哈爾濱工業大學

毕业设计（论文）

题　　目　基于 Python 实时图像
　　获取及处理软件设计

专　　业　测控技术与仪器

学　　号　140220212

学　　生　刘 阳

指 导 教 师　王 玲

答 辩 日 期　2018 年 6 月

摘　要

随着信息技术的发展、电子设备的普及以及通信技术的进步，人们已经越来越习惯于使用图像这种便捷高效的方式进行交流，图像获取与处理也由此进入了一个高速发展的时期。

现如今，图像处理相关技术在日常生活中随处可见。USB 摄像头因其即插即用的便捷性、相对廉价的成本和不弱的性能，成为最具优势的图像获取设备。而在图像处理方面，人脸识别无疑是当前最热门的领域之一。随着技术的进步，尤其是机器学习、人工智能的突破性发展，人脸识别技术逐渐应用到众多的个人设备上，用于解锁、移动支付，甚至交友相亲等方方面面。

本文是在 Windows 系统下基于 Python 语言开发的图像获取与处理软件，实现了图像实时获取、显示并处理的功能。其中，图像获取使用的便是 USB 摄像头，并且能够摄像头的部分属性参数进行调整以便获取更高质量的数字图像；图像处理部分实现了灰度化、直方图、边缘检测以及人脸检测与识别功能。选用开源的可以跨平台运行的计算机视觉库 OpenCV 来实现以上各种功能；最后使用 GUI 开发库 PyQt 5 整合各功能模块，编写生成可执行的软件供用户使用。

关键词：图像获取；图像处理；人脸识别

Abstract

With the development of information technology, the popularization of electronic equipment and the progress of communication technology, people have become more and more accustomed to the use of image such a convenient and efficient way to communicate, image acquisition and processing technology has also entered a period of rapid development.

Nowadays, image processing technology can be seen everywhere in our daily life. As the most commonly used image acquisition device, cameras are not only covered by streets and lanes, but also come into thousands of households and are often held in the hands of people. USB camera has become the most advantageous image acquisition device because of its convenience, relatively cheap cost and weak performance. About image processing,face recognition is undoubtedly one of the hottest areas. With the progress of technology, especially the breakthrough development of machine learning and artificial intelligence, face recognition technology is gradually applied to personal equipment, which is used in unlocking, mobile payment, even making friends and other aspects.

This paper is about the development of image acquisition and processing software based on Python language in Windows system, realizing the function of image real-time acquisition, display and processing. Among them, image acquisition uses the USB camera, and some properties of the camera can be adjusted to obtain higher quality digital images. The image processing part realizes gray scale, histogram, edge detection and face detection and recognition. OpenCV, an open source, cross-platform computer vision library, is chosen to implement all of these functions. Finally, the GUI development library PyQt 5 is used to integrate various functional modules and write and generate executable software for users to use.

Keywords: image acquisition, image processing, face recognition

目　录

第 1 章 绪　论

1.1 课题背景及研究的目的和意义

随着信息技术的发展、电子设备的普及以及通信技术的进步，人们已经越来越习惯于使用图像这种便捷高效的方式进行交流，图像获取与处理也由此进入了一个高速发展的时期。摄像头作为最常用到的图像获取设备，非但已经遍布大街小巷，更是走进了千家万户、被人们时常捧在手中。USB 摄像头因其即插即用的便捷性、相对廉价的成本和不弱的性能，成为最具优势的图像获取设备。而在图像处理方面，人脸识别无疑是当前最热门的领域之一。

本课题旨在利用 Python 语言，提取 USB 摄像头相关参数及图像，并对数字图像进行灰度化、直方图绘制、边缘检测以及人脸检测和识别等处理。通过在 Windows 操作系统下使用 Python 语言编写的可视化软件，用户可以方便的查看和设置 USB 摄像头的主要参数，并实时监控、保存图像的各特征，着力提高摄像头获取图像的便捷性和实用性，而人脸识别技术的应用，更大大扩展了该软件的应用空间。

1.2 课题国内外研究概况

从图像获取到数字图像处理是一项系统性的工程。

图像获取涉及图像传感器、数据传输与设备通信相关领域息息相关。随着自动化程度的提高，成像设备应用越来越广泛。目前可见光成像所采用的图像传感器主要为 CCD 或 CMOS。虽然 CCD 的技术已经相当成熟，且一般来说噪声相对较低等特点，但近年来 CMOS 有了很快的发展，它具有功耗低、集成度高、价格相对便宜等的特点，且其成像质量已不比 CCD 逊色，因此应用也越来越广[1]。

USB 具有接口简单、传输速度快、通用性强、可扩展性强等特点，在数据传输中应用非常广泛目前 USB3.0 的最大传输带宽已经高达 5.0Gbps，不但超过前一代 USB2.0 速度的十倍以上[2]，同时更加节能，已经逐渐成为硬件厂商的必备接口。采用 USB 作为数据传输通路足以满足 CMOS 图像数据采集系统的要求。摄像头图像存至 USB 接口芯片中后，PC 可以通过 USB 读取 USB 接口芯片中获得的图像数据。

数字图像处理技术是从 20 世纪 60 年代起随着计算机技术和 VLSI（Very Large Scale Integration）产生和不断成熟起来的一个新兴技术领域，它在理论上和实际应用中都取得了巨大的成就[3]。

数字图像处理是一个宽泛的概念。包括包括图像变换、图像压缩与编码、图像增强与复原、图像分割、图像描述、图像识别等内容。

从图像源角度看，视觉图像是最符合人类认知的一种图像，因为视觉在人类感知中扮演着最重要的角色；然而，人类的感知仅限于电磁波谱的视觉波段，与人类不同，成像机器几乎可以覆盖从伽马射线到无线电波的整个电磁波谱，它们可以对非人类习惯的那些图像源进行加工，这些图像包括超声波、电子显微镜和计算机产生的图像。

而从处理角度来看，图像处理止于哪些或其他相关领域（如图像分析和计算机视觉）从哪里开始，目前也并没有一致的看法。从图像处理到计算机视觉的这个连续的统一体内并没有明确的界限。不过、如果从低级、中级和高级处理三个角度来考虑的话，低级处理往往涉及初级操作如降低噪声的图像预处理、对比度增强和图像尖锐化等。低级处理以输入、输出都为图像为特征。中级处理涉及诸多任务，譬如图像分割、边缘检测等。中级图像处理以输入为图像但输出是从这些图像中提取的特征为特点。最后，高级处理涉及“理解”已识别目标的总体，以及在连续统一体的远端执行与视觉相关的认知功能。[4]譬如人脸识别等。

人脸识别技术作为生物特征识别领域中一种基于生理特征的识别，是通过计算机提取人脸特征，并根据这些特征进行身份验证的一种技术[5]。

人脸识别研究主要包括人脸检测、人脸表征、人脸识别、面部表情/姿态分析和生理分类五个方面。常用算法有基于几何特征的人脸识别、基于特征脸的人脸识别、基于模板匹配的人脸识别、基于神经网络的人脸识别基于贝叶斯决策的人脸识别和基于支持向量机的人脸识别等等[6]。各个阶段如表 1-1 所示。

表 1-1 人脸识别发展历史

阶段	1964~1990 年	1991~1997 年	1998 年~现在
主要特征	基于人脸面部几何结构特征方法是主流。	在比较理想条件下的识别问题研究； 基于线性子空间分析和统计学的方法是主流。	重点是非理想条件、用户不配合、大规模数据库的识别问题；3D 建模人脸识别是趋势。
代表性的人脸识别技术与方法及相关关键性事件和作品	已知最早研究论文； 基于剪影分析的人脸识别； 人脸低维表示； 人脸几何结构特征；	特征脸算法（Eigenface）； 局部特征分析法（LFA）； 基于双子空间的贝叶斯概率学习； 基于特征发方法与基于模板的方法对比； Fisherface 人脸识别方法；	光照锥技术； 3D 可变模型； 朗伯反射与线性空间分析 基于 AdaBoost 的人脸检测技术； SVM 用于人脸识别

最近几年，越来越多的人开始研究人脸识别技术，技术、算法越来越成熟，识别人脸和匹配人脸的成功率越来越高。目前有很多互联网企业都在研究人脸识别技术，而且研发了很多产品。

现阶段数字图像处理技术的应用非常广泛，社会许多行业都能看到它的身影。在科学技术的发展状况下，数字图像处理技术将会沿着高分辨率、立体化、高速化、智能化等全方面的方向发展[7]。而本课题用到的相对简单的灰度化、直方图、边缘检测技术，也有大量的成熟算法可以借鉴应用。

最后，本课题采用的软件技术也已相当成熟。Python 语言经过近 30 年的发展，目前官方网站版本已经更新至 Python 3.6.4 和 Python 2.7.14，强大的标准库奠定了 Python 发展的基石，丰富的第三方库保证了 Python 不断发展的[8]。成千上万的第三方库资源，为编程提供了极大的便利，如与本课题相关图像处理和计算机视觉库 OpenCV 和 GUI 库 Tkinter、wxPython、PyQT 等。

1.3 摄像头图像获取及人脸检测与识别的应用

如今，摄像头在人们生产生活中已经随处可见。人脸识别技术也已广泛应用于各个行业，如：楼宇人脸门禁、人脸考勤系统；互联网移动支付终端、交友、相亲终端 APP 系统等。

打开摄像头，刷一下脸，便可以完成支付；从大量的人脸数据库中，公安部门通过人脸识别系统更轻松地找出嫌疑人，如近日警方接二连三的在张学友演唱会中通过人脸识别系统抓获犯罪嫌疑人，如此壮举举国瞩目。这些从侧面反映出这些人脸识别技术如科幻电影一样，已经变成现实。

当然在当下极为火热的人脸考勤系统中，进行“刷脸”考勤，而目前考勤正确率已经达到很高的级别，最典型的就是“刷脸”考勤机，这其实就是一种新型的存储类考勤机，在这之前首先只需以员工人脸图像作为采集对象建立起训练数据库，当员工在上下班考勤时，只需要站在考勤机的识别区域内，考勤机上就会根据输入的图像与自身的训练数据库作为比对，同时也将记录此时的人脸数据并存储。

人脸识别在安防系统中也扮演着不可忽视的作用与价值。人脸识别门禁系统在住所、办公大楼等重要场所也发挥着作用，它利用着人脸识别技术相当于一把“钥匙”作为通行证，这项技术的关键是通过扫描设备将所扫描的人脸图像作为输入人脸图像与预先录入的人脸库比对，如果比对一致将使门禁系统打开，否则关闭，这其实就是“刷脸”来开启或关闭建筑物内的人脸识别门禁系统。同样的近年来许多火车站进站口也改用了人脸识别来代替人工核验身份证。

而从去年 iPhone X 发布以后，更是引爆人脸识别技术在手机等科技产品，人

脸识别解锁几乎成了今年旗舰设备的标准配置。

由此可见人脸识别技术应用前景之广阔。

1.4 本文的主要研究内容

本文是在摄像头图像获取和数字图像处理对相关原理进行广泛研究的基础上，在 Windows 系统下基于 Python 语言开发的图像获取与处理软件。其中，图像获取使用的便是 USB 摄像头，并且能够摄像头部分属性参数进行调整以便获取更高质量的数字图像。而图像处理部分从基础操作，如灰度化、直方图，到边缘检测技术，最终实现人脸检测与识别。针对这些需求，选用开源的可以跨平台运行的计算机视觉库 OpenCV 来实现各种功能。最后使用 GUI 开发库 PyQt 5 整合各功能模块，编写生成可执行的软件供用户使用。

文章结构清晰，第二章介绍相关原理，第三章介绍具体的技术实现方法。每一章内部大致可以区分为图像获取和图像处理两大部分，第三章另外介绍了使用 PyQt 5 进行 GUI 编程和软件打包的相关内容。

第 2 章 实时图像获取及处理的相关原理

2.1 实时图像获取设备

图像采集与处理在实时图像处理系统占有重要地位，是数字图像处理的一个关键的前提。只有实现目标场景图像的视频捕获显示及其数据获取，才能对之进行相应的分析与处理。视频采集是将一个视频流数字化，然后储存在硬盘或其它存储介质上形成一个整块集中的数据块，以供进一步处理。所以本文将首先介绍图像获取的相关内容。

2.1.1 USB 摄像头图像采集原理

当前，几乎所有的图像都是通过光学镜头获取的，而我们更关心的是对图像的数字化采集。捕获图像可以采用视频采集卡，也可使用 USB 接口的数码摄像头，视频采集卡一般提供了特定品牌和型号的动态连接库和开发工具包，价格相对较贵，USB 接口的数码摄像头价格适中，性能尚可。另外，采用 USB 接口的数码摄像头，系统的硬件体系结构比较简洁，无需再配图像采集卡上位机便可直接进行图像数据处理[9]。基于对课题需求的分析，显然 USB 摄像头是最合适的选择。

数字摄像头其实是把感光器件和视频捕捉器件做到了一起，一般是由镜头、感光器件、A/D（模/数转换器）、微处理器、内置存储器和接口（计算机接口、电视机接口）等部件组成。 其中感光器件一般有 CCD 和 CMOS 两种。而 USB 技术由于其便捷的操作, 热插拔形式的支持, 可以实现即用即连, 在多个领域有着广泛应用。

不同品牌不同型号的摄像头功能和性能是各不相同的。在与软件交互、实现对摄像头图像采集的控制时，实际是一操作系统（驱动层）为中介完成的，如图 2-1。因此我们必须关注 USB 摄像头的驱动。

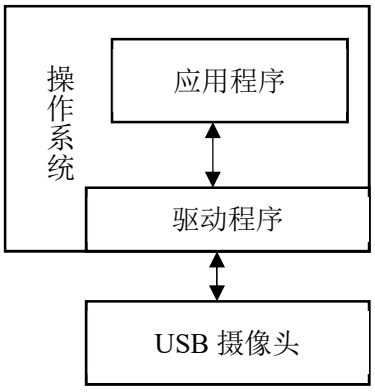


图 2-1 软硬件通信

2.1.2 Windows 操作系统下摄像头的驱动

在本课题中，选择了基于 USB 的普通摄像头，上位机采用 PC 机。在上位机中，系统采用 Windows 操作系统，在 Windows 操作系统下 USB 设备的驱动和视频应用开发都是一项非常复杂的工作。幸运的是，Windows 内核承担了大部分的工作。为了提高系统的稳定性，Windows 操作系统对硬件操作进行了隔离；应用程序一般不能直接访问硬件，程序开发者是通过接口在应用层面，而非硬件层面，来完成相关软件开发。

根据使用驱动程序的类型不同，目前市场上的图像获取设备大致分为两大类：VFW（Video for Windows）和 WDM（Windows Driver Model）。前者是一种趋于废弃的驱动模型，而后者是前者的替代模型。

VFW 是 Microsoft 1992 年推出的关于数字视频的一个软件包。在 Windows 9x 及以上版本系统中，当用户在安装 VFW 时，安装程序会自动地安装配置视频所需要的组件，如设备驱动程序、视频压缩程序等。VFW 软件开发工具包为在 Windows 系统中实现视频捕获提供的标准接口可以被大多数视频采集卡支持，并有多种视频压缩驱动供选择。

现在许多新的视频捕捉设备都采用的是 WDM 驱动方法（Windows Driver Model(WDM)的缩写，中文意思是“视窗驱动程序模块”），在 WDM 机制中，微软提供了一个独立于硬件设备的驱动，称为类驱动程序。驱动程序的供应商提供的驱动程序称为 Minidrivers。Minidrivers 提供了直接和硬件打交道的函数，在这些函数中调用了类驱动。

操作系统仍然支持 VFW 驱动程序，但是依赖于 VFW 的开发将逐渐减少。

表 2-1 WDM 驱动下常用接口

DirectShow 接口	相关属性
IAMTuner	PROPSETID_VIDCAP_TUNER
IAMTVAudio	PROPSETID_VIDCAP_TVAUDIO
IAMCrossbar	PROPSETID_VIDCAP_CROSSBAR
IAMVideoProcAmp	PROPSETID_VIDCAP_VIDEOPROCAMP
IAMAnalogVideoDecoder	PROPSETID_VIDCAP_VIDEODECODER
IAMAnalogVideoEncoder	PROPSETID_VIDCAP_VIDEOENCODER
IAMCameraControl	PROPSETID_VIDCAP_CAMERACONTROL

从编程实现的角度看，对 WDM 卡选择视频输入端子、设置采集输出的图像格式、设置图像的对比度、亮度、色调、饱和度等显示参数，都是通过 COM 接口 IAMCrossbar、IAMStreamConfig 和 IAMVideoProcAmp 来实现的。但对于 VFW 卡，

我们却不能编程实现上述设置，而必须将驱动程序定制的设置对话框直接显示给用户（让用户在这些对话框上做出选择）；VFW 驱动程序定制的设置对话框包括：Video Source（设置图像源属性）、Video Format（设置图像输出格式）和 Video Display（设置图像显示参数）。

对于应用程序开发人员来说，以上内容原理不可以不知，但也没必要研究得太透彻，而只需有一个总体的认识、作为背景知识了解一下就够了。

2.2 数字图像处理与人脸识别原理

数字图像处理概念广泛。就本课题而言，图像源已确定为从 USB 摄像头获取的图像，故为视觉图像。而在图像处理方面，在摄像头获取的彩色图像的基础上，进行初级的灰度化处理、中级的边缘检测、灰度直方图和高级的人脸检测和识别。

2.2.1 图像处理基础

（1）彩色空间与灰度化

图像获取是图像处理的第一步，通过 USB 摄像头获取的图像一般是与人类视觉图像相仿的彩色图像。彩色模型（也成为彩色空间或彩色系统）的目的是在某些标准下用通常可以接受的方式方便地对彩色加以说明。本质上，彩色模型是坐标系统和子空间的说明，其中，位于系统中的每种颜色都由单个点表示。

在数字图像处理中，实际中最通用的面向硬件的模型是 RGB（红、绿、蓝）模型，该模型用于彩色监视器和大多数彩色视频摄影机。本课题需要处理的图像即 RGB 彩色模型的数字图像。

在 RGB 模型中表示的图像由 3 个分量图像组成，每种原色一幅分量图像。当送入 RGB 显示器时，这三幅图像在屏幕上混合生成一幅合成的彩色图像。在 RGB 空间中，用于表示每个像素的比特数称为像素深度。考虑一幅 RGB 图像 ，其中每一幅图像都是 8 比特图像，在这种条件下，可以说每个 RGB 彩色像素有 24 比特的深度。

而很多图像处理操作是以灰度图像为基础的，因为将各种格式的图像转变成灰度图多通道的彩色图像变换为单通道的灰度图像后，这样一来计算量就会少得多了[10]。

一幅图像可以定义为一个二维函数 $f(x,y)$，其中 x 和 y 是空间（平面）坐标，而在任何一对空间坐标（x,y）处的幅值 f 称为图像在该点处的强度或灰度。

一般有分量法、最大值法平均值法加权平均法四种方法对彩色图像进行灰度化。

图 2-2 灰度化图片

而灰度变换是直接以图像中的像素操作为基础的空间域处理。空间域就是简单的包含图像像素的平面，某些图像处理任务在空间域执行更容易或者更有意义，而且通常空间域技术在计算上更有效，且在实行上需要较少的处理资源。

灰度变换是所有图像处理技术中最简单的技术。

（2）直方图

一幅图像由不同灰度值的像素组成，图像中灰度的分布情况是该图像的一个重要特征。图像的灰度直方图就描述了图像中灰度分布情况，能够很直观的展示出图像中各个灰度级所占的多少。图像的灰度直方图是灰度级的函数，描述的是图像中具有该灰度级的像素的个数：其中，横坐标是灰度级，纵坐标是该灰度级出现的频率。

具体的说：灰度级范围为[0,L-1]的数字图像的直方图是离散函数 $h(r_k)=n_k$，其中 r_k 是第 k 级灰度值，n_k 是图像中灰度为 r_k 的像素个数。在实践中，经常用成绩 MN 表示的图像像素的总数除它的每个分量来归一化直方图，通常 M 和 N 是图像的行和列的维数。因此，归一化后的直方图由 $h(r_k)=n_k/MN$ 给出，其中 k=0,1,…,L-1。简单地说 $p(r_k)$是灰度级 r_k 在图像中出现的概率的一个估计。

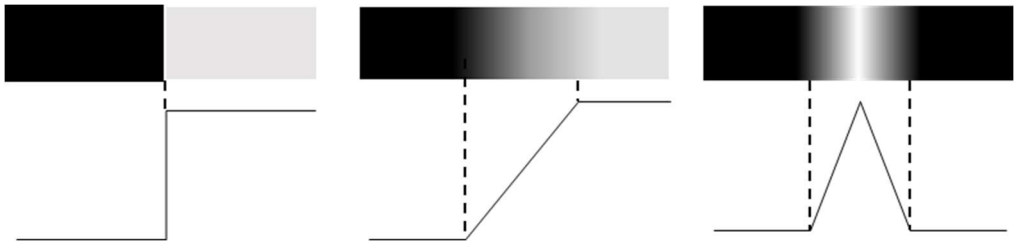
直方图是多种空间域处理技术的基础。直方图在软件中计算简单，而且有助于商用硬件实现，因此已经成为实时图像处理的一种流行工具。

（3）边缘检测

边缘检测是图像处理和计算机视觉中的基本问题，其目的是标识数字图像中亮度变化明显的点。图像属性中的显著变化通常反映了属性的重要事件和变化。这些包括深度上的不连续、表面方向不连续、物质属性变化和场景照明变化。边缘检测是图像处理和计算机视觉中，尤其是特征提取中的一个研究领域。

边缘检测是基于灰度突变来分割图像的最常用方法。我们从介绍一些边缘建模的方法开始，讨论一下边缘检测的原理和几种方法。

边缘模型根据它们的灰度剖面来分类，一般可以分为台阶、斜坡和屋顶三种模型。下图展示了这三种模型的灰度剖面和理想表示：



a.台阶模型　　b.斜坡模型　　c.屋顶模型

图 2-3 三种灰度剖面的理想模型

实际中。虽然噪声和模糊会导致与理想模型有偏差，但类似与这三种模型的边缘图像并不罕见。因此我们进行边缘检测时，能够根据这几种模型写出边缘数学表达式，这些算法的性能将取决于实际边缘和算法开发过程中所用模型之间的差别。

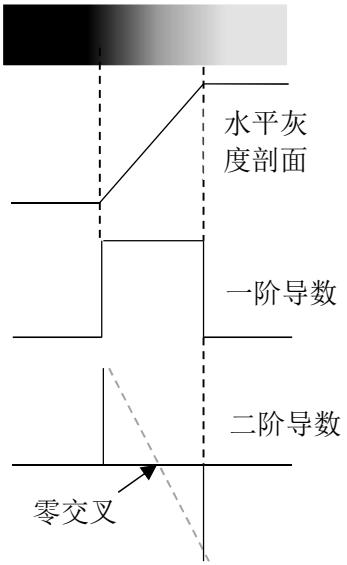


图 2-4 理想斜坡模型及其各阶导数

观察理想斜坡模型及其各阶导数，如右图所示。很容易看出，在斜坡开始和斜坡上各个点处，一阶导数为正。而在灰度区域一阶导数为零。在斜坡开始时，二阶导数为正；在斜坡结束处，二阶导数为负；在斜坡上和恒定灰度区域的各点二阶导数均为零。于是我们可以得出结论，一阶导数的幅值可用于检测图像中的某个点处是否存在一个边缘，二阶导数的符号可用于确定边缘像素位于该边缘亮的一侧还是暗的一边，同时二阶导数零交叉点可用于定位粗边缘的中心。类似的结论可以适用于图像中任何区域的边缘。

根据以上结论，可以得出边缘检测的三个基本步骤：平滑降噪、边缘点检测和

边缘定位。

最通用的方法是使用一阶和二阶导数来检测。检测灰度值的不连续性。从梯度来看，根据其微分特性，在恒定灰度区域的值为零，且其梯度值与灰度变化值变化程度相关。要得到一幅图像的梯度，则要求在图像的每个像素位置计算偏导数，当对对角线方向的边缘感兴趣时，我们需要一个二位模板。常用的模板有 Sobel 模板、Prewitt 模板等。使用模板求得各方向响应，再结合合适的阈值，便能够找到图像的边缘。

一些更为先进的边缘检测技术尝试考虑诸如图像噪声和边缘本身特性的因素改进简单的边缘检测，以获得更好的效果。成功的尝试有 Marr-Hildreth 边缘检测器、Canny 边缘检测等。大量的计算机视觉库都有相关算法的集成，在使用时直接调用即可。

2.2.2 基于机器学习的人脸检测与识别

人脸识别技术，作为一种典型的模式识别问题，简而言之，这是首先利用计算机分析人脸图像，进而从中提取有效的识别信息，最后用来辨认身份的一门技术。模式识别主要分为两大领域：决策理论方法和结构方法。第一类方法处理的是定量描绘子来描述的各种模式。第二类方法处理的是由定性描绘子来描述的各种模式。下文所述为本课题所要实现的基于机器学习的人脸检测与识别原理。

机器学习本质是将数据转换为信息——通过从数据中提取规则或模式来把数据转换为信息。在学习了一系列数据之后，我们需要机器能够回答与这些数据有关的问题。就人脸识别而言，假如我们有一个数据库，它是对 10000 张人脸图像进行边缘检测，然后收集特征，如边缘方向、边缘长度、边缘相对脸中心的偏移度。我么从每张脸中获得一个含有 500 个数据和 500 项特征向量。然后我们通过机器学习技术根据这些特征数据创建某种模型。如果我们仅仅想把这些数据分成不同的类（如检测判断是否为人脸），一个“聚类”算法就足够了。如果想学习从人脸的边缘的模式测算他的年龄，甚至识别出他是谁的话，我们则需要一个“分类”的算法。为了达到这个目的，机器学习算法分析我们收集的数据，分配权重、阈值和其他参数来提高性能。这个调整参数来达到目的的过程被称为“学习”（learning）。

机器学习用到的大量的原始数据一般会分为三部分使用，即训练集、验证集和测试集。我们利用训练集来训练我们的分类器，学习人脸模型，训练结束后，我们用测试集来测试检测识别人脸。验证集常常在开发训练分类系统时使用。有的时候测试整个系统是一个太大的工作。我们需要在提交分类器进行最终测试之前调整参数。即当训练完之后，我们尝试性的提前用验证集来测试分类器效果。只有对分

类器性能感到满意，才用测试集做最后的判断。

基本上，所有机器学习算法都使用特征构成的数据向量作为输入。在本课题中，人脸识别的最终目标是能够从摄像头获取的图像中检测人脸，并最终识别出作者本人。因此所有的训练集、验证集和测试集数据都是通过摄像头获取，并通过必要的预处理工作，提取出高质量的人脸数据用于训练和测试。

数据准备好后，下面就是选择分类器了。分类器不存在“最好”，但如果给定了特定的数据分布的话，通常存在一个最好的分类器。一般分类器的选择需要考虑计算速度、数据形式和内存大小。对于人脸检测和识别这类不需要很快训练而需要准确度高、判断快的任务，神经网络可以满足要求，但性能更好的选择是 boosting 和随机森林算法。

Boosting 是多个分类器的组合。最终的分类决策是由各个子分类器的加权组合来决定的。在训练时，逐个训练子分类器，且每个子分类器是一个弱分类器（只是优于随机选择的性能）。这些弱分类器由单变量决策树组成，被称为树桩，而且还根据识别精度学习“投票”的权重。当逐个训练分类器的时候，数据样本的权重被重新分配，使之能够给与分错的数据更多的注意力。训练过程不停地执行，直到总错误低于某个已经设置好的阈值。为了达到好的效果，这种方法需要很大数据量的训练数据。

综上所述：人脸识别由图像获取、人脸检测部位、图形预处理、特征提取和选择、训练和识别六大功能模块组成。

本课题用到的 Haar 人脸检测分类器便是巧妙地使用了 boosting 算法，建立了 boost 筛选式级联分类器。具体实现方法将在第三章相关章节介绍。通过调用训练的到的 xml 文件，人脸检测的难题便迎刃而解了。而人脸识别需要根据识别对象的大量数据训练对应的 xml 文件用于识别。

2.3 本章小结

本章总体介绍了本次毕业设计中绝大多数内容的基本原理，是编程实现各功能的理论基础。首先着眼于图像获取功能，比较了目前常用的图像获取技术的特定，从而的得出应选用 USB 摄像头完成本课题的图像获取部分。接下来，从 USB 设备与上位机（PC）的通信的角度入手，介绍了 Windows 操作系统下，如何驱动 USB 摄像头，以获取高质量的数字图像。最后介绍了一些图像处理操作，如灰度化、直方图边缘检测等，的基本原理。着重介绍了人脸识别功能。

第 3 章 基于 Python 的软件设计与编程

本课题使用 Python 作为编程语言。诞生于 1989 年的 Python 语言，作为第四代计算机编程语言 ，具有卓越的通用性高效性、平台移植性和安全性，同时又因其功能强大的解释性、交互性和面向对象编程的特性，既简单易学、又清晰高效，成为当下极为流行的编程语言之一。

Python 代码很容易阅读和编写，并且非常清晰没有什么隐秘的。Python 是一种表达能力非常强的语言, 这意味着, 在设计同样的应用程序时, 使用 Python 进行编码所需要的代码量远少于其他语言的代码量。目前 Python 官方网站版本已经更新至 Python 3.6.4 和 Python 2.7.14，强大的标准库奠定了 Python 发展的基石，丰富的第三方库保证了 Python 不断发展的，其开源特性和活跃的社区更是提供了强大的发展动力。

Python 自带的 IDLE，精简又方便，不过一个好的编辑器能让 python 编码变得更方便，更加优美些。常用的 Python 编译器有 PyCharm、Sublime 等。不过本课题选用 Visual Studio Code 作为编译器。这是微软公司发布的一款跨平台编辑器，轻便又易用。

当我们搭建好 Python 开发环境，安装好编译器以后，就可以着手开始软件编写了。接下了一整章的内容，我们将详细介绍如何基于 Python 语言实现本课题的所有功能。

3.1 图形用户界面设计

图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。在目前的软件设计过程中，用户界面的设计是相当重要的部分，美观、实用、大方的用户界面，能够极大的提高用户体验和软件亲和力。

本课题的所有功能，最终都将呈现在 GUI 上，所以在软件总体设计阶段，对 GUI 界面的设计是首当其冲的。本次毕业设计软件界面的主要功能结构设计如下图 3-1 所示。

作为一款成熟的软件，标题栏、菜单栏、中心窗口是必不可少的。图像获取和处理的结果将直接在窗口中展示，实时参数、实时图像、灰度图像、边缘检测图像和灰度直方图等。而各种功能的调用除了可以在菜单栏找到外，在中心窗口的合适位置也将放置部分按钮以方便用户操作。而相对复杂的设置等，会通过弹出对话框

的形式处理。同时，状态栏将会实时展示程序的各种运行状态。具体实现见 3.5 节。

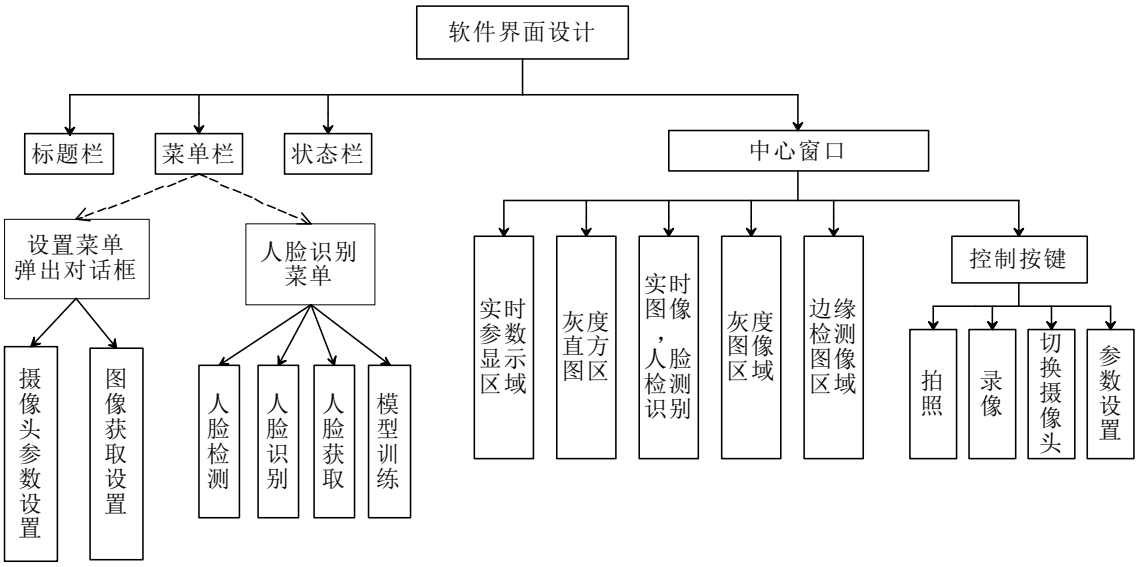


图 3-1 图形用户界面结构设计

3.2 程序结构设计

图形用户界面设计与程序结构设计是互为表里的。或者说，程序结构设计是软件设计最本质、最核心的内容。徒有界面而内部逻辑结构混乱的软件一无是处。

Windows 操作系统是一款图形化的操作系统，相比于早期的计算机使用的命令行，图形界面对于用户来讲更易于接受。GUI 程序是一种基于消息驱动模型的可执行程序，程序的执行依赖于和用户的交互，实时响应用户操作。GUI 程序执行后不会主动退出。GUI 程序运行模型如图 3-2 所示。

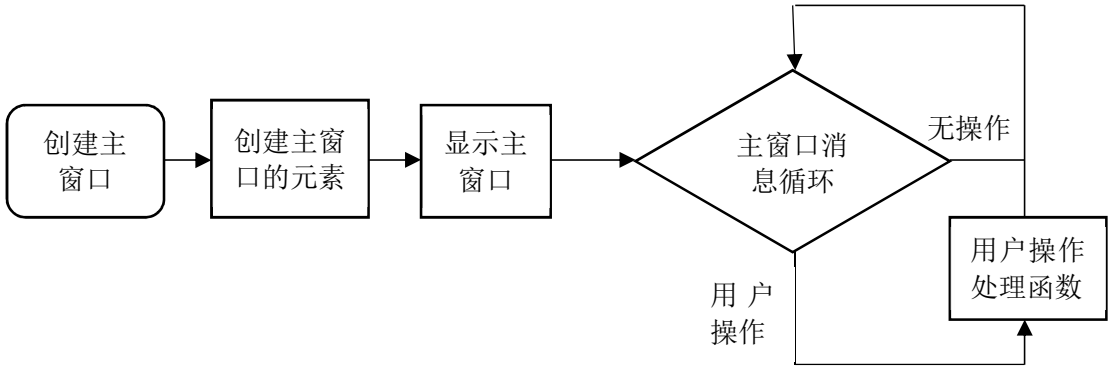


图 3-2 GUI 程序运行模型

程序框图是展现程序结构最直观、最清晰的方式，该软件总体的结构框图将在文末附录Ⅰ中给出。

本程序充分继承了 Python 语言面向对象编程的优势，各模块之间相互独立又彼此联系，同时 GUI 编程是采用了界面显示与业务逻辑分开的思想，既结构清晰又便于软件维护更行。为了使条理更清晰、也限于篇幅，图 3-2 仅着重展现了程序各模块之间的相互关系。各模块内部逻辑，将在下文分别给出。

3.3 利用 OpenCV 获取图像

3.3.1 从摄像头获取图像

如 2.1 节所述，在 Windows 操作系统下调取摄像头是一件牵涉到应用程序、驱动程序、硬件接口等的相当复杂的事情，但使用 OpenCV 让它变的简单起来。

OpenCV 在 HigeGUI 模块中做了很多工作，整合了 DirectShow 等等，使我们在获取摄像头图像时不需要考虑这些问题，只需要调用其封装好的接口，便能获取摄像机图像序列，像一般的图片或视频文件一样进行处理。

本次毕设中用到的 VideoCapture 类，便是 OpenCV 的 HigeGUI 中的获取视频图像的常用工具。在 Python 中的调用方式主要有以下两种：

cv2.VideoCapture(filename) → <VideoCapture object>

cv2.VideoCapture(device) → <VideoCapture object>

前者用于从文件中获取视频图像，”filename”即视频文件获取路径及名称。后者即我们需要用到的读取摄像头视频的形式，”device”是已加载的摄像头设备的 ID，如果仅仅连接了一个摄像头设备的话，只需输入“0”即可默认选中。如果我们想要允许用户自主选择的话，可以输入参数“-1”，这样在程序运行时会自动弹出一个对话框，列出所有可供选用的摄像头设备。

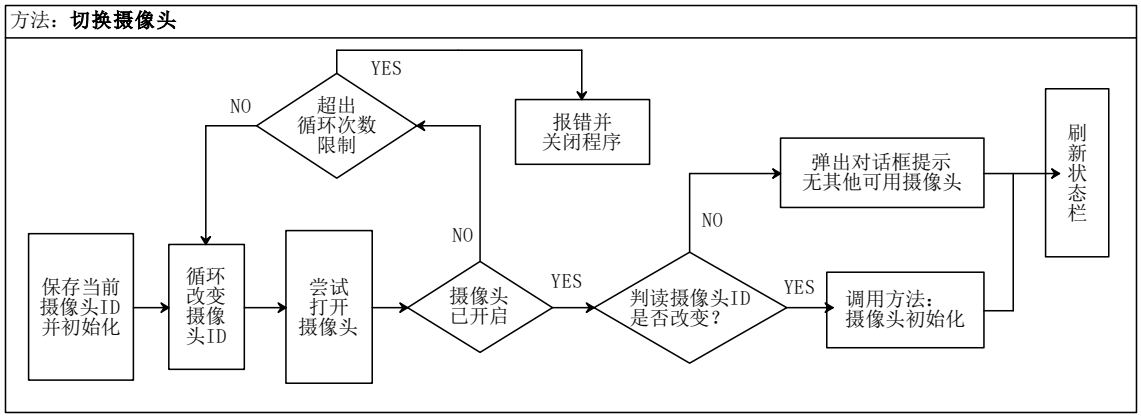


图 3-4 切换摄像头方法程序框图

本课题中，切换摄像头功能的实现是通过循环输入设备 ID 的列表来实现的，其程序程序框图如图 3-4 所示。这样的话，我们需要判断使用该 ID 的设备是否存

在。于是用到了 VideoCapture 类。

cv2.VideoCapture.isOpened() → flag

该方法的返回值是一个布尔值，即当摄像头成功开启的话，返回 True，否则返回 False。

之后，我们只需要再次调用 VideoCapture 中的另一个方法：

cv2.VideoCapture.read([image]) → successFlag, image

便能够从所选中的摄像头中获取当前帧了。

当然，从摄像头获取图像时有诸多属性参数是有必要进行一些调节的，这个操作我们同样可以使用 OpenCV 来完成。

在此之前，有一点还是需要明确的：OpenCV 虽然十分强悍，但它本质上仍是计算机视觉库，而不是视频流编码器或者解码器。之所以将视频获取这一部分做的强大一些，只是为了更方便后续的处理。而我选用 OpenCV 一方面是因为其方便易用的特点，更重要的是因为它能够满足本次毕业设计对摄像头操作的需求。如果想要突破限制实现功能更强大的多媒体功能的话，OpenCV 并不是一个好的选择。

3.3.2 摄像头相关参数获取与设置

我们在使用 USB 数字摄像头获取图像的时候，在不同的外部环境条件下，有一些属性参数——如亮度、对比度等——是需要随时做出调整的。当然，目前多数摄像头都是可以自动调节的，但是针对不同的图像获取需求，对属性参数进行手动设置还是不可或缺的。

不同型号摄像头可调节的参数是各不相同的，有些高端摄像头可调参数甚至多达上百个。就一般需求而言，最常用到的属性参数，无非就是亮度、对比度、饱和度、增益、锐度、白平衡、色调、灰度系数等等。对于这些参数，我们同样可以通过 OpenCV 的 VideoCapture 类来获取和设置：

cv2.VideoCapture.get(propId) → retval

cv2.VideoCapture.set(propId, value) → retval

前者用于参数值获取，后者用于设置。其中，“propID”即属性标识符（Property identifier），部分可选项及其含义列于表 3-1。

不过对于属性参数可调的范围，OpenCV 并未给出确定的方法。不过我们可以通过循环设置参数值的方法，找到上下限。

摄像头参数设置的功能通过程序弹出对话框的方式实现，因为涉及到 GUI 编程的知识，更详细介绍和程序结构示意图将在 3.5.3 节中呈现。

表 3-1 部分属性标识符及其含义

属性标识符	涵义
cv.CAP_PROP_FRAME_WIDTH	Width of the frames in the video stream.
cv.CAP_PROP_FRAME_HEIGHT	Height of the frames in the video stream.
cv.CAP_PROP_FPS	Frame rate.
cv.CAP_PROP_BRIGHTNESS	Brightness of the image (only for those cameras that support).
cv.CAP_PROP_CONTRAST	Contrast of the image (only for cameras).
cv.CAP_PROP_SATURATION	Saturation of the image (only for cameras).
cv.CAP_PROP_HUE	Hue of the image (only for cameras).
cv.CAP_PROP_GAIN	Gain of the image (only for those cameras that support).
cv.CAP_PROP_EXPOSURE	Exposure (only for those cameras that support).
cv.CAP_PROP_CONVERT_RGB	Boolean flags indicating whether images should be converted to RGB.

3.3.3 保存图像和视频

将从摄像头获取到的图像保存到本地相对来说就容易多了，再次仅给出所使用的 OpenCV 方法：

retval, buf = cv.imencode(ext, img[, params])

具体的用法读者可以从 OpenCV 的官方文档中找到详尽的叙述。

保存视频则用到了一个新的类：

cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]])→<VideoWriter object>

使用所要保存的视频的文件名、格式、帧率、画幅、色彩的创建一个 VideoWrite 类后，再按照帧率读取摄像头图像存入即可。如 3.3.1 节末尾所述，在 Windows 系统下，其本质是集成了 FFMPEG 和 VFW 来实现时候视频操作，局限颇多，比如视频格式有限且不能收取音频等。但对于图像获取的需求来说已经足够了。

关于定时将读取当前帧并存取，用到的是 GUI 编程多线程中 QTimer 模块，同样的，将留待 3.5.4 节中再做详述。

保存视频则用到了一个新的类：

cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]]) → <VideoWriter object>

使用所要保存的视频的文件名、格式、帧率、画幅、色彩的创建一个 VideoWrite 类后，再按照帧率读取摄像头图像存入即可。如 3.3.1 节末尾所述，在 Windows 系统下，其本质是集成了 FFMPEG 和 VFW 来实现时候视频操作，局限颇多，比如视频格式有限且不能收取音频等。但对于图像获取的需求来说已经足够了。

关于定时将读取当前帧并存取，用到的是 GUI 编程多线程中 QTimer 模块，同样的，将留待 3.5.4 节中再做详述。

3.4 使用 OpenCV 进行图像处理

在第二章中，我们已经详细的讨论过了需要进行的图像处理及其原理，本节内容则是在实际编程过程中，如何使用 OpenCV 来便捷高效的完成这些操作[14]。其中，灰度化、直方图、边缘检测等相对基础的操作不会占用太多篇幅，重点将会放在第二部分人脸检测与识别上。

3.4.1 基本图像处理

（1）色彩空间变换与灰度化

灰度化是彩色空间变换的一种。在 OpenCV 中，函数 cv2.cvtColor 可以将图像从一个颜色空间（通道的数值）转换到另一个，其用法如下：

dst=cv.cvtColor(src, code[, dst[, dstCn]])

其返回值为转换后的图像，具体转换操作由参数 code 来制定，下表列出了此参数的部分取值。

表 3-2 色彩空间变换函数 cv.cvtColor 参数 code 取值

Code	解释
cv.COLOR_BGR2BGRA cv.COLOR_RGB2BGRA	在 RGB 或 BGR 图像中加入 alpha 通道
cv.COLOR_RGB2BGR cv.COLOR_BGR2RGB	在 RGB 或 BGR 色彩空间之间转换
cv.COLOR_RGB2GRAY cv.COLOR_BGR2GRAY	转换 RGB 或者 BGR 色彩空间为灰度空间

灰度化过程中用到了 cv2.COLOR_BGR2RGB、cv2.COLOR_BGR2GRAY，即将摄像头获取到的 BGR 彩色空间的帧图像转换至 OpenCV 图像处理常用的 RGB 空间后，再转换至灰度空间，完成灰度化。

灰度化是图像处理最基础的工作，接下来的边缘检测、直方图，甚至人脸检测与识别，都将在灰度化得到的灰度图像的基础上进行。

（2）灰度直方图

直方图的计算是很简单的，无非是遍历图像的像素，统计每个灰度级的个数[11]。在 OpenCV 中封装了直方图的计算函数 calcHist，为了更为通用该函数的参数有些复杂，其声明如下：

hist=cv.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

该函数能够同时计算多个图像，多个通道，不同灰度范围的灰度直方图。使用该函数可以得到一个数组列表，包含所求灰度级范围内不同灰度级的像素个数，进而可以有多种方式使用这些数据绘制直方图，我们将会在 3.5.5 节中再做讨论。

（3）边缘检测

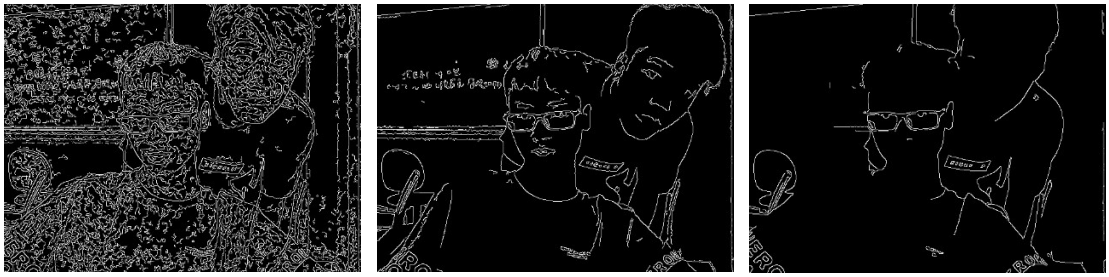
OpenCV 中常用于边缘检测的是 Canny 边缘检测法。Canny 算子首先使用高斯平滑滤波器卷积降噪；接着在 x 和 y 方向求一阶导数，按照 sobel 滤波器的步骤来操作，计算梯度幅值和方向，梯度方向一般取 0 度、45 度、90 度、135 度这 4 个可能的角度之中达到局部最大值的点组成边缘候选点；最后，Canny 算法最重要的一点就是运用滞后阈值来试图将独立边的候选像素拼装成轮廓，这需要两个阈值（高阈值和低阈值）：若某一像素位置的幅值超过高阈值，该像素被保留为边缘像素；若某一像素位置的幅值小于低阈值，该像素被排除；若某一像素位置的幅值在两个阈值之间，该像素仅仅在连接到一个高于高阈值的像素时被保留[12]。

因此，Canny()函数通过如下格式调用：

edges = cv.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])

其中，threshold1 为第一个滞后性阈值，用于边缘连接；threshold2 为第二个滞后性阈值，用于控制强边缘的初始段；高低阈值比在 2:1 到 3:1 之间。

当高低阈值分别取不同值时，从下图可以看出边缘检测的不同效果如图 3-5 所示。



a.阈值为 10，25　　b.阈值为 20，50　　c.阈值为 100，250

图 3-5 阈值取不同值时边缘检测效果对比

3.4.2 人脸检测与识别

人脸检测与识别是数字图像处理中比较高级的玩法，且目前十分流行。从第二章可以看到，其原理是相当复杂的。不过，如果借用强大的计算机视觉库 OpenCV 的话，实现起来会容易的多。

（1）人脸检测

OpenCV 的 CV 库提供的机器学习算法“Hear 分类器”可以用于人脸检测。这个物体检测方法巧妙的使用了 Boosting 算法。OpenCV 提供正面人脸检测器，它的检测效果非常好。

Haar 分类器是一个基于树的技术，它是首先由 Paul Viola 和 Michael Jones 设计的 boost 筛选式级联分类器，又称为 Viola-Jones 检测器。它使用 Haar 特征，更

准确的描述是类 Haar 的小波特征，该特征由矩形图像区域的加减组成[13]。

OpenCV 包含一系列预先训练好的物体识别文件，存储于 OpenCV 安装目录的如下路径：…\opencv\sources\data\haarcascades,其中本次毕业设计中使用的是正面人脸识别效果最好的模型文件”haarcascade_frontalface_alt2.xml”。

测试效果如图 3-6。

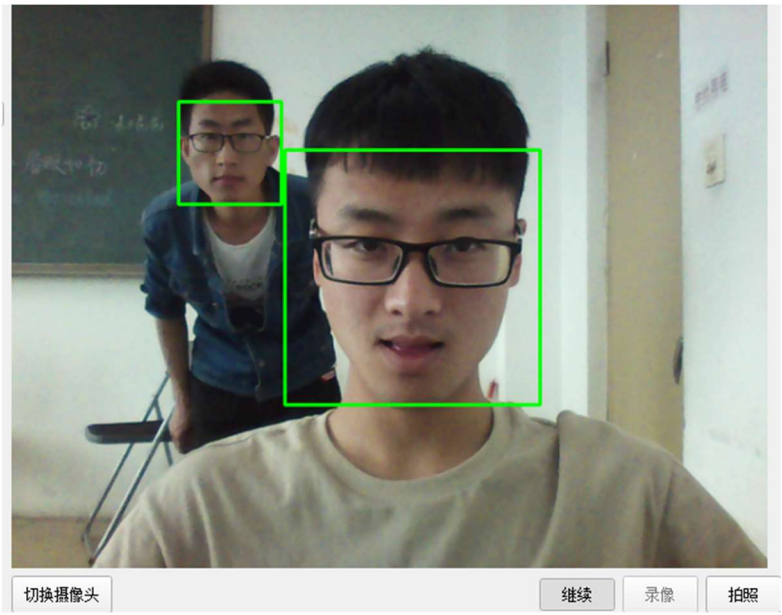


图 3-6 人脸检测

（2）人脸识别

以上我们已经知道怎么加载和运行一个预先训练并存储在 xml 文件中的沉降分类器以实现人脸检测的功能。现在我们转到怎么训练分类器来进行人脸识别。

OpenCV 中有一个专门用于人脸识别的 face 模块。其中 FaceRecognizer 这个类目前包含三种人脸识别方法：基于 PCA 变换的人脸识别(EigenFaceRecognizer)、基于 Fisher 变换的人脸识别(FisherFaceRecognizer)、基于局部二值模式的人脸识别(LBPHFaceRecognizer)。

我们选用 LBPHFaceRecognizer 来完成人脸识别，其过程大致可以分为一下的 3 个步骤[15]：

a. 收集打算学习的物体数据集——在这里，指的就是大量的鄙人正面人脸图片了。如果希望分类器充分发挥作用，则需要收集很多高质量的数据（1000—10000 个正样本）。高质量指已经把所有不需要的变量从数据中除掉。利用上文人脸检测程序可以很方便的收集数据：只需将从摄像头图像中检测到的人脸区域裁剪下来，并经过相同规格的预处理存储到指定路径中即可。使用此方法收集得到的部分数据如图 3-7 所示。

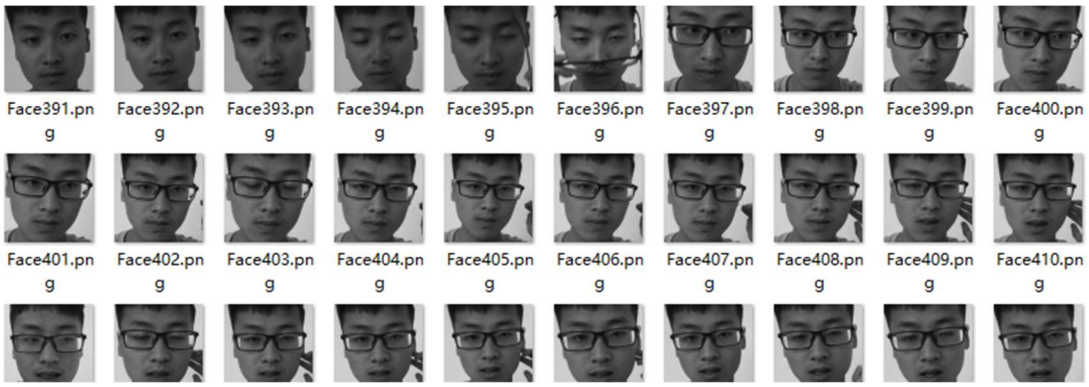


图 3-7 收集的部分人脸图像

b. 使用 cv2.face.LBPHFaceRecognizer 中的 train()函数来训练模型。需要将训练集中的图像创建为一个数据列表，并为每幅图像添加标签，要告知训练函数分别不同的人脸。根据数据量的大小和计算机性能的不同，训练时间往往会花费几分钟乃至于更长。因此为了避免每次识别时都进行训练，我们可以把训练得到的分类器用 save()函数保存成 XML 文件存储下来，下次用的时候直接用 laod()加载就行。

c.要使用训练的得到的模型来识别一幅人脸图像，只需要使用 prient()函数即可。该函数返回一个元祖格式值（标签，系数）。系数和算法有关，对于 LBPH 人脸识别算法，通常认为系数低于 50 为相对可靠，80-90 之间已经不可靠了，而高于 90 几乎可以确定识别正确率无穷小了。

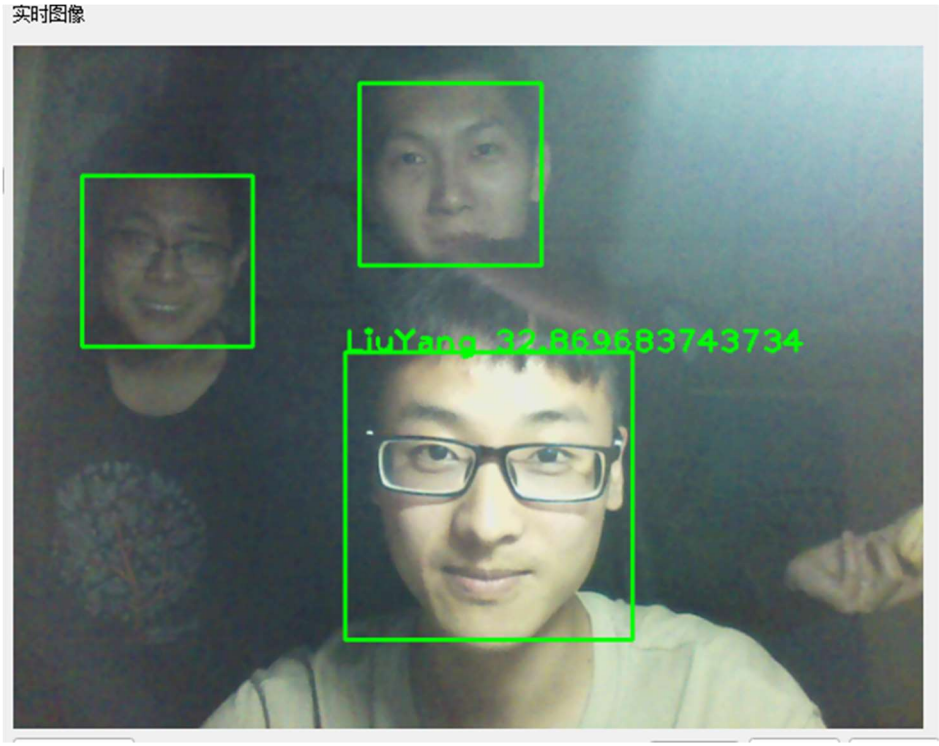


图 3-8 人脸识别

对于基于 PCA 变换和基于 Fisher 变换的人脸识别算法，其使用方法类似。以上本毕业设计就是利用 OpenCV 的 face 模块实现人脸识别的全部过程。

3.5 PyQt 5 和 GUI 编程

在软件设计过程中，图形用户界面（GUI:Graphical User Interface）的设计相当重要。美观、易用的用户界面能相当大的提升软件质量和使用量。Python 最初作为一门脚本开发语言，并不具备 GUI 功能。但由于其自身的良好的扩展性，目前已经有相当多的 GUI 控件集可以在 Python 中使用了，经常用到的有 PyQt、Tkinter、wxPython 等，其中 PyQt 是 Qt 框架专门为 Python 提供的 GUI 控件集，兼具 C++ 的高效和 Python 的优雅，同时因其开源的特性，受到越来越多软件开发者的青睐。因此，本次毕业设计选用 PyQt 目前的最新版本 PyQt 5 来进行图形用户界面的编写[16]。

下面将对本次毕业设计中用到的 PyQt 5 控件和在 GUI 上实现各功能的方法一一介绍。

3.5.1 QMainWindow 窗口控件

PyQt 5 中的窗口类有 QMainWindow、QWidget 和 QDialog 等，可以直接使用，也可以继承后再使用。

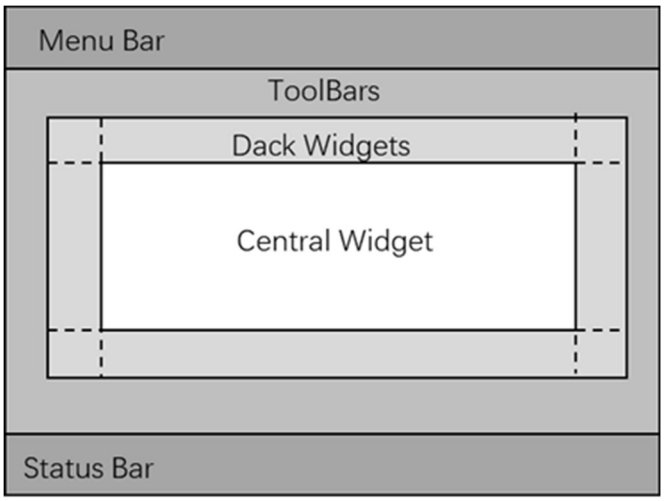


图 3-9 QMainWindow 类

QMainWindow 主窗口最常用的窗口类。如果一个窗口包含一个或多个窗口，那么这个窗口就是父窗口，被包含的窗口就是子窗口，没有父窗口的窗口就是顶层窗口，QmainWindow 就是一个顶层窗口，它包含很多界面元素，如菜单栏、工具栏、状态栏、中心窗口等，如图 3-9 所示。QmainWindow 不能设置布局，而是通

过 setCentralWidget()函数来设置中心窗口。

在本次毕业设计中，QMainWindow 是实现 3.1 中所设计的图形用户界面的基础，而具体控件的绘制，我们将使用 Qt Designer 来完成，它提供了一个可视化的 GUI 编程方式，可以通过拖曳的方式加入标签、按钮、文本框等各个控件并设置相关属性和布局。进而实现显示与业务逻辑的分离，相关内容将在 3.5.6 节中详述。

而 PyQt 中对象之间的通向使用的是信号（Signal）与槽（Slot）的方式。我们只需在逻辑文件中为菜单、按钮等关联对应的槽函数即可。

3.5.2 在 PyQt 界面上显示实时画面

在用户界面上显示实时画面是本次毕业实际 GUI 编程的核心功能。

PyQt 5 中有显示视频的模块 QMovie，但更多的是用与加载 GIF 动画效果的，对于从摄像头获取的实时图像显然不是一个好的选择。而显示图片则相对简单的多：使用最常用到的标签对象 QLabel 类中的 setPixmap()方法即可。从图片和视频的关系上，我们很自然的可以想到，如果按照一定的时间间隔不断刷新同一标签上显示的画面，岂不是就实现了实时画面的显示了么？而且这与前文所述的从摄像头获取图像的操作也是一致的，刷新图像的频率也即视频的帧率，故此二者可同步进行。

一般情况下，应用程序都是单线程运行的，但是对于 GUI 程序来说，单线程有时候满足需求。比如在实现实时图像显示时，主窗口循环相应菜单、按钮等操作占用了主线成，而我们想要按一定频率刷新画面，也是需要不断运行的，这样的话在执行过程中整个程序就会卡顿甚至被 Windows 系统关闭。要解决这种问题就涉及多线程的知识了。

PyQt 多线程技术涉及三种方法，其中一种是使用计时器模块 QTimer;一种是使用多线程模块 QThread；还有一种是使用事件处理的功能。对于在应用程序中周期性地执行某项操作，定时器 QTimer 是一个好的选择。

QTimer 的使用也非常简单：在引入 QTimer 类，实例化一个 QTimer 对象，将超时信号 timeout 连接到对应的槽函数，最后根据帧率设置时间间隔并启动定时器即可，即：

```
self.timer = QTimer(self)                          #实例化一个定时器
self.timer.timeout.connect(self.CameraPicture)     #超时信号连接至槽函数
self.timer.start(30)                               #按照帧率设置定时并启动
```

类似的，实时处理过的灰度图像和边缘检测图像也可以在同一个定时器的控制下显示在界面上。而视频录制的功能则需要实例化一个新的定时器来完成，方式类似。

更关键的问题在于该定时器所对应的槽函数 self.CameraPicture()。每次定时器调用槽函数后，需要完成获取摄像头帧、灰度化、直方图、边缘检测、人脸识别等一系列图像获取和处理的工作，这对各部分算法的效率有一个较高的要求。所幸，OpenCV 足够高效的完成了这个任务，相关算法已在对应章节讨论过了，下图给出了从摄像头获取当前帧到界面显示当前画面之间的程序结构框图如图 3-10。

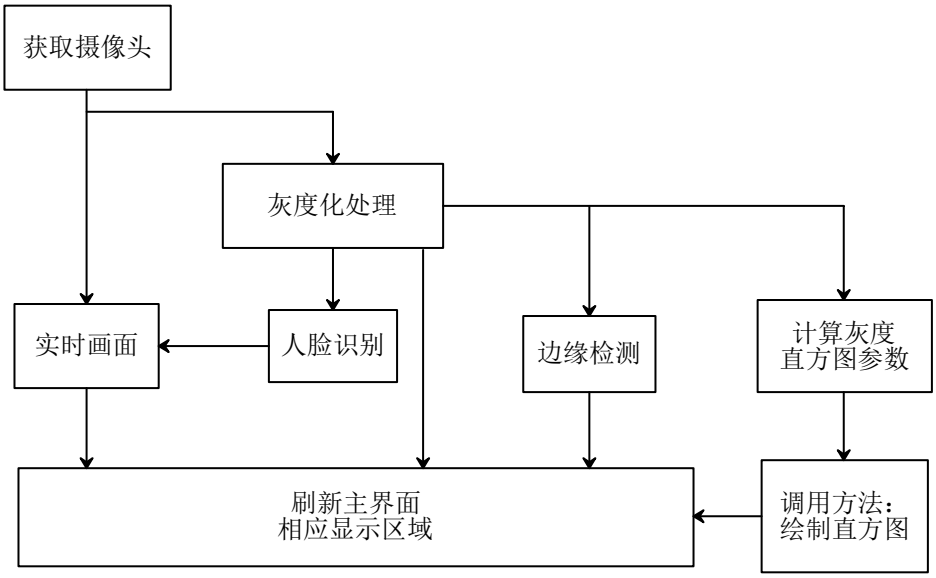


图 3-10 摄像头图像获取与处理函数结构框图

3.5.3 弹出对话框的使用

为了更好的实现人机交互，一些不适合堆在主界面上的功能往往会使用对话框来完成。本次毕业设计有多处用到了弹出对话框：

一是对摄像头参数进行设置；因为需要设置的参数较多，如果放在主界面上的话，一来显得拥挤凌乱，二来也不方便参数调整，故选用对话框实现。对话框与 QmainWindow 主窗口本质是一样的，因此也是使用 Qt Designer 完成界面文件的设计，然后通过继承、实例化，并在主窗口类中将该窗口中调节参数用的滑动条与对应槽函数连接即可。该对话框流程图如图 3-11。

只需要将对话框槽函数与对应的信号（如按钮、菜单等）相连，当该信号发出时，就会弹出对话框进入该窗口循环。

二是对图像获取设置对话框，该对话框实现了对拍照、录像设置，包括对图片和视频的保存位置的设置、使用复选框供用户自主选择是否保存实时图像、灰度图像或边缘检测图像以及对拍摄视频的帧率设置等。具体实现和调用的方法与摄像头参数设置对话框类似，不再赘述。

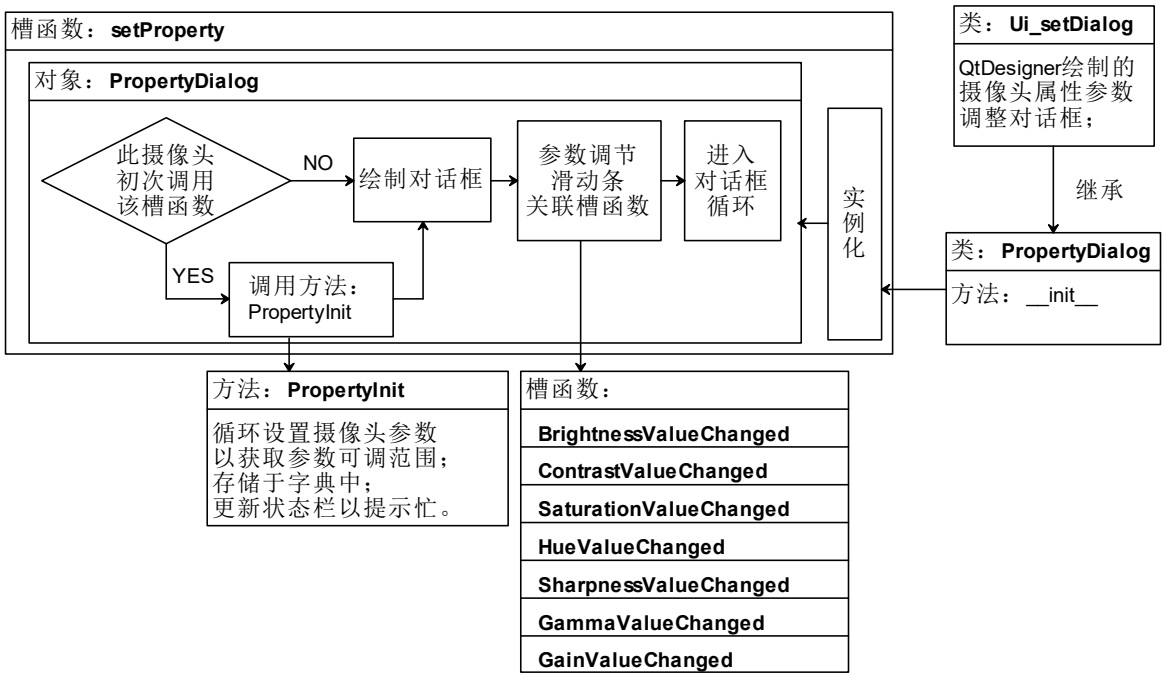


图 3-11 摄像头属性参数设置函数结构框图

另外，在 PyQt 5 中还定义了一系列的标准对话框来完成特定场景下的功能，主要有 QMessageBox、QFontDialogg、QFontDialog、QInputDialog 等。比如用于打开和保存文件的标准对话框 QFileDialog，在图像获取设置对话框中对图像视频的保存位置进行设置时就用到了它来获取文件目录：

self.filenameImage = QFileDialog.getExistingDirectory(self,None)

返回值为字符串格式的文件目录，使用非常方便。而 QMessageBox 是一种通用的弹出式对话框、用于显示消息，允许用户通过点击不同的标准按钮对消息进行反馈。在本毕业设计中，提示、关于、警告、报错等功能都使用其来进行交互。

3.5.4 利用按钮实现暂拍照、录像和暂停功能

基于前文的分析，拍照、录像和暂停功能的实现方法其实都已经交待清楚了。简单总结一下：“暂停”功能通过对定时器一启停，进而控制主界面实时画面的刷新来实现的；“拍照”功能实际就是将从摄像头获取的当前帧存储至本地的操作，用到了函数 cv2.imencode()，而“录像”操作主要是通过定时器二来完成的,详情见 3.3.3 节。

除此之外，为了在拍照瞬间给用户以拍照成功的反馈，我们除了在状态栏显示以外，还会控制定时器停顿一下来模拟拍照的感觉；而拍照时是否保存实时图像、灰度图像和边缘检测图像，录像时的帧率大小以及文件存储路径统统可以在图像获取设置对话框里进行设置。

由于这三种操作都涉及定时器的启停，因此相互之间会有影响，比如在暂停期间停止从摄像头获取图像，此时录像会出错。对此处理方法是当暂停时，禁用录像按钮。其它问题的具体编程逻辑依然通过程序框图给出，如图 3-12。

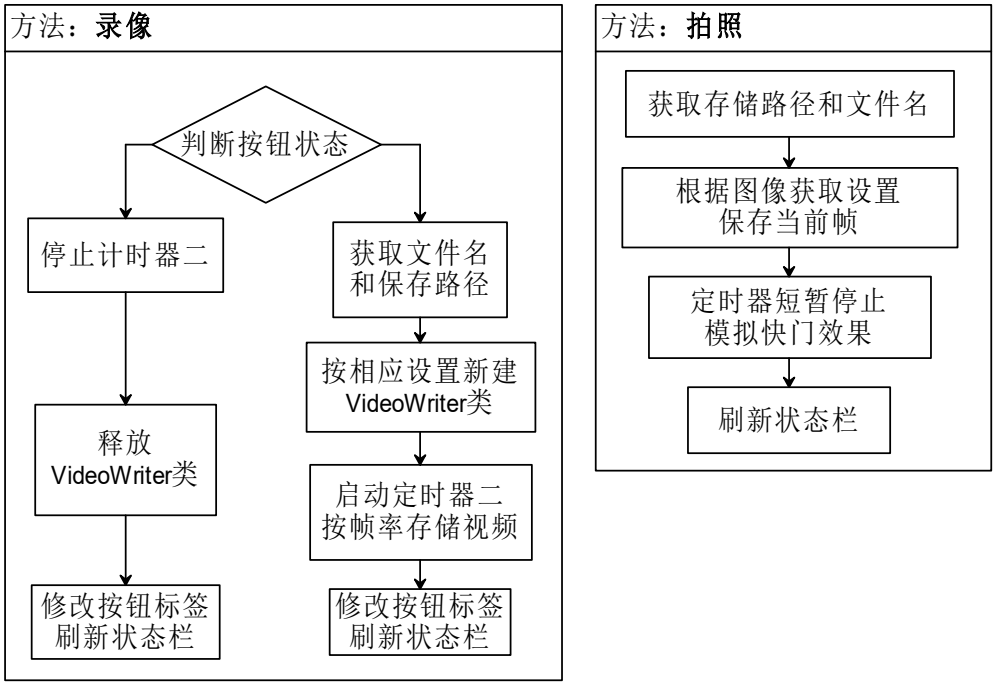


图 3-12 录像与拍照函数结构框图

3.5.5 在 PyQt 中应用 Matplotlib 绘制灰度直方图

在 3.4.1 节中我们介绍了如何使用 OpenCV 统计灰度图像的灰度直方图，得到的是一个包含 256 个成员的一维数组，每个数据对应该灰度级的像素个数。那么，该如何利用此数据在我们的用户界面上呈现出直方图来呢？在这里，我们用到了一个 Python 的绘图模块 Matplotlib。

Matplotlib 是 Python 最著名的绘图库，它提供了一整套和 MATLAB 相似的命令 API，十分适合交互进行制图。它提供了上百种类型的绘图源代码，这些图像基本可以满足我们日常的绘图需求。将 Matplotlib 嵌入到 PyQt 中后，我们便无需通过底层方式手动实现 PyQt 的绘图功能了。

直方图是非常基础的图像，只许在创建的画布中调用函数：

```
axes.plot(histr)
```

其中变量 histr 是我们如前文所述计算获得的直方图数据，便可以绘制出如右图所示的直方图图像了。接着 3.5.2 节中显示实时画面同样的方法不断重绘，即可得到与实时画面同步的灰度直方图了。

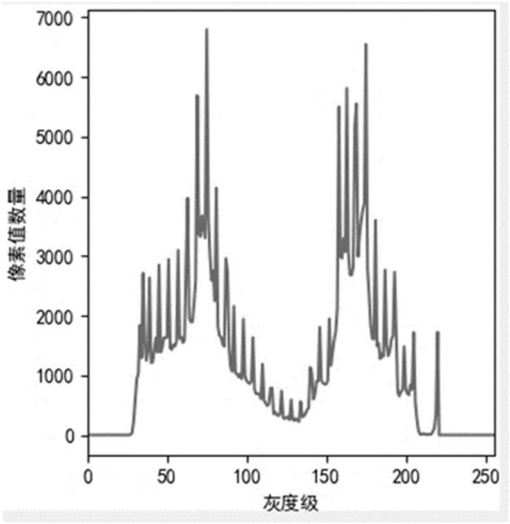


图 3-13 直方图

更值得关注的是如何将 Matplotlib 图像嵌入到 PyQt 5 界面中。我们用到的“设置提升的窗口控件”的方法。

具体来说，我们需要将绘图函数封装为一个绘图类，然后调用该接口，传入相应参数即可。过程如图 3-14 所示。

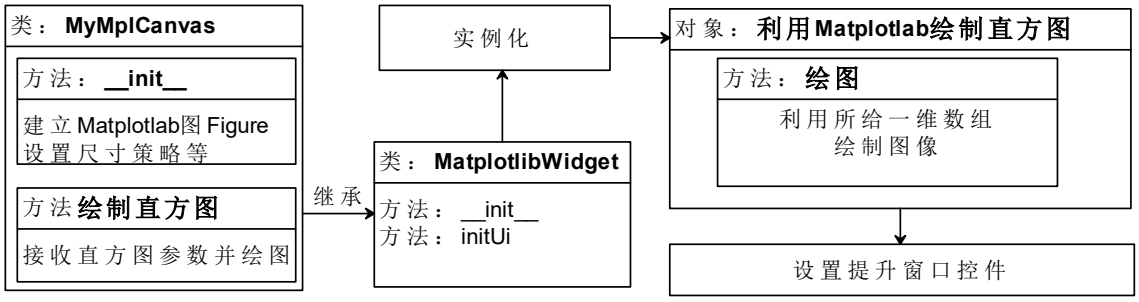


图 3-14 图像获取函数结构框图

“设置提升的窗口控件”可以使用 Qt Designer 完成，而 Qt Designer 的相关知识将在下一节介绍。

3.5.6 使用 Qt Designer 实现界面显示与业务逻辑的分离

在 PyQt 5 使用前文介绍的各中控件制作 UI 界面一般有两种方法：纯代码编写方式和使用 UI 制作工具。

Qt Designer，即 Qt 设计师，是一个专门用来制作 PyQt 程序中 UI 界面的强大的、灵活的可视化 GUI 设计工具，它生成一个扩展名为.ui 的 UI 界面文件，并可以通过命令将.ui 文件转换成.py 格式的文件，并被其它 Python 文件调用和继承。由于此.py 文件是由.ui 文件编译而来的，因此当.ui 文件变化时，.py 文件也会随之

变化。我们把这个由.ui 文件编译生成的.py 文件称为界面文件，而调用该界面文件的.py 文件称为逻辑文件或业务文件。这样一来，我们的程序代码便被清晰的区分为了界面文件和逻辑文件两部分，即所谓实现了“界面与逻辑的分离”（或称之为“显示和业务的分离”）。

界面与逻辑分离最大的好处就是当我们想要更新界面时，只需对.ui 文件进行更新并编译成对应的.py 文件；而逻辑文件只需视情况做出少量的调整即可。这无疑极大的便利了程序的编写与维护。不过，想要做出华丽的界面还是需要一些代码的。因此在本次毕业设计中，采用了 Qt Designer 为主，纯代码编写为辅的 GUI 制作方式。

Qt Designer 使用起来非常简单，只需通过简单的拖曳和点击，加入我们用到的控件并设置相应的布局等，即可完成复杂的界面设计。如图 3-15 所示。

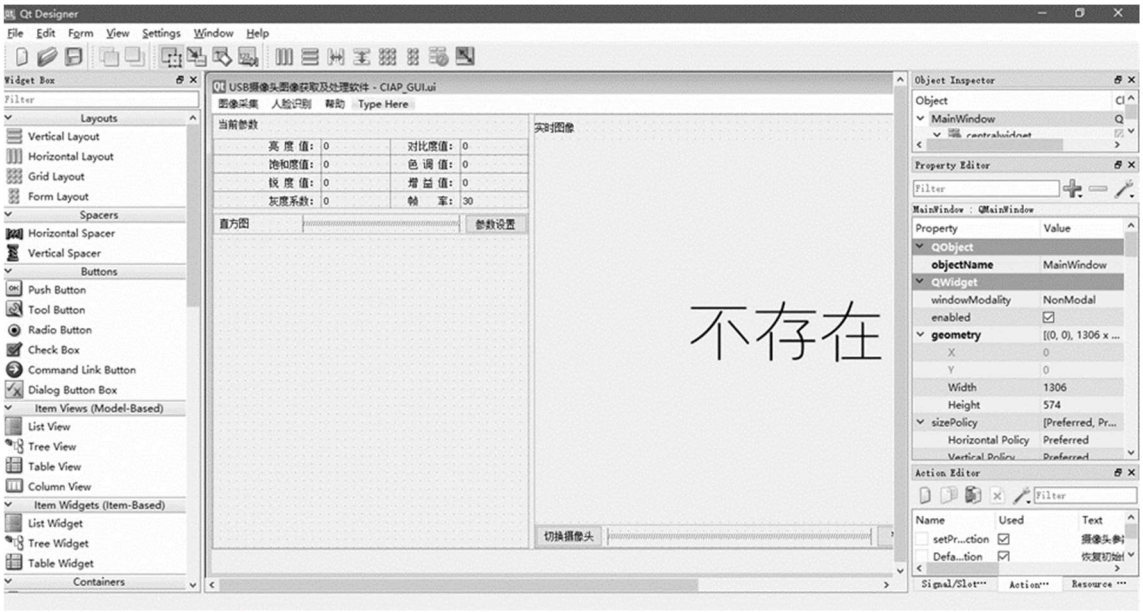


图 3-15 使用 Qt Designer 绘制 GUI

接下来，将.ui 文件编辑为.py 文件可以使用 PyQt 5 提供的命令行工具 pyuic5 轻松实现。例如在本次毕业设计中，需要将名为“CIAP_GUI.ui”的文件编译成名为“CIAP_GUI.py”的文件，则我们要在当前文件夹下进入命令提示符窗口，并输入命令：

```
pyuic5 -o CIAP_GUI.py CIAP_GUI.ui
```

便可以在同一文件夹下获得对应的.py 界面文件了。

3.6 软件打包与测试

我们开发的软件不只是给自己使用的，当我们将软件提供给用户的时候，使用

者可能并不清楚如何运行.py 文件，这时就有了把.py 文件编译成.exe 文件的以便在未配置 Python 环境的 PC 上运行的需求。在此，我们是使用 PyInstaller 打包项目生成 EXE 文件的，其原理其实就是把 Python 解释器和脚本打包成一个可执行文件。

PyInstaller 的使用非常简单。因为 PyInstallerjin 已经在 Scripts 目录下生成了可执行的 pyinstaller.exe 文件，所以在命令行窗口中进入需要打包的代码所在的目录下，然后运行下面的命令：

Pyinstaller [opts] yourprogram.py

可选的参数有：

-F，-onefile，打包生成一个 EXE 文件。

-D，-onedir，创建一个目录，包含 EXE 文件，但会依赖很多文件（默认选项）

-c，-console，-nowindowed，使用控制台，无窗口（默认）

-w，-windowed，-noconsole，使用窗口，无控制台。

输入指令后，首先 PyInstaller 会分析该脚本所依赖的其它依赖，然后进行查找，并复制，把所有相关的依赖都收集起来并进行加密处理，包括 Python 解释器，最后把这些文件放在一个目录下，或者打包一个可执行文件中。然后就可以直接运行所生成的可执行文件了。

于是根据课题需求，对于本次毕业设计名为 CIAP.py 的文件，我们可以使用命令：

pyinstaller -D -w CIAP.py

来打包生成 EXE 文件。

需要注意的是，程序需要的一些资源文件有时候需要我们手动添加至程序目录下。移植到其它设备上时，有时候难免会因为缺少某些文件不能运行。因此我们需要不断测试，以修改完善程序。

3.7 本章小结

本章从软件的整体设计到各部分功能能具体实现方案详细的记述了使用 Python 语言完成整个软件设计的主要内容。主要部分为计算机视觉库 OpenCV 和 GUI 库 PyQt 5 的使用。其中，OpenCV 是实现各功能模块的主要工具，从图像获取，到基本图像处理再到人脸检测识别都离不开它。PyQt 5 则是 GUI 编程的主要工具，串联和整合了各个模块，通过图形界面呈现给用户并供其使用。最终经过测试、打包，生成可执行的 EXE 文件。

结　论

本课题利用 Python 语言编写软件用于提取 USB 摄像头相关参数及图像，主要解决设备驱动、图形界面及仪器参量设定等问题。能够对图像提取像素矩阵，能对图像做相应算法，如灰度，显示直方图，边缘检测等。并能用够使用摄像头完成实时人脸识别。最终设计图形用户界面显示相应图像或曲线，供用户使用。

本文分别从原理和实践两个方面入手，详尽的介绍了从原始图像获取，到数字图像处理和人脸识别的各个模块实现方法。通过阅读本文，读者能够对 USB 摄像头的驱动与设置、图像处理的基本概念、基于机器学习的人脸识别算法有一个初步的认识。并且能够尝试使用 OpenCV 解决数字图像处理和计算机视觉的相关问题。同时，本文的程序编写继承了 Python 语言面向对象编程的特点，结构清晰、易于理解、便于维护。程序的 GUI 主要通过 PyQt 5 完成，采用了界面与逻辑分离的编程思想，Qt Designer 在大大提高了编程效率的同时，也增强了代码的质量。最终打包生成的可执行 EXE 文件可以在任何 Windows 系统的计算机上运行，拓展了程序的应用范围。

最后，本文虽然初步实现了人脸检测与识别的功能，但毋庸置疑的是，其仍有相当大的不足有待改进。最主要的就是识别正确率的问题，由于摄像头获取的人脸数据质量问题和样本数量的限制，训练的到的模型分类器并不足以胜任对识别准确率要求较高的任务，这需要今后在对相关算法加深理解的基础上对模型训练做出改进。同样的问题在采集实时数据、训练模型并识别的时候就更明显了。同时这一部分的程序逻辑并不完善，也有待今后重新设计来为用户带来更人性化的服务。

参考文献

[1] 李哲涛，李仁发，魏叶华，等. 无线传感器网络中时间同步与测距协同算法[J]. 计算机研究与发展. 2010，47(4)：638-644.

[2] Anonymous. USB3.0 connectors offer 10-fold speed hike over USB2.0[J]. Electronics Weekly,2010(2435).

[3] 刘中合,王瑞雪,王锋德,马长青,刘贤喜.数字图像处理技术现状与展望[J].计算机时代,2005(09):6-8.

[4] Rafael C. Gonzalez,Richard E. Woods.Digital Image Processing,Third Edition[M], Pearson Prentice Hall,2008.07.

[5] 梁文莉. 基于独立成分分析的人脸识别算法研究[D].西安科技大学,2012.

[6] 左腾.人脸识别技术综述[J].软件导刊,2017,16(02):182-185.

[7] Peter W. Hamilton. How to take and process digital images for publication[J]. Diagnostic Histopathology,2010,16(10).

[8] 肖旻,陈行.基于 Python 语言编程特点及应用之探讨[J].电脑知识与技术,2014,10(34):8177-8178.

[9] 吴学功. 基于 USB 摄像头的数字图像检测技术研究[D].东南大学,2006.

[10] 张桂华,冯艳波,陆卫东.图像处理的灰度化及特征区域的获取[J].齐齐哈尔大学学报,2007(04):49-52.

[11] 汪启伟. 图像直方图特征及其应用研究[D].中国科学技术大学,2014

[12] 高朝阳,张太发,曲亚男.图像边缘检测研究进展[J].科技导报,2010,28(20):112-117.

[13] Smietana Mateusz, Bock Wojtek J, Mikulic Predrag et al.. Detection of bacteria using bacteriophages as recognition elements immobilized on long-period fiber gratings.[J]. Optics Express, 2011, 19(9).

[14] Bradski,G.Learning OpenCV[M].BeiJing:Tsinghua University Press,2009.09.

[15] 刘丽,谢毓湘,魏迎梅,老松杨.局部二进制模式方法综述[J].中国图象图形学报,2014,19(12):1696-1720.

[16] 王硕,孙洋洋.PyQt 5 快速开发与实战[M].北京:电子工业出版社,2017.10.

[17] Radu Adrian Ciora,Carmen Mihaela Simion. Industrial Applications of Image Processing[J]. ACTA Universitatis Cibiniensis,2015,64(1).

致　谢

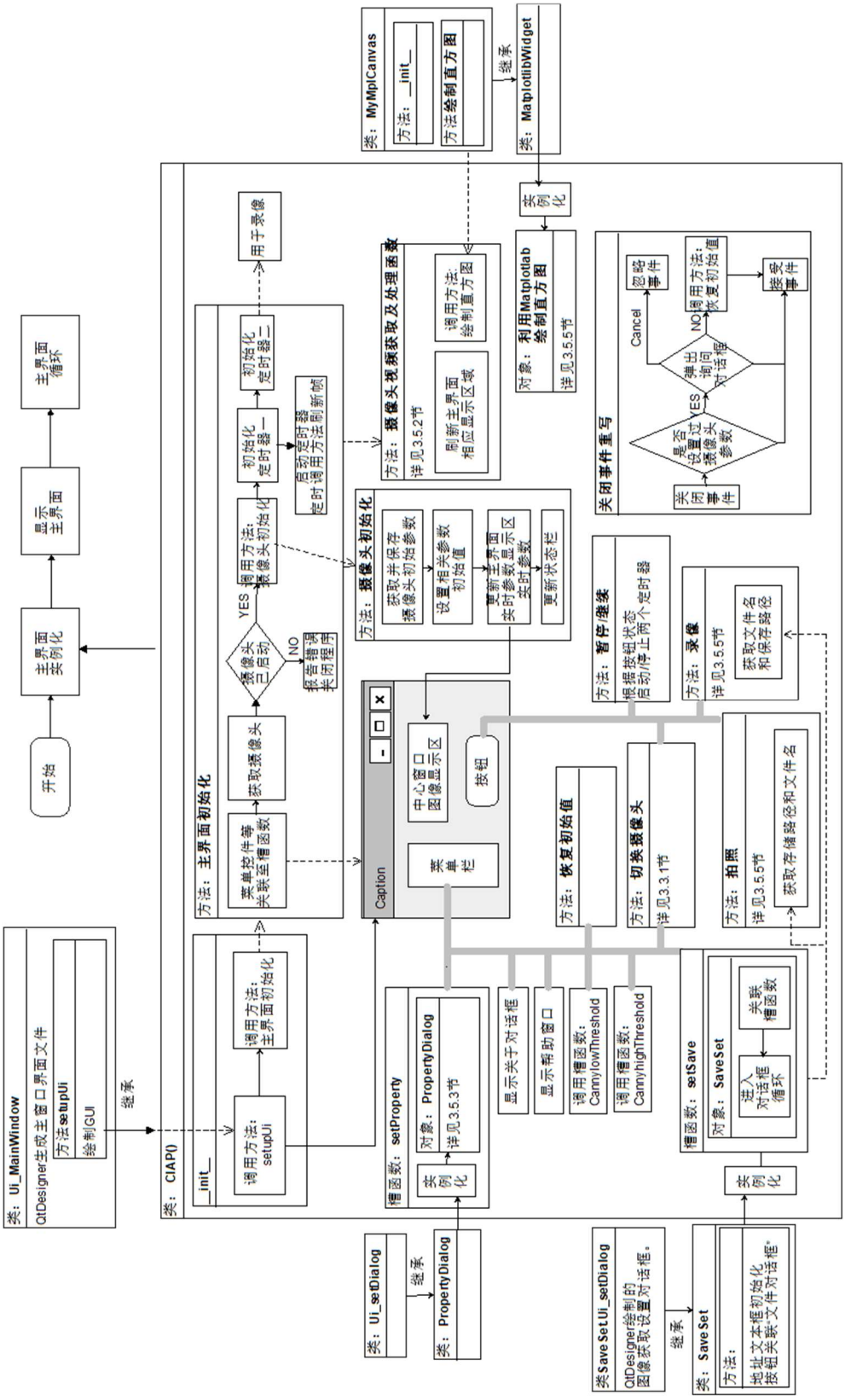
感谢哈尔滨工业大学“规格严格，功夫到家”的校训。

衷心感谢哈尔滨工业大学威海校区测控技术与仪器专业的诸位老师的悉心帮助和指导，尤其是我的导师王玲老师。

同时也要感谢半年来在 H421 实验室相互陪伴的同学们。

特此致谢。

附录Ⅰ 软件总体设计框图



附录 II 毕业设计软件效果