

1. OpenFace

OpenFace是一个开源免费的基于深度神经网络的人脸识别框架。

1. OpenCV

OpenCV 是一个广受欢迎的开源计算机视觉库， 它提供了很多函数，实现了很多计算机视觉算法，算法从最基本的滤波到高级的物体检测皆有涵盖。

1. NumPy

NumPy（Numeric Python）提供了许多高级的数值编程工具，如：矩阵数据类型、矢量处理，以及精密的运算库。专为进行严格的数字处理而产生。

1. Torch

Torch是一个广泛支持机器学习算法的GPU优先的科学计算框架。高效且易于使用，这得益于一个方便快捷的脚本语言，LuaJIT和底层的C/CUDA实现。

1. sci-learn

sci-learn也是一个开源的Python的机器学习框架， 它是一个简单有效的数据挖掘和数据分析工具，基于NumPy、SciPy，matplotlib。

1. ROS

机器人操作系统（ROS）是一个灵活的编写机器人软件的框架。是一个为了简化在各种各样的机器人平台创建复杂、强大的机器人行为的任务的工具、库和规范的集合。

1. rospy

rospy是一个纯的面向ROS的Python客户端库。rospy的客户端API使得Python程序员快速接入ROS的话题、服务和参数。rospy的设计有利于运行时性能的实现速度（即开发时间），所以算法可以在ROS上快速被原型化和测试。它也是非关键路径代码的理想选择，例如配置和初始化代码。许多ROS工具是用rospy编写的，以利用类型检查功能。许多ROS工具，如rostopic和rosservice，都是建立在rospy之上的。

1. Qt

Qt是一个跨平台C++图形用户界面应用程序开发框。Qt是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）以及一些宏，易于扩展，允许组件编程。

1. Qt Designer

Qt Designer是Qt用于从Qt组件设计和构建图形用户界面（GUI）的工具。

1. PySide

PySide项目是Qt提供LGPL许可的Python绑定。它还包括用于快速生成任何基于Qt的C ++类层次结构的绑定的完整工具链。PySide Qt绑定允许免费开放源代码和专有软件开发，最终旨在支持Qt平台。

1. Bootstrap

Bootstrap来自Twitter，是目前很受欢迎的前端框架。Bootstrap 是基于HTML、CSS、JavaScript的，它简洁灵活，使得 Web 开发更加快捷。它是一个CSS/HTML框架，提供了优雅的HTML和CSS规范。

1. jQuery

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。它封装了JavaScript常用代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。

1. WebSocket

WebSocket协议是基于TCP的一种新的网络协议。它实现了浏览器与服务器全双工(full-duplex)通信，允许服务器主动发送信息给客户端。

1. FileReader

FileReader允许Web应用程序异步读取存储在用户计算机上的文件（或原始数据缓冲区）的内容，使用File或Blob对象指定要读取的文件或数据。其中File对象可以是来自用户在一个input元素上选择文件后返回的FileList对象,也可以来自拖放操作生成的 DataTransfer对象,还可以是来自在一个HTMLCanvasElement上执行mozGetAsFile()方法后返回结果。

1. HTML5 Canvas

Canvas最早由Apple引入WebKit，用于Mac OS X的 Dashboard，后来又在Safari和Google Chrome被实现。基于Gecko 1.8的浏览器,比如 Firefox 1.5，同样支持这个元素。canvas元素是WhatWG Web applications 1.0规范的一部分,也包含于HTML 5中。

# 系统需求分析

## 3.1功能需求分析

本系统包括人脸检测、人脸学习、人脸识别以及导入图片学习等功能。功能如下：

1. 人脸检测

在摄像头捕获的图像，标记出所有出现的人脸，并具有人脸跟踪功能。

1. 人脸学习

用户先输入ID和名字，系统对当前的人脸图像进行学习，并将训练数据保存下来。对于已经学习过的人脸，可以选择继续学习该人脸或者直接删除。

1. 人脸识别

系统对当前人脸图像进行识别，给出识别结果（用户的名字）。

1. 导入图片学习

针对用户可能不在当前摄像头下的情况，系统支持导入多张同一个人的人连图片进行学习，并将训练数据保存下来。

## 3.2 用例需求分析

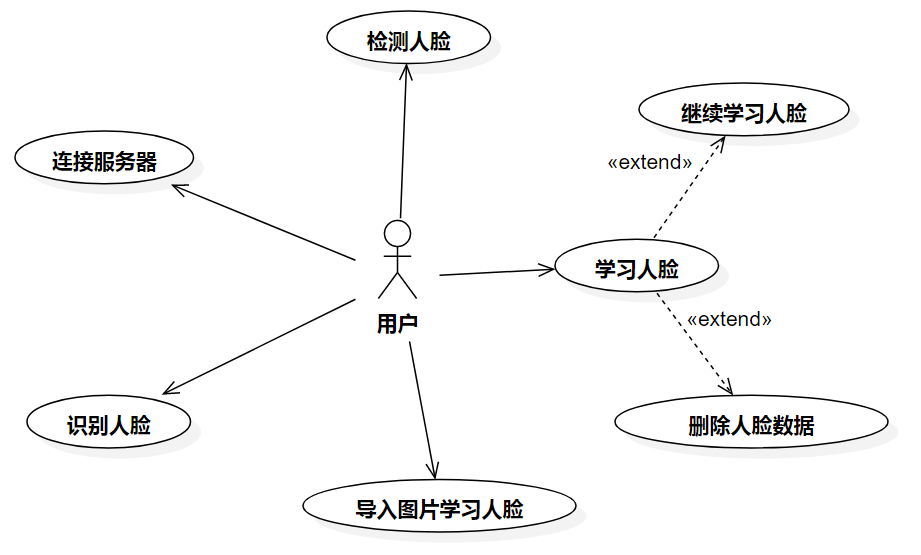


图3. 1系统用例图

表3. 1系统用例列表

| 序号 | 用例 | 描述 |
| --- | --- | --- |
| 1 | 连接服务器 | 用户输入IP地址连接至服务器 |
| 2 | 人脸检测 | 标记出所有的人脸，并持续跟踪 |
| 3 | 人脸学习 | 用户提供输入ID和名字，系统对当前的人脸图像进行学习，并将训练数据保存下来 |
| 4 | 继续学习人脸 | 对于已经学习过的人脸，继续学习以增加识别率 |
| 5 | 删除指定人脸 | 删除指定人脸，之后系统会识别为Unknown（无名氏） |
| 6 | 人脸识别 | 系统对当前人脸图像进行识别，将检测到的最大的人脸标记出来，并给出对应人脸的识别结果（用户的名字） |
| 7 | 导入图片学习 | 导入多张同一个人的人脸图片进行学习，并将训练数据保存下来 |

## 3.3 业务流程分析

* B/S架构系统具体业务流程分析如下：

1. 连接服务器

用户输入IP地址，连接到服务器。若输入格式错误，系统会给出提示。直到格式正确为止。

1. 人脸检测

在系统连上服务器后，用户点击“人脸检测”按钮，系统的图像区则会开始标记人脸，多个人脸会分别标记，并实时跟踪。若无人脸，则不会有任何标记。点击 “停止”按钮，系统则会停止标记人脸。

1. 人脸学习

用户先输入ID和名字，系统对当前的人脸图像进行学习，并将训练数据保存下来。ID和名字的格式均正确才会开始学习人脸，且在学习过程中不能更换ID和名字。点击“停止”按钮，系统会停止学习人脸。

1. 继续学习人脸

用户在人脸库列表里面找到对应人脸信息，点击“继续学习”按钮，系统对当前的人脸图像进行学习，并将训练数据保存下来。点击“停止”，系统会停止继续学习人脸。

1. 删除指定人脸

用户在人脸库列表里面找到对应人脸信息，点击“删除”按钮，经用户确认删除后，系统会删除人脸库中对应的数据。

1. 人脸识别

点击“人脸识别”按钮，系统对当前人脸图像进行识别，给出识别结果（用户的ID和名字）。点击“停止”按钮，系统会停止识别人脸。

1. 导入图片学习

点击“导入图片学习”按钮，跳入导入图片学习页面，用户先输入ID和名字，选择多张同一个人的人脸图片，点击“上传”按钮，系统对上传的图片进行学习，并实时返回进度信息。

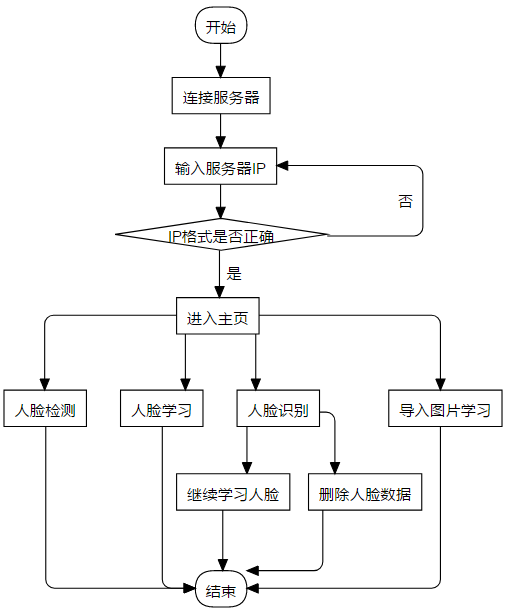


图3. 2 B/S架构流程图

* C/S架构系统具体业务流程分析如下：

1. 人脸检测

在系统客户端启动后，用户点击“人脸检测”按钮，系统的图像区则会开始标记人脸，多个人脸会分别标记，并实时跟踪。若无人脸，则不会有任何标记。点击 “停止”按钮，系统则会停止标记人脸。

1. 人脸学习

用户先输入ID和名字，系统对当前的人脸图像进行学习，并将训练数据保存下来。ID和名字的格式均正确才会开始学习人脸，且在学习过程中不能更换ID和名字。

1. 继续学习人脸

用户在人脸库列表里面找到对应人脸信息，点击“继续学习”按钮，系统对当前的人脸图像进行学习，并将训练数据保存下来。

1. 人脸识别

点击“人脸识别”按钮，系统对当前人脸图像进行识别，给出识别结果（用户的ID和名字）。

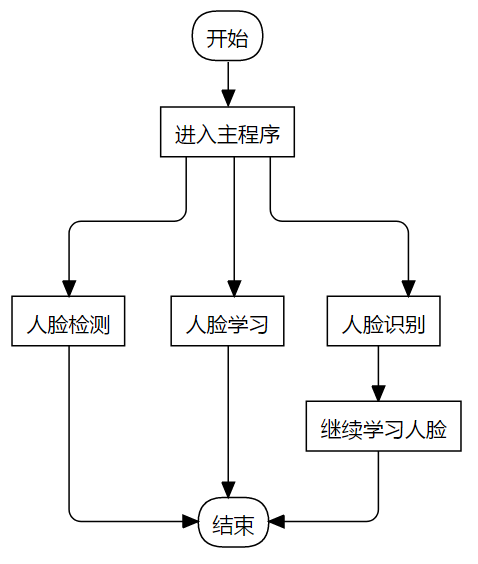


图3. 3 C/S架构业务流程图

# 系统总体设计

## 4.1 B/S架构系统设计

本系统由四部分组成：人脸检测/学习/识别的核心模块、ROS节点通信模块、WebSocket服务器模块、前端呈现模块。

核心模块具有检测人脸，学习人脸，识别人脸的功能。人脸检测部分使用到了dlib的库检测人脸。人脸学习部分对人脸图像进行预处理后，将数据交给torch框架进行学习，最后对这一帧人脸图像生成特征数据，并把数据保存到人脸库中。人脸识别部分会捕获摄像头前的人脸，预处理后交给torch框架进行学习，将学习出来的数据与人脸库中的数据进行对比，最后生成一个预测结果，从而达到识别人脸的目的。

对于ROS节点通信模块，它承担了从ROS节点网络中获取到摄像头节点采集的图像的责任，摄像头节点作为消息发布者（Publisher），将消息发布给对此话题（Topic）感兴趣的订阅者（Subscriber），然后订阅者节点将图像数据共享至WebSocket服务器。这样，数据就从ROS节点网络流向了WebSocket服务器模块，以便于进行进一步的处理。

WebSocket服务器模块会处理浏览器端发来的JSON格式的请求，根据请求内容的不同，返回不同的响应。本模块集成了核心模块，对ROS节点网络发来的图像进行处理，返回给浏览器端。

前端呈现模块使用Bootstrap框架编写用户交互的界面， 使用jQuery进行逻辑代码的编写，浏览器与WebSocket服务器之间建立若干条WebSocket连接，根据用户点击，发出不同的请求数据，根据WebSocket服务器的响应，将数据呈现在浏览器上。

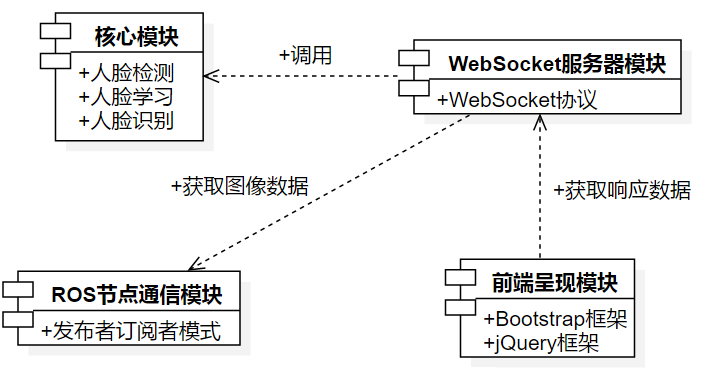


图4. 1 B/S架构系统设计图

## 4.2 C/S架构系统设计

本系统由四部分组成：人脸检测/学习/识别的核心模块、ROS节点通信模块、Qt客户端模块。

核心模块和ROS节点通信模块同B/S架构系统。只不过ROS节点网络的数据发送到了Qt客户端模块。

Qt客户端模块使用跨平台的Qt GUI框架编写，本模块集成了核心模块，对图像数据进行处理，使用Qt内建的Signal/Slot（信号和槽）通信机制更新自身的GUI组件。

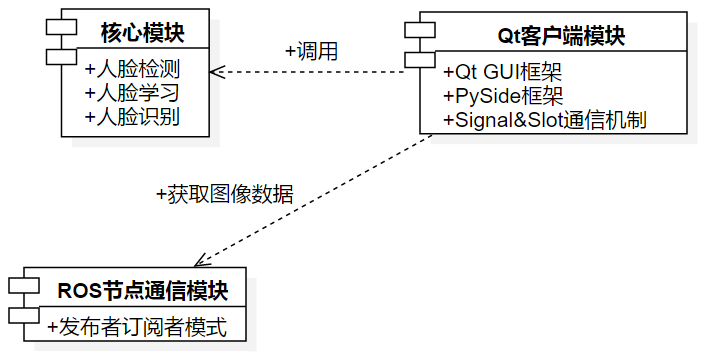


图4. 2 C/S架构系统设计图

# 系统详细设计与实现

## 5.1 公共模块详细设计与实现

### 5.1.1 人脸检测/学习/识别模块详细设计与实现

本模块为核心模块，也是公共模块，使用Python语言编写。负责处理人脸图像数据，对外提供接口以供其他模块调用。

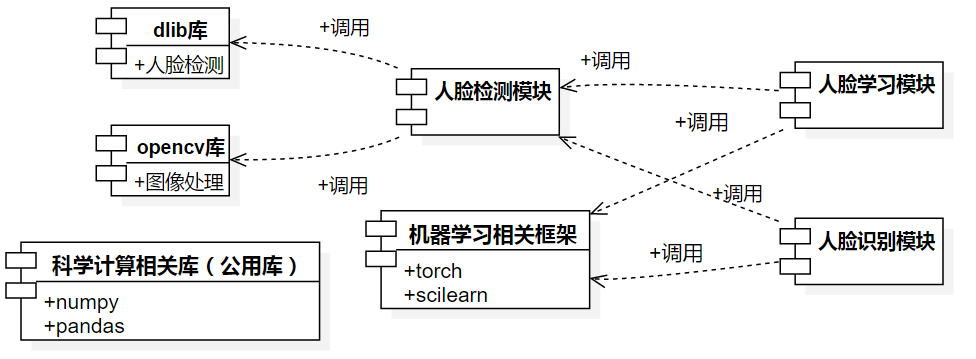


图5. 1核心模块总设计图

1. 人脸检测模块详细设计

本模块提供两个基础接口，以供其他模块调用，它们的作用分别是检测出图像中所有人脸的位置和检测出图像中最大的人脸的位置。

以下为人脸检测模块接口设计：

表5. 1接口getAllFaceBoxes

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | getAllFaceBoxes(*npImg*, *detector*='dlib') |
| 作用 | 返回所有检测到的人脸在图像中的位置 |
| 参数npImg | numpy的数组（处理后的图像数据） |
| 参数detector | 使用的检测器（默认为dlib库的detector） |
| 返回值 | 为Python数组，是所有检测到的人脸在图像中的位置。如果没有检测到人脸则返回空数组 |

表5. 2接口getLargestFaceBox

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | getLargestFaceBox(*npImg*) |
| 作用 | 返回检测到的最大的人脸在图像中的位置 |
| 参数npImg | numpy的数组（处理后的图像数据） |
| 返回值 | 检测到的最大的人脸在图像中的位置，类型为Python数组 |

1. 人脸学习模块详细设计

本模块内部维护了两个字典：images和persons。images以裁剪、处理后的人脸图像的哈希值作为键（保证图像的唯一性），以经过torch框架学习后的图像的特征数组（numpy数组）为值。persons以用户的ID作为键，名字作为值。这两个字典共同作为人脸库，人脸识别模块在生成预测结果时，会使用到这个人脸库。在主程序（正常或非正常）结束之后，会将这两个字典序列化并存储到本地的文件系统中。下次模块被载入内存中时，会从本地文件系统中读取文件，将其反序列化，并更新到images和persons字典中。其中序列化/反序列化技术用到了Python自带的pickle库，生成的文件格式为.pkl。

本模块的学习接口以学习人的ID，名字和人脸图像作为参数，返回检测到的最大的人脸在图像中的位置。首先将原始图像数据转换成numpy数组，按规则更新persons字典，然后调用人脸检测模块的getLargestFaceBox接口得到人脸位置，之后对人脸进行裁剪处理，交给torch进行学习，最后以裁剪处理后的图像的哈希值为键，学习得到的特征数据为值，更新images字典。

以下为人脸学习模块接口设计：

表5. 3接口initData

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | initData() |
| 作用 | 在模块载入时，初始化人脸库数据 |
| 参数 | 无 |
| 返回值 | 无 |

表5. 4接口saveData

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | saveData() |
| 作用 | 在主程序退出时，及时的将人脸库数据写入到本地文件系统中 |
| 参数 | 无 |
| 返回值 | 无 |

表5. 5接口trainFace

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | trainFace(*name*, *identity*) |
| 作用 | 该模块核心方法，用于人脸学习 |
| 输入参数 | name：用户的名字，用于用户的身份标识  identity：用户的ID，用于确认用户的唯一性 |
| 返回值 | 检测到的最大的人脸在图像中的位置，类型为Python的list |

1. 人脸识别模块详细设计

本模块在载入时，需要读取人脸学习模块的人脸库的数据，以此来训练识别器（Recognizer），训练可选多种模型。首先，模块读取原始的人脸图像数据，调用人脸检测模块的接口，裁剪、处理出最大的人脸图像，将其交由torch框架学习。最后调用识别器（Recognizer）的预测接口，对当前生成的人脸特征数据进行预测，生成预测结果。

另外，模块需要提供更新识别器（Recognizer）的接口，即重新训练识别器（Recognizer）。

以下为人脸识别模块接口设计：

表5. 6接口recognizeFace

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | recognizeFace() |
| 作用 | 该模块核心方法，用于人脸识别 |
| 输入参数 | 无 |
| 返回值 | 包含检测到的最大的人脸在图像中的位置、预测出来的人的标识（名字和ID），类型为Python的dict |

表5. 7接口updateRecognizer

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | updateRecognizer() |
| 作用 | 重新训练人脸识别用的识别器（Recognizer） |
| 输入参数 | 无 |
| 返回值 | 无 |

### 5.1.2 ROS通信模块详细设计

ROS是一种分布式结构的操作系统，开发者可以在其上部署各种节点，如传感器节点、摄像头节点等。本模块负责ROS节点之间的通信，将摄像头节点采集到的图像数据传输到其他模块。由于图像数据量大，传输频率高，所以模块内部要控制相关参数，让传输更流畅。本模块中节点与节点之间的通信机制采用ROS内建的发布者/订阅者（Publisher/Subscriber）模式，发布者节点可以在某个话题（Topic）上发布消息，对此话题（Topic）感兴趣，订阅了该话题的订阅者则会等待、接受消息。

而ROS Master节点作为消息传递的中介人，它管理着节点（Node）信息、话题（Topic）信息，负责将发布者（Publisher）发布的消息转发给订阅了指定话题（Topic）的订阅者。所以在所有节点初始化之前，要先运行ROS Master。

首先，摄像头节点会循环请求摄像头，直到请求成功。之后会初始化发布者节点，定义话题（Topic），消息类型，消息队列长度等。最后不断获取摄像头数据，包装成一个消息体，发布出去。消息体的类型是ROS內建的CompressedImage类型。

然后，订阅者节点被初始化，定义话题（Topic）、消息类型、处理消息的回调函数（callback）、消息队列长度（queue\_size），缓冲区大小（buff\_size）等。消息类型与发布者节点的消息类型同为CompressedImage。buff\_size应该设置为queue\_size \* 消息体平均大小，目的是使得节点通信传输更为稳定。

另外，由于本模块涉及到对同一个文件的I/O操作，所以需要用到文件锁，在文件写入时，其他人不能对其读写。具体实现是用的Python的fcntl库。

ROS通信模块接口设计：

表5. 8接口getCamera

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | getCamera(camId) |
| 作用 | 获取摄像头接口，得到图像数据 |
| 参数camId | USB摄像头的设备ID |
| 返回值 | 无 |

表5. 9接口imgPub

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | imgPub() |
| 作用 | 发布图像数据 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 10接口saveImg

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | saveImg() |
| 作用 | 接收并保存图像数据 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 11接口callback

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | callback(rosData) |
| 作用 | 每当接收到一个消息时，调用callback回调函数，处理消息 |
| 参数rosData | rosData：接收到的消息体 |
| 返回值 | 无 |

### 5.1.3 工具类模块详细设计

本模块中的工具类函数是通用的，可以被所有模块调用，是一些实用函数。

工具类模块接口设计：

表5. 12接口bytes2NpImg

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | bytes2NpImg(*bytes*) |
| 作用 | 将字节数组转化成numpy数组，用于图像文件数据的预处理 |
| 参数bytes | 待处理的字节数组 |
| 返回值 | numpy数组类型的图像数据，是可以直接参与科学计算的格式 |

表5. 13接口npImg2Bytes

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | npImg2Bytes(*npImg*) |
| 作用 | 将numpy数组转化成字节数组，用于图像文件数据的传输 |
| 参数npImg | 待处理的numpy数组 |
| 返回值 | 字节数组类型的图像数据，是可以直接数据传输的格式 |

表5. 14接口window.String.prototype.format

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | window.String.prototype.format() |
| 作用 | 利用带有占位符的模板字符串快速生成指定的字符串 |
| 参数 | 无 |
| 返回值 | 指定格式的字符串 |

## 5.2 B/S架构系统详细设计与实现

### 5.2.1 WebSocket服务器模块详细设计与实现

本模块集成了核心模块，会解析前端呈现模块发来的WebSocket请求，根据请求内容，调用核心模块的接口，返回相应的响应数据，消息格式为JSON格式。

为了使消息传输更流畅，本模块用到了两个WebSocket服务器（图像处理服务器和图像传输服务器），两个服务器均支持WebSocket协议，具体实现的技术是autobahn的WebSocket协议库，本文可以用它方便地创建WebSocket服务器，只需要扩展autobahn的基础类WebSocketServerProtocol，重写其中的几个回调函数即可。

其中图像处理服务器专职提供人脸检测、学习、识别、导入图片学习等服务。WebSocket服务器模块和前端呈现模块之间会订立一套请求与响应的消息体规范，格式为JSON。根据浏览器发来的消息体，调用核心模块提供服务。

图像传输服务器专职提供传输原始图像数据的服务，由于浏览器请求该服务的频率高（实时显示摄像头图像，延迟不能太高），服务器响应返回的消息体很大（实时的摄像头图像数据），所以本文决定将此服务分离出来，单独在一个WebSocket服务器上运行。这样不会影响到图像处理服务的使用。

WebSocket服务器模块接口、类设计：

表5. 15接口startServer

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | startServer() |
| 作用 | 在指定端口开启WebSocket服务器 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 16类MyServerProtocol

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类 | MyServerProtocol类 |
| 作用 | 重写autobahn的WebSocketServerProtocol的回调函数，定义服务器的各种行为 |
| 父类 | autobahn.twisted.websocket.WebSocketServerProtocol |
| 重写方法 | onConnect、onOpen、onMessage、onClose |

表5. 17类接口MyServerProtocol.onMessage

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类接口方法 | MyServerProtocol.onMessage(*self*, *payload*, *isBinary*) |
| 作用 | 重写autobahn库的WebSocketServerProtocol的回调函数，定义服务器收到消息后的行为 |
| 参数self | 对实例对象的引用 |
| 参数payload | WebSocket客户端发来的数据 |
| 参数isBinary | 是否为二进制数据 |
| 返回值 | 无 |

表5. 18消息体MSG规范

|  |  |
| --- | --- |
| 名称 | 内容 |
| 代码段 | WebSocket服务器返回至浏览器的消息体规范 |
| 作用 | 规范WebSocket服务器与WebSocket客户端之间的消息传输 |
| 具体定义 | 见附录A |

### 5.2.2 前端呈现模块详细设计与实现

本模块使用Bootstrap栅格系统进行布局，jQuery对DOM进行逻辑处理。使用JavaScript內建的WebSocket对象与服务器进行交互。

1. 连接模块

需要用户输入WebSocket服务器的IP地址，然后跳转到主页面index.html。本文使用了localStorage存储正确格式的IP地址，使用正则表达式确保IP地址格式正确。



图5. 2连接模块用户界面

连接模块接口设计：

表5. 19接口validateIP

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | validateIP(*ip*) |
| 作用 | 验证IP地址格式是否正确 |
| 参数ip | 地址字符串 |
| 返回值 | 类型为window.Boolean，格式正确返回true，其他情况返回false |

1. 主页模块

首先本模块会调用localStorage的getItem方法，取得WebSocket服务器的IP地址，然后与之建立WebSocket连接，周期性的向其发出请求，根据其响应做出对应的逻辑操作。请求的消息体遵循之前订立的请求/响应规范。

与图像传输服务器连接的WebSocket对象在收到图像数据时，将其渲染到HTML页面中的img标签中，而与图像处理服务器连接的WebSocket对象在收到发回的数据时，将调用HTML页面中的canvas标签的context对象进行图形绘制，比如在检测到的人脸的位置绘制矩形，打印预测人脸的名字等。

网页布局排版时，img元素和canvas元素被设置为长度、宽度分别相同，位置重合。这样就可以同时显示原始图像和人脸检测/学习/识别的信息。



图5. 3主页模块页面设计

主页模块设计：

表5. 20接口createWebSocket

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | createWebSocket(*host*, *port*) |
| 作用 | WebSocket对象的工厂方法 |
| 参数host | 服务器IP地址 |
| 参数port | 服务器端口 |
| 返回值 | ws对象实例的引用，类型为window.WebSocket |

表5. 21接口decorateSocket

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | decorateSocket(*socket*) |
| 作用 | WebSocket对象的装饰器方法，重写回调方法 |
| 参数socket | WebSocket对象实例 |
| 返回值 | ws对象实例的引用，类型为window.WebSocket |

表5. 22接口requestOnce

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | requestOnce(*socket*, *msg*) |
| 作用 | 向服务器发出一次请求 |
| 参数socket | ws（WebSocket）对象实例 |
| 参数msg | 消息体对象 |
| 返回值 | 无 |

表5. 23接口requestLoop

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | requestLoop(*socket*) |
| 作用 | 周期性向服务器发出请求（请求内容可能会发生改变） |
| 参数socket | ws（WebSocket）对象实例 |
| 返回值 | 无 |

表5. 24接口reuqestFixedLoop

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | requestFixedLoop(*socket*, *msg*) |
| 作用 | 周期性向发出请求（请求内容不会发生改变），用于请求原始图像数据 |
| 参数socket | ws（WebSocket）对象实例 |
| 参数msg | 消息体对象 |
| 返回值 | 无 |

表5. 25接口handleMsg

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | handleMsg(*e*) |
| 作用 | 处理WebSocket收到的消息 |
| 参数e | 收到的消息对象 |
| 返回值 | 无 |

表5. 26接口detect

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | detect() |
| 作用 | 检测人脸，将requestLoop方法请求的内容变成检测人脸 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 27接口train

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | train(*identity*, *name*) |
| 作用 | 开始人脸学习，将requestLoop方法请求的内容变成学习人脸 |
| 参数identity | 当前正在学习的用户的ID |
| 参数name | 当前正在学习的用户的名字 |
| 返回值 | 无 |

表5. 28接口validateNameId

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | validateNameId(*name*, *identity*) |
| 作用 | 验证人脸学习时发送的用户ID和名字 |
| 参数identity | 当前正在学习的用户的ID |
| 参数name | 当前正在学习的用户的名字 |
| 返回值 | 类型为bool，格式正确返回true，其他情况返回false |

表5. 29接口stopTrain

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | stopTrain() |
| 作用 | 停止人脸学习，将requestLoop方法请求的内容变成检测人脸 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 30接口recognize

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | recognize() |
| 作用 | 识别人脸，将requestLoop方法请求的内容变成识别人脸 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 31接口doNothing

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | doNothing() |
| 作用 | 停止使用服务，将requestLoop方法请求的内容变成空消息，清除canvas上的内容 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 32接口delPerson

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | delPerson(thiz, identity) |
| 作用 | 删除人脸库中对应ID的人脸数据 |
| 参数thiz | DOM对象的引用 |
| 参数identity | 用户ID |
| 返回值 | 无 |

表5. 33接口fitPersonsTable

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | fitPersonsTable(*persons*) |
| 作用 | 将人脸信息以表格的形式显示在网页中 |
| 参数persons | 人脸信息对象 |
| 返回值 | 无 |

表5. 34接口createCanvasCtx

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | createCanvasCtx() |
| 作用 | HTML5 Canvas Context对象的工厂方法 |
| 输入参数 | 无 |
| 返回值 | 无 |

表5. 35接口clearCanvas

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | clearCanvas() |
| 作用 | 清除canvas元素上的内容 |
| 输入参数 | 无 |
| 返回值 | 无 |

1. 导入图片学习模块

本模块与WebSocket服务器会重新建立一条连接，用户需要输入待学习的人脸ID、名字，从本地选择多张人脸图片（至少15张，以便充分学习），上传至服务器。使用FileReader类读取文件，包装成消息体，通过WebSocket连接传输至服务器。上传过程中实时显示进度。



图5. 4导入图片学习模块页面设计

导入图片学习模块设计：

表5. 36接口createFileReader

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | createFileReader() |
| 作用 | FileReader对象的工厂方法 |
| 输入参数 | 无 |
| 返回值 | 被装饰过的FileReader对象实例 |

表5. 37接口read

|  |  |
| --- | --- |
| 名称 | 内容 |
| 接口方法 | read(*fr*, *f*) |
| 作用 | 将单个文件的内容载入到FileReader的缓冲区中 |
| 参数fr | FileReader对象实例的引用 |
| 参数f | File实例对象的引用 |
| 返回值 | 无 |

## 5.3 C/S架构系统详细设计与实现

### 5.3.1 Qt客户端模块详细设计与实现

本模块负责用户界面的交互，内部集成了核心模块，使用跨平台GUI框架PySide（Qt在Pyhton语言上的绑定）开发前台界面，由于界面层次结构较为复杂，为了快速有效率的开发以及将开发重点放在客户端逻辑代码编写上，本文选取Qt Designer作为开发环境快速生成UI文件。将UI文件生成对等的Python脚本需要调用命令pyside-uic，为了避免繁琐操作，本文将其过程抽取成为一个函数以被调用。

主界面类MainWidget继承自Qwidget，由于主线程执行耗时操作会阻塞整个程序的运行，而在非UI线程更新UI组件会抛出异常。所以本文会使用Qt內建的通信机制Signal/Slot（信号和槽），将各种事件发出的信号（signal）与处理信号的槽函数（slot）连接起来，这样就能在子线程发出相应信号，间接地更新UI组件了。MainWidget类编写了多个槽函数，用于提供人脸检测/学习/识别、更新UI组件等服务。

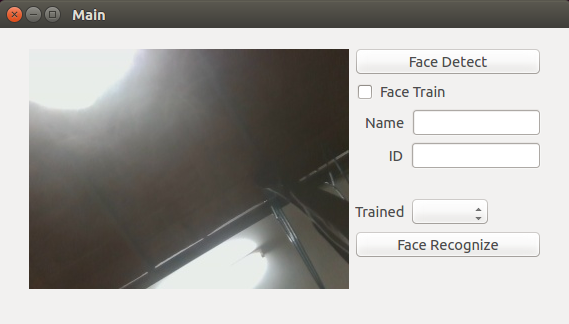


图5. 5Qt客户端模块界面设计

Qt客户端模块接口和类设计：

表5. 38类MainWidget

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类 | MainWidget |
| 作用 | 主界面类，负责承载各种UI组件 |
| 父类 | PySide.QtGui.QWidget |
| 重写方法 | \_\_init\_\_ |

表5. 39类成员字段MainWidget.imgSig

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员字段 | MainWidget.imgSig |
| 作用 | MainWidget内置的自定义信号，用于传输图像数据（PySide.QtCore.QByteArray类型） |
| 类型 | PySide.QtCore.QSignal |

表5. 40类成员方法MainWidget.detectSlot

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.detectSlot(*self*) |
| 作用 | 用于响应人脸检测的槽函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 41类成员方法MainWidget.trainSlot

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.trainSlot(*self*) |
| 作用 | 用于响应人脸学习的槽函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 42类成员方法MainWidget.recognizeSlot

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.recognizeSlot(*self*) |
| 作用 | 用于响应人脸识别的槽函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 43类成员方法MainWidget.imgSlot

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.imgSigSlot(*self*) |
| 作用 | 用于更新UI图像数据的槽函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 44类成员方法MainWidget. personChangeSlot

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.personChangeSlot(*self*) |
| 作用 | 用于更新用户ID和名字的槽函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 45类成员方法MainWidget. fitComboBox

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.fitComboBox(*self*) |
| 作用 | 用于将更新的人脸库的信息显示到界面上 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 46类成员方法MainWidget. emitData

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.emitData(*self*) |
| 作用 | 在子线程中执行，不断发送请求的函数 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 47类成员方法MainWidget. handleMsg

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.handleMsg(*self*, *respMsg*) |
| 作用 | 用于处理调用核心模块返回的消息 |
| 参数self | MainWidget对象引用 |
| 返回值 | 无 |

表5. 48类成员方法MainWidget. validateNameId

|  |  |
| --- | --- |
| 名称 | 内容 |
| 类成员方法 | MainWidget.validateNameId(*self*, *name*, *identity*) |
| 作用 | 用于处理调用核心模块返回的消息 |
| 参数self | MainWidget对象引用 |
| 参数identity | 当前正在学习的用户的ID |
| 参数name | 当前正在学习的用户的名字 |
| 返回值 | 类型为bool，格式正确返回true，其他情况返回false |

# 系统测试

## 6.1 测试背景

信息技术的飞速发展，使软件产品在社会各个领域广泛应用，人们开始更加关注软件产品本身的质量。软件公司为了更快地抢占现有市场，必须把产品质量的维护集成到正常的开发流程大众，以免竞争者行列中落后出局。而软件用户为了顺利完成自己的业务，当然会考虑使用优质的软件。对系统进行测试，可以发现一些潜在的漏洞，提高稳定性。尽早地在项目开发中集成测试，可以有效控制项目质量，开发效率会更快，后期维护会更简单。

## 6.2 测试用例设计

本文将整个系统进行功能测试。按照功能模块划分测试用例，以表格方式呈现。如下所示：

表6. 1 B/S架构系统测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 前置条件 | 执行步骤 | 期望结果 | 实际结果 |
| 1 | 连接上服务器 | 点击人脸检测按钮 | 图像区显示并标记出人脸 | 符合预期 |
| 2 | 连接上服务器 | 点击人脸识别按钮 | 图像区显示并标记出人脸和名字，若人脸库中没有学习这个人脸，则名字显示为Unknown | 符合预期 |
| 3 | 连接上服务器，正在检测或识别人脸 | 点击停止按钮 | 图像区只显示实时图像，不作任何标记 | 符合预期 |
| 4 | 连接上服务器，输入要学习的新人脸的名字 | 点击人脸学习按钮 | 图像区显示并标记出人脸，后台实时学习 | 符合预期 |
| 5 | 连接上服务器，正在学习人脸 | 点击人脸学习按钮旁边的停止按钮 | 图像区只显示实时图像，不作任何标记，并且会刷新已学习人脸的数据，显示在表格区 | 符合预期 |
| 6 | 连接上服务器 | 点击已学习人脸表格区的继续学习按钮 | 图像区显示并标记出人脸，后台实时学习 | 符合预期 |
| 7 | 连接上服务器，正在学习已学习的人脸 | 点击已学习人脸表格区的停止按钮 | 图像区只显示实时图像，不作任何标记，后台停止学习 | 符合预期 |
| 8 | 连接上服务器 | 点击已学习人脸表格区的删除按钮 | 后台会删除对应人脸的学习数据 | 符合预期 |
| 9 | 连接上服务器，点击有关人脸检测/学习/识别的按钮 | 调节处理速率条 | 人脸检测学习识别的处理速率会发生变化，从0.5倍到1.5倍 | 符合预期 |
| 10 | 与服务器的连接断开 | 点击重连按钮 | 浏览器会试图重新连接服务器 | 符合预期 |
| 11 | 进入导入图片学习页面，并连接上服务器 | 输入ID、名字，并选择多张同一个人脸的图片。点击上传按钮 | 实时显示进度信息，后台对图片进行学习 | 符合预期 |
| 12 | 载入主页面后 | 点击导入图片学习按钮 | 进入新页面（导入图片学习页面） | 符合预期 |
| 13 | 进入导入图片学习页面，与服务器的连接断开 | 点击重连按钮 | 浏览器会试图重新连接服务器 | 符合预期 |

表6. 2 C/S架构系统测试用例设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 前置条件 | 执行步骤 | 期望结果 | 实际结果 |
| 1 | 程序启动 | 点击人脸检测按钮 | 图像区显示并标记出人脸 | 符合预期 |
| 2 | 程序启动 | 点击人脸识别按钮 | 图像区显示并标记出人脸和名字，若人脸库中没有学习这个人脸，则名字显示为Unknown | 符合预期 |
| 4 | 程序启动 | 输入ID和名字，并点击人脸学习复选框 | 若格式均正确，则可以进行人脸学习，其他情况则提示格式错误 | 符合预期 |
| 5 | 程序启动，输入要学习的新人脸的ID和名字 | 点击人脸学习复选框 | 图像区显示并标记出人脸，后台实时学习，并且输入框不能进行编辑 | 符合预期 |
| 5 | 程序启动，正在学习人脸 | 点击人脸学习复选框 | 图像区只显示实时图像，不作任何标记，并且会刷新已学习人脸的数据，显示在下拉框中，输入框重新可以编辑 | 符合预期 |

## 6.3 测试分析

针对上述测试用例，本文对系统进行了测试

表6. 3测试bug记录

|  |  |  |
| --- | --- | --- |
| bug编号 | 详细功能模块 | 情况描述 |
| 1 | 人脸学习 | 当已学习的人脸个数小于两个时，人脸识别模块会出错。 |
| 2 | 人脸学习 | 当人脸库中没有数据时，新建人脸学习的隐藏自增ID将为undefined。 |
| 3 | 人脸学习/识别 | 这两个模块存在对共享数据的并发读写操作，如果没有锁机制协调，后台将会抛出异常。 |
|  | WebSocket服务器 | 在浏览器与服务器连接未完全建立时，即WebSocket对象状态为WebSocket.OPEN以外的情况，向服务器发送数据时非法的，系统会抛出异常。 |
| 4 | 导入图片学习 | FileReader对象读取文件流是异步操作，需要控制FileReader对象的状态在FileReader.LOADING以外，否则系统会抛出异常。 |

* B/S结构系统功能测试截图



图6. 1人脸检测功能测试截图



图6. 2人脸学习功能测试截图



图6. 3人脸识别功能测试截图（该人脸已被学习）



图6. 4人脸识别功能测试截图（该人脸已被学习）

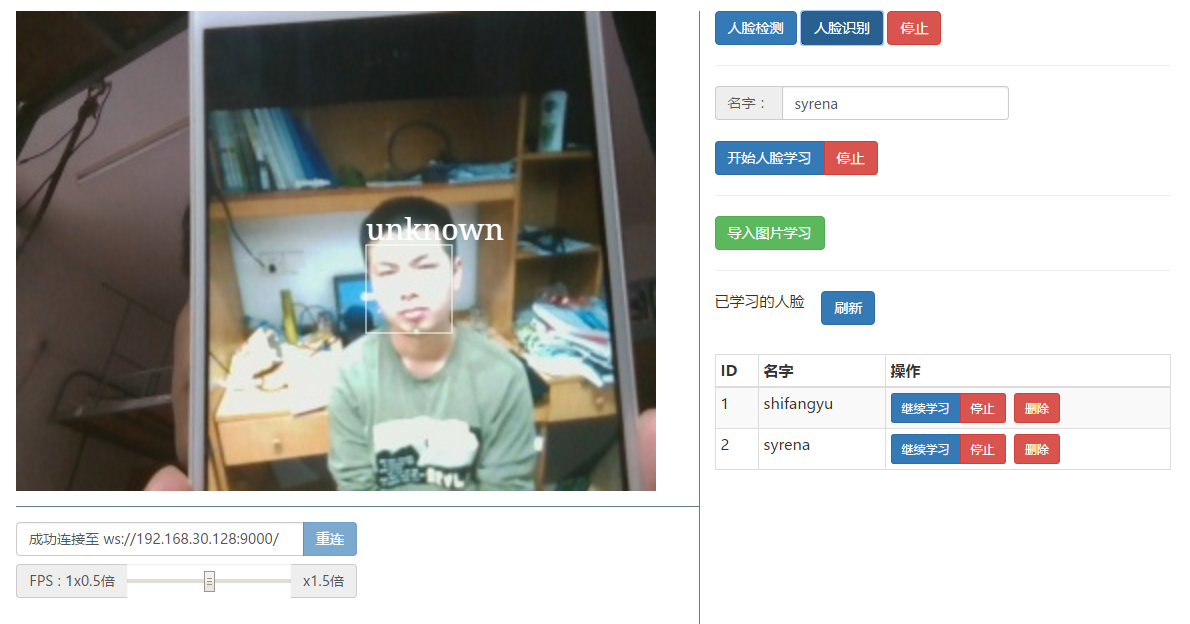


图6. 5人脸识别功能测试截图（该人脸未被学习）



图6. 6导入图片学习功能测试截图



图6. 7导入图片学习功能测试截图（显示damme已经被学习过）



图6. 8导入图片学习功能测试截图（人脸识别结果）

* C/S结构系统功能测试截图

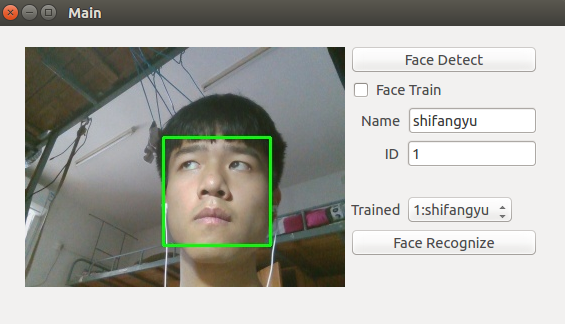


图6. 9人脸检测功能测试截图

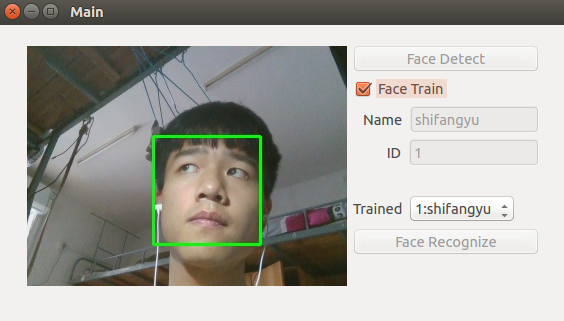


图6. 10人脸学习功能测试截图

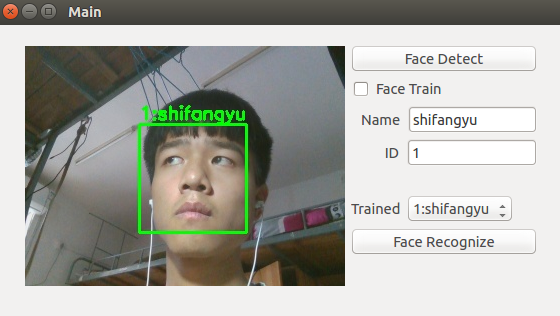


图6. 11人脸识别功能测试截图

# 总结和展望

本文通过对人脸检测/识别技术和ROS系统的研究，使用Python语言开发出了一套实时的人脸检测与识别系统。使用模块化设计，将系统功能划分为若干模块，各个模块之间相对独立，相互调用，降低耦合度，使得系统扩展性强，模块易于升级。系统有B/S架构和C/S架构的实现，在各个应用场景，都能够使用。

然而，系统仍然存在许多不足之处，需要进一步的优化和完善。由于开发时间有限，系统在人脸识别算法方面没有做实质性的优化。如果在未来有能力、精力来维护和完善这个系统，会将人脸检测/学习/识别相关算法的实现独立出来，在其之上，做更深层次的性能优化，增加算法的易用性。其次，C/S架构的客户端程序功能上相对于B/S架构系统有所阉割，不够完善，界面不够美观，有时性能会跟不上。所以，会将C/S架构的系统功能做得更为完善，充分发挥本地客户端程序的优势，进一步从业务逻辑优化代码，提高性能。

参考文献

[1]张利芳.基于PCA算法的人脸识别系统研究[D].中北大学,2015.

[2] 朱兴统,习洋洋.基于C++和OpenCV的人脸识别系统的设计与实现[J].自动化与仪器仪表,2014,(8):127-128,131.DOI:10.3969/j.issn.1001-9227.2014.08.0127.

[3] 王蕾.基于视频序列的人脸识别系统[D].兰州理工大学,2015.DOI:10.7666/d.D711096.

[4] 李文琴,曾广朴,文俊浩等.基于神经网络的人脸识别系统[J].激光杂志,2015,36(3):54-57.DOI:10.14016/j.cnki.jgzz.2015.03.054.

[5]买买提江玉山.基于C++的人脸识别系统的设计与实现[D].西安电子科技大学,2011.DOI:10.7666/d.d215925.

[6] 周玉华.基于微信平台的人脸识别系统研究与实现[D].江苏大学,2016.

[7] https://en.wikipedia.org/wiki/Face\_detection#cite\_note-8[DB/OL].2016

[8] https://en.wikipedia.org/wiki/Torch\_(machine\_learning)[DB/OL].2016

[9] https://en.wikipedia.org/wiki/OpenCV[DB/OL].2016

[10] 黄开宏等.基于ROS户外移动机器人软件系统构建[J].机器人技术与应用,2013,(4):P37-41.

[11]https://zh.wikipedia.org/wiki/%E6%A9%9F%E5%99%A8%E4%BA%BA%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1[DB/OL].2016

[12] http://wiki.ros.org/[EB/OL].2015

致 谢

在论文快要完成之时，我首先要感谢肖雄仁老师，在毕业设计过程的每一周都和同学一起开会，讨论各自目前的工作、遇到的问题以及接下来一周的计划。孜孜不倦的位同学们解答技术问题。为了各位同学开发方便，还为同学们租来毕设用到的相关设备和硬件。为了同学们开发方便，更有效率，老师还特定给我们找了单独的会议室做毕设。

同时还要感谢室友张德威和钱宇航，这两位同学经常和我讨论相关的技术问题，结合所学知识，对相关疑点进行合理的质疑，对遇到的问题提出合理的解决方案。给予了本文作者莫大的帮助。

最后，我还要感谢我的父母，他们一直以来都是默默的支持我、鼓励我。从小到大，他们无时无刻不在关心着我的状态和发展，马上就要步入社会，开始自己人生的另一个起点。希望父母能够自由、快乐地享受生活，而不是天天处于女儿将来发展的忧虑之下。谢谢你们！

# 附录

## 附录A 消息体规范

* 请求消息体规范

|  |  |
| --- | --- |
| 名称 | 内容 |
| NOTHING | 空消息体，服务器收到此请求，不会做任何处理和响应 |
| RAW | 向服务器请求当前帧的摄像头图像 |
| DETECT | 向服务器请求人脸检测服务 |
| TRAIN | 向服务器请求人脸学习服务 |
| STOP\_TRAIN | 向服务器请求停止人脸学习服务 |
| RECOGNIZE | 向服务器请求人脸识别服务 |
| GET\_PERSONS | 向服务器请求当前人脸库的标识信息（名字和ID） |
| DEL\_PERSON | 向服务器请求删除指定人脸的数据 |
| UPLOAD\_PIC | 向服务器上传一张人脸图片 |

* 响应消息体规范

|  |  |
| --- | --- |
| 名称 | 内容 |
| EMPTY | 空消息体，浏览器收到此请求，不会做任何处理 |
| DETECTED\_FACES | 服务器返回检测到的所有人脸的位置信息 |
| RECOGNIZED\_FACE | 服务器返回检测到的最大人脸的位置和名字信息 |
| TRAIN\_STOPED | 服务器告知浏览器人脸学习已经停止 |
| PERSONS | 当前人脸库所有的标识信息（名字和ID） |
| PERSON\_DELETED | 告知浏览器指定人脸数据已经被删除 |
| UPLOADED\_SIZE | 告知浏览器当前导入图片学习的进度 |
| ALL\_PIC\_TRAINED | 告知浏览器所有导入的图片均已学习完成 |