中南大學

CENTRAL SOUTH UNIVERSITY

《Java、Python 课程设计》

实 验 报 告

实验名称：　员工刷脸考勤系统

课程名称：　《Java、Python 课程设计》

指导老师：　[redacted]

姓名 [redacted]　**成绩**

班级 [redacted]　**学号** [redacted]

日期 2019.9　**地点** 综合楼四楼

备注：

目录

一、 需求分析 3

1． 产品描述 3
2． 产品需求分析 3
3． 产品预期目标 3

二、 总体设计 4

1. 基本思路 4
2. 遇到的问题与解决方案 4
3. 完整设计过程 5
4. 流程图 7

三、 程序详细设计 8

1. 编写及测试环境 8
2. 程序依赖库 8
3. 调用函数清单 8
4. 函数调用关系 11

四、 程序运行结果测试与分析 12

1. 第一版程序展示（无 GUI 界面） 12
2. 第二版程序展示 12

五、 结论与心得 12

六、 程序源码及参考文献 13

1. 程序源码 13
2. 参考文献及资料 13

一、 需求分析

1. 产品描述

员工刷脸考勤系统。需要使用 Python 作为程序开发语言，能够通过摄像头添加员工信息并通过摄像头识别员工。另外拓展要求是能够导出每日考勤表。

2. 产品需求分析

程序要求使用 Python 作为编程语言，作为目前行业最流行的编程语言，使用 Python 能够大大提高代码的可读性，提高编写效率，对于这种比较小的程序来说能够缩短开发周期。另外通过使用摄像头添加和识别员工，这就要求不能使用静态图片来添加信息而应该是实时监测，实时执行的系统。最后，扩展要求需要导出每日考勤表这就需要程序具有读写表格的功能以及类似数据库的存储能力。

3. 产品预期目标

第一代版本应该具备调用摄像头实时监测人脸的能力，并且在用户的确认下能够自动为员工注册或签到，在后台数据库更新员工的签到状态，并能够手动或自动的导出考勤表。

第二代版本可以添加 GUI 界面使程序更易用，美观，降低学习成本和使用的复杂性。

二、 总体设计

1. 基本思路

根据给出的要求，各个击破。首先要求使用摄像头收集信息，那么我们就使用OpenCV[1]模块，利用该模块优秀的兼容性，我们能够通过函数调用任何笔记本的基础摄像设备[2]从而提供视频流的输入。

之后，我们考虑在人脸识别方面使用 dlib[3]库作为我们的基础识别工具，用于在摄像头开启的状态下对每一帧实时的检测人脸，得益于它的轻量化与易用性，这一项功能对系统的计算能力要求不高[4]，可以应用于普通的嵌入式设备中。

关于接下来的员工注册环节，考虑到需要使用数据库或类似数据库的工具来保存员工信息，所以我们打算使用云端服务进行人脸的比对和数据库的建立。毕竟，如果数据库保存在本地有被篡改的风险，并且容量也是不容忽视的一点，经过多次比对，我们使用百度 AI 开放平台作为我们实现人脸比对+数据库功能的平台。云端加持，我们能够节省很多存储空间和运算能力，更加适合轻量化的设备。

2. 遇到的问题与解决方案

a) 最初遇到的问题就是学习百度 AI 开放平台。我们最初担心学习成本比较高。但是在耐心阅读了相关开发文档后，我们发现百度人脸识别的接口简单易用，非常方便。另外，这个发开文档对于我们之后遇到的一系列疑问起到了很好的指导作用。

b) 第一个问题就是我们在测试百度 AI 平台人脸识别功能的时候，一直无法识别人脸图像。平台要求传入的图像必须是以 BASE64[5]字符串或 URL 字符串或 FACE_TOKEN 字符串的方式上传。我们选择了最普通的 BASE64 方式上传，但是传回的信息是：‘*error_code:222203 image check fail*’，在网上查找解决方案才了解到，我们使用的本地编码器得到的字符串不是 utf-8[6]格式，云端无法扫描，于是我们添加了格式转换。问题解决。

c) 通过调用 dlib 库能够很好的实时监测人脸，起初我们打算在用户确认签到时将整个图像帧保存并上传，但是我们调用的云端功能只能识别一个人脸并与数据库比对，另外有一个多重比对的功能似乎并不能满足我们的需求，于是我们期望能够将确认的图像帧中所有的人头区域截取出来然后再依次上传。经过查阅，我们最后依据识别人脸提供的边框参数对图像进行裁剪然后记录下人脸数之后依次上传。并依次接收检测的参数。

d) 在后期想要加入 GUI 界面的时候我们使用的是 wxPython[7]库，在网络上查看相

---

[1] OpenCV 是一个基于 BSD 许可（开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows、Android 和 macOS 操作系统上。它轻量且高效，由 C 和 C++编写，同时提供了 Python、Ruby、MATLAB 等语言接口，实现了图像处理和计算机视觉方面的很多通用算法。

[2] 基础摄像设备指的是笔记本自带的摄像头，如果有附加的摄像头也可以通过修改函数参数来开启。

[3] Dlib 是一个用 C++编程语言编写的通用跨平台开源软件库。包含一系列机器学习算法和工具，它广泛应用于工业界和学术界，包括机器人、嵌入式设备，移动电话和大型高性能计算环境。

[4] 经测试，普通的英特尔低压处理器 1.6GHz 主频完全能够流畅运行。

[5] Base64 是网络上最常见的用于传输 8Bit 字节码的编码方式之一，Base64 就是一种基于 64 个可打印字符来表示二进制数据的方法。Base64 编码是从二进制到字符的过程，可用于在 HTTP 环境下传递较长的标识信息。

[6] UTF-8 是一种针对 Unicode 的可变长度字符编码，由 Ken Thompson 于 1992 年创立，现在已经标准化为 RFC 3629。

[7] wxPython 是 Python 语言的一套优秀的 GUI 图形库。允许 Python 程序员很方便的创建完整的、功能键全的 GUI 用户界面。wxPython 是作为优秀的跨平台 GUI 库 wxWidgets 的 Python 封装和 Python 模块的方式提

关的设计代码的时候发现有的使用了类结构，有的直接上函数然后在主函数里画程序框，我们一开始不想封装成类，因为参数的修改工程浩大，但是在试验了后者（不使用类）的方法之后，我们发现不使用类无法满足设计要求，最后只能作罢。

e) GUI 界面制作好之后，我们开启摄像头的界面与主程序界面不是同一个，是另外弹出的。在指导老师的要求下，我们需要把图像嵌入在程序框中。这怎么嵌啊。搜索可行的方案之后才了解到，可以利用 OpenCV 在接受图片之后直接在程序框内画图像。多了这么一个过程，帧率比之前有所下降。

f) 在最后的程序优化中，有一处异常。在开启摄像头之前点击关闭摄像头会提示变量未定义，即抛出异常 AttributeError[8]。这是因为类中的 camera 参数还没有被创建[9]。那么我在这里就加了一个异常处理来规避这个问题。

g) 我们通过使用 os 库来获取文件的绝对路径，但是得到的字符串路径实际上是该文件所在文件夹的路径，要想对这个文件进行操作，仍需要手动在该路径后面添加文件名。

h) 如果人脸在程序中只有一半，但是用户仍然点击了注册，那么程序会报错。因为裁剪程序获取了推测的人脸边框数据，但是在裁剪的时候超出了边界，抛出异常 IndexError[10]。为了处理这个异常，在检测到程序抛出 IndexError 异常后提示用户不要半张脸注册签到。

3. 完整设计过程

首先拿到题之后我们根据前问所述的基本思路开始寻找能够使用的模块并依次对每一个模块进行测试，确保模块能够单独运行。首先考虑摄像头的问题，配置好 OpenCV 之后，测试了调用摄像头的代码，能够完美运行。

之后，开始测试百度 AI 开放平台提供的人脸识别接口。根据开发文档的指导，配置好了 baiduAPI 基础库，研究具体接口的用法。我们先在云端创建了一个人脸用户组，并且上传了一张我本人的照片作为测试。然后通过程序向云端发送另一张我的照片，在经历了上述的错误以及排错过程后，终于传回了 *SUCCESS* 的信息。有趣的是，云端传回的处理结果是一个非常长的字典和列表多层嵌套的结构，我又专门复习了一遍字典的相关概念以及从这种结构中提取到想要的信息。最终我们提取的信息就是，识别状态，识别的人脸对应的名称。值得注意的是，员工在云端的人脸库注册的 ID 必须和本人姓名一致，否则在写入表格环节无法认证并修改签到状态。不过这是后话了。

接下来开始测试 dlib 库的人脸识别和人脸截取模块。网络上有一个可以参考的实时检测程序，他的代码很有借鉴意义，我简化了他的程序，仅仅是实时在画面中用方框标出人脸即可，这样既可以让用户很直观的看到程序识别出了人脸，而且简化了计算过程加快了。切割的代码是利用了 dlib 识别的时候返回的人脸的基本参数，然后利用 OpenCV 进行裁剪。测试的时候是输出到指定的目录。经测试，在单独运行之后能够在指定目录找到单独的人脸图像。

我们在这个时候打算先不做数据库及图表写入的模块，因为上述这些已经测试的模块，把它们整合起来的代码量已经不短了。于是，我们开始整合模块，写函数。

供给用户的。

[8] AttributeError:对象没有这个属性

[9] Camera 参数并非随着类的初始化而创建，而是在第一次开启摄像头时被创建，而关闭摄像头的按钮并不会销毁这个变量。

[10] IndexError: List index out of range 对列表进行索引取值的时候，索引超出列表范围

整个过程并不难，就是一些逻辑关系的判断和特定时候的调用。尤其需要注意的是部分语句的顺序，比如摄像头的开启和释放。

在完成之后就开始调试表格写入的功能了。在查阅了网上相关的 Python 操作 Excel 表格的库，最终使用了 xlrd[11]，xlwt[12]，配合来读写 Excel 表格。首先在初始化程序的时候就联络云端数据库获取人脸库的 ID 列表然后创建表格并写入 ID。之后在每次注册成功之后，也就是云端传回 SUCCESS 的消息之后就在表格对应的 ID 后写入 Registered 以及从系统获取的当前时间。

这样，整个程序的所有模块就都已经整合了。开始添加一些优化的代码，添加了网络检查工具，因为整个程序必须需要网络环境，否则会报错。然后为了兼容所有设备，写了一个创建目录的函数，用于缓存图片文件和存储历史考勤日志，这个目录创建函数被作为初始化函数在程序打开时即运行并创建缓存目录。另外，我们还完善了员工的注册功能，如果有图像传到云端并识别失败，那么会展示为识别出人的头像，然后询问用户是否注册，如果用户注册，那么在用户输入这个人的姓名之后，自动上传图像为其注册。如果用户不注册，那么就将该图片上传至 guest 人脸图像组，ID 命名为检测时间。

这些全部做完之后，因为没有做 GUI 界面[13]，所以还不太放心，于是开始在网上寻找 Python 相关的 GUI 模块。最终选定用 wxPython 模块制作图形界面。我们在网上找到了两种方法，一个是需要编写类的，这个类也是必须按照这个模块的格式来写，另一个就是不用编写类，直接在主函数里面画框。因为时间不多，我们选择了不使用类，因为如果用类的话，类内部的参数需要修改的地方太多了，直接编写更快一些，但是限制也颇多。向图形界面中融合了已经编写的内容之后，程序大体就完成了。

然后迎接老师的检查。最终老师给出的改进意见是把指令从图像上拿出来。原本在没有图形界面之前，我们的指令是嵌在摄像头的画面上的，为了融合图形界面，我们也就没有更改。另外一点，我们的图形界面点 open camera 的时候，摄像头程序框是单独再弹出来一个的，老师要求我们把这个摄像头框嵌入在程序里，不要再多弹出一个。根据老师的要求，我们具体做了如下的更改：

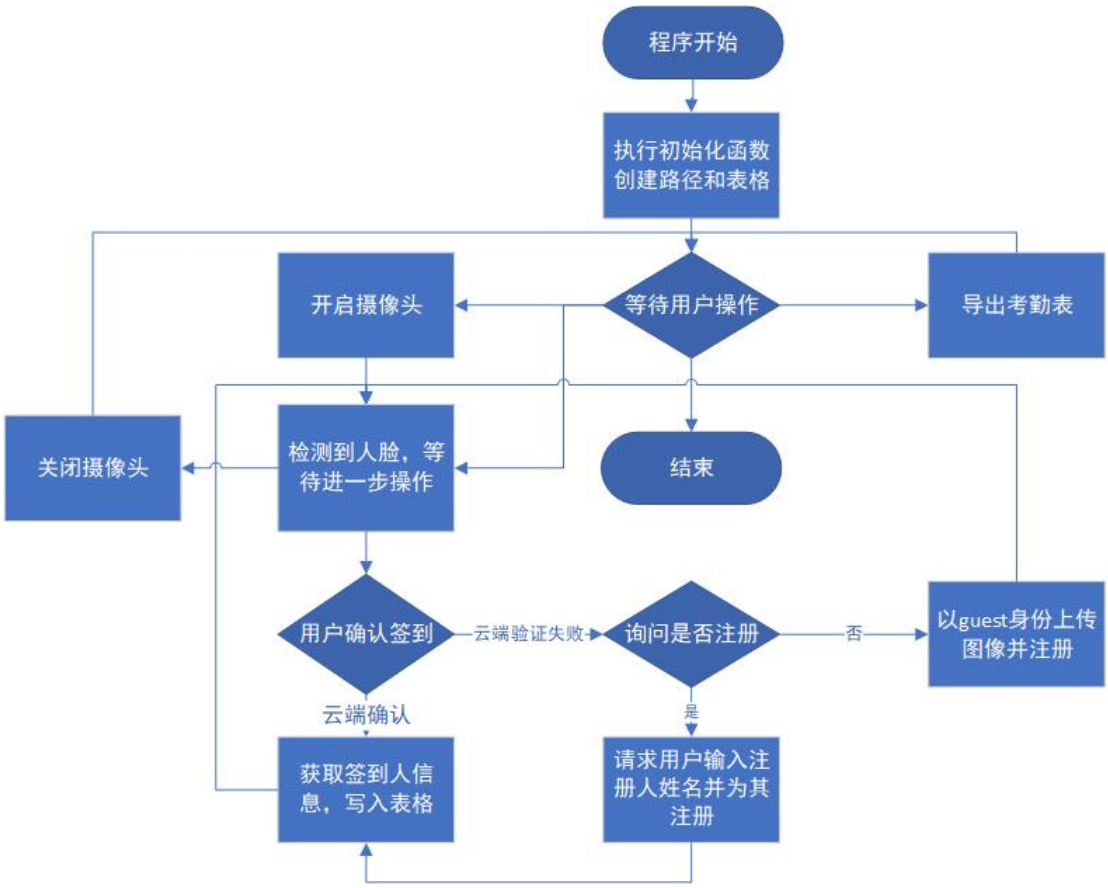
a) 移除图像上的所有命令，把这些命令都转化为程序框中的按键。这可就有点麻烦了，因为我们没有用类，写按钮很麻烦。所以我们干脆一不做二不休，就使用了类。重新修改了所有参数，新写了一个类，在类中添加按钮方便多了。于是乎，我们把摄像头的开启，关闭，签到，注册[14]以及使用说明都添加到了程序中。
b) 把视频流嵌入到程序中这个要求，经过网上的搜索，我们了解到，这个过程实际上是用 OpenCV 把获取的图像在程序框中一帧一帧的画一遍。经过一番研究之后终于在我们的程序上实现了这个功能。
c) 修补了程序中的一些逻辑漏洞。比如在没开摄像头的时候点关闭摄像头会提示，开启摄像头之后再点开启会提示。这里面的原理最开始是用简单的判断 isOpend[15]这样的函数返回值来确定是否开启或者关闭，但是后面就有一种情况发生，就是在没有打开摄像头的时候，这个类内部的 camera 变量还没定义，会有异常报出，提示变量未定义。于是我就在 isOpend 这个

[11] Xlrd 是 Python 语言中读取 Excel 的扩展工具。可以对指定表单，指定单元格进行操作。
[12] Xlwt 是 Python 语言中写入 Excel 的扩展工具。可以对指定表单，指定单元格进行操作。
[13] 没有做 GUI 界面是因为扩展要求并没有给出。
[14] 注册功能只有在识别失败的情况下才会询问并弹出注册窗口。
[15] OpenCV 内置函数，返回值为布尔变量，用于判断摄像头状态。

函数外面加上了异常处理，如果有未定义异常同样提示未开启，就完美解决了。

以上，便是我们程序设计的完整过程。

4. 流程图



三、 程序详细设计

1. 编写及测试环境

操作系统[16]：macOS Mojave 10.14.6[17]

IDE：PyCharm Professional 2019.01

Python 3.7

2. 程序依赖库

```
import wx  # 图形界面依赖库
import xlwt  # Excel 表格写
import xlrd  # Excel 表格读
import dlib  # 人脸识别核心库
import time  # 用于获取当前时间
import shutil  # 文件的相关操作
import numpy as np  # 人脸识别相关
import base64  # base64 编码器
import os  # 调用系统功能，路径，控制台等
import cv2[18]  # opencv，用于图像识别和摄像头等
import matplotlib.pyplot as plt  # 图像显示
from PIL[19] import Image  # 图像处理配合 matplotlib 使用
from aip import AipFace[20]  # AI core
from xlutils.copy import copy  # Excel 相关
from tkinter[21] import messagebox  # 用于弹出简易的消息框[22]
```

除了注释标注的部分外，其余 import 的依赖库均是其本名，可以直接从 PyCharm 的集合库[23]中搜索安装。

3. 调用函数清单

（类）外部调用函数：

a)
```
def network_test():
    """
    临时起意想写的测试网络的小函数。。。
    如果网络状态不好，则会在 message box 弹出消息要求网络连接。
    """
```

---

[16] 此处提示操作系统的目的是因为部分依赖库对于不同的操作系统是不同的。比如 OpenCV 和 xlrd 等对于 macOS 和 Windows 是不同的。

[17] 经测试，程序同样可以运行在更老版本的 macOS 系统下。比如 macOS Captain。

[18] 导入的时候名字是 opencv-python

[19] PIL 是 Python 的第三方图像处理库，但是由于其强大的功能与众多的使用人数，几乎已经被认为是 Python 官方图像处理库了。

[20] 即链接百度云端的依赖库。控制台执行：Pip install baidu-aip 以安装

[21] Tkinter 是 Python 的标准 Tk GUI 工具包的接口。

[22] 事实上，wxPython 是有能力弹出 message box 的，但是在类参数初始化之前，无法调用，所以这个额外的 import 也是无奈之举。

[23] PyCharm → Preference → Project: XX → Project Interpreter → + → 搜索添加

b)
```
def makedir():
    """
    初始化创建必需的缓存文件夹。包括截图文件夹，人脸分离文件夹、人脸检测缓存文件夹，考勤日志存储文件夹
    :return: 返回三种文件夹的路径str
    """
```

c)
```
def initialize(client):
    """
    初始化函数，主要是在执行主程序前准备好出勤表格
    :param client: 为了从云端服务器获取员工信息并在本地建立表格
    :return: 返回获取的员工列表
    """
```

d)
```
def show_img(path):
    """
    辅助用函数，用于显示图片。入口参数为图片路径。用户不可修改路径，软件内置。
    :param path: 图片路径
    """
```

e)
```
def clear_images(path_save, path_cache):
    """
    清除所在文件夹的所有图片，用于初始化
    :param path_save: 截图存储路径
    :param path_cache: 缓存图片存储路径
    """
```

f)
```
def face_separate(cc_path, sep_path, detector):
    """
    人脸分离函数。主要功能是将人脸从静态图片中剪切并导出。
    :param cc_path: 缓存图片路径，用于存储原始图片
    :param sep_path: 被剪切的人脸图像保存路径
    :param detector: dlib参数，此处用作入口参数传入减小代码重复率
    :return: 返回检测到的人脸数量
    """
```

g)
```
def face_recognize(client, read_path):
    """
    该模块用于人脸比对，利用百度ai人脸识别接口
    :param client: 百度ai客户端验证
    :param read_path: 人脸图片读取路径
    :return: 返回识别结果
    """
```

h)
```
def face_register(client, read_path, group_id, usr_name):
    """
    人脸注册模块，主要是向云端上传人脸图片以及自定义人名
    :param client: 减少复用，client作为参数传入
    :param read_path: 上传图片的路径
    :param group_id:云端图片组名称
```

```
        :param usr_name: 新注册用户名称
        :return: 返回注册结果
        """
```

i)
```
def live_cam_detect(sep_path, cc_path, usr_list, self):
        """
    程序核心函数，主要利用opencv 使用摄像头，然后调用上述函数进行完整的程序过程
        :param self:在类中使用
        :param usr_list: 从参数入口传入初始化获取的员工列表
        :param sep_path: separate 路径
        :param cc_path: 缓存路径
        """
```

类内部[24]调用函数：

a)
```
def _learning_face(self,event):
            """
    GUI 程序核心调用函数，用于开启摄像头并能够执行后续相关程序
            :param event: 类要求，实际并未显式调用
            """
```

b)
```
def confirm_face(self, event):
            """
    这个确认签到函数是在第二版程序最后加上的，因为要满足指令移除出摄像头画面的要
    求，我们单独写了一个签到函数
    实际上就是把第一个版本的注册函数单独从 live_cam_detect 中分离了出来做成一个按
    钮
            :param event: 类要求，实际并未显式调用
            :return: 事实上这个函数是没有返回值的，只有message box 对于当前状态
    的提示
            只有在出错的情况话才会返回1，表示出了问题
            """
```

c)
```
def learning_face(self, event):
            """
    使用多线程，子线程运行后台的程序，主线程更新前台的UI，这样不会互相影响
            :param event: 类要求，实际并未显式调用
            """
```

d)
```
def close_face(self,event):
            """
    关闭摄像头，显示封面页
            :param event: 类要求，实际并未显式调用
            :return: 如果有异常抛出，则返回1
            """
```

e)
```
def view_log(self,event):
            """
    查看当前日志记录，语句比较简单，利用控制台直接打开表格文件。
```

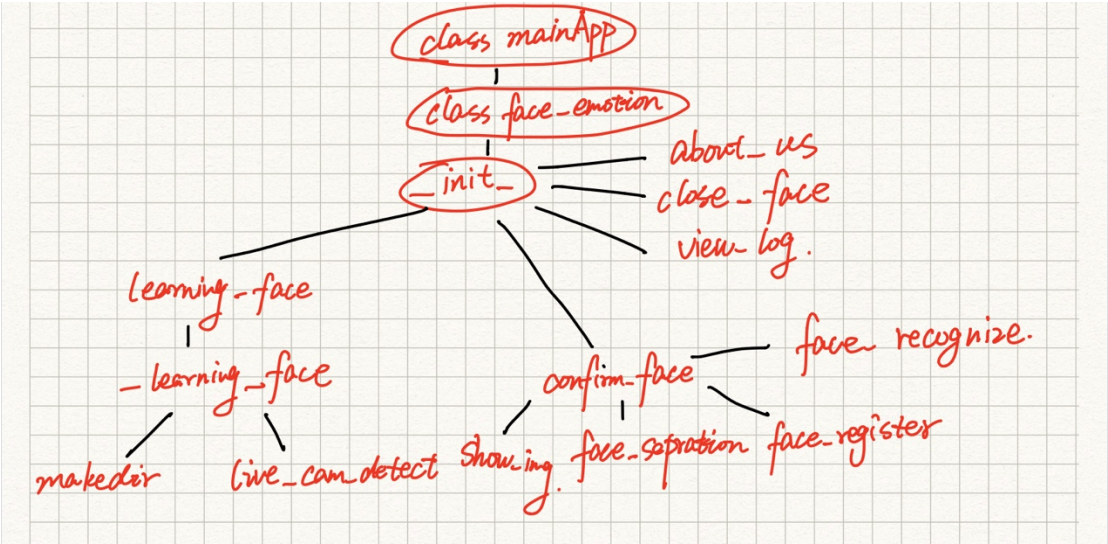
[24] 即写在程序框大类里面的函数，参数基本要求均为(self, event)

```
            :param event: 类要求，实际并未显式调用
            """
f)  def about_us(self, event):
            """
        就是弹出一个简单介绍作者基本信息的框
            :param event: 类要求，实际并未显式调用
            """
```

4. 函数调用关系

文字描述：以最后一个版本的代码为例。整个程序的基础就是 wx 模块框架。首先初始化运行类 mainApp 然后调用该类的唯一函数 OnInit(self)运行 wx frame 基类（class face_emotion）执行__init__画出一个程序框和基础的按键布局。执行 initialize(client)创建表格，执行 clear_images(path_save,path_cache)清除缓存路径下的文件方便进一步操作。然后等待用户操作。”Open Camera”执行 learning_face(self, event)，该函数创建多线程任务并开始执行_learning_face(self, event)开始主程序阶段。执行 makedir()创建基础路径，紧接着执行核心函数 live_came_detect(separate_path, cache_path, self.userlist, self)开启摄像头并等待进一步操作。用户按下 confirm 键，开始执行 confirm_face(self, event)为其进行签到或注册，首先是 face_separate(cc_path, sep_path, detector)函数，分割人脸图像，然后执行 face_recognize(client, read_path)进行识别签到，如果数据库中没有此人，则提示是否注册，如果是，调用 show_img(path)显示陌生人的头像，请求用户输入人命。然后开始运行 face_register(client, read_path, group_id, usr_name)进行注册。点击 close camera 按钮执行 close_face(self, event)关闭摄像头。点击 View Log 执行 view_log(self, event)打开当前状态的考勤表。点击 about us 执行 about_us(self, event)弹出简介消息框。

简易图像描述：



四、 程序运行结果测试与分析

1. 第一版程序展示（无 GUI 界面）
2. 第二版程序展示

五、 结论与心得

六、 程序源码及参考文献

1. 程序源码

该程序现已开源，可自行阅览：https://github.com/BrucePoki/Attendance-system

2. 参考文献及资料

[1]. Davis E. King. Dlib-ml: A Machine Learning Toolkit[J]. Journal of Machine Learning Research 10, pp. 1755-1758, 2009

[2]. Noel Rappin, Robin Dunn. wxPython in action[M]. British: Manning Publications, 2006:1-552.

[3]. coneypo.Python3 利用 Dlib 实现摄像头实时人脸检测和平铺显示 [EB/OL].https://www.cnblogs.com/AdaminXie/p/10317066.html,2019-1-24.

[4]. conetypo.Python3 利用 Dlib 实现人脸检测和剪切 [EB/OL].https://www.cnblogs.com/AdaminXie/p/8339863.html,2018-1-24.

[5]. monster_ygs.Python 创建目录文件夹 [EB/OL].https://www.cnblogs.com/monsteryang/p/6574550.html,2017-3-18.

[6]. 仿佛泣雪如画.如何在 Python 中创建 Excel 表格 [EB/OL].https://blog.csdn.net/qq_41646358/article/details/81292310,2018-7-31.

[7]. yanyingli.python 中的 wx 模块 [EB/OL].https://blog.csdn.net/yaningli/article/details/86736108,2019-2-1.

[8]. Baidu.百度人脸识别 Python SDK 文档[EB/OL].https://ai.baidu.com/docs#/Face-Python-SDK/6e12bec2,2019.

[9]. zhuzaiming2004.将 opencv 的视屏流嵌入 wxpython 的框架中 [EB/OL].http://blog.sina.com.cn/s/blog_49b3ba190102yukm.html,2018-11-12.

[10]. inspurer.WorkAttendanceSystem[EB/OL].https://github.com/inspurer/WorkAttendanceSystem,2018-9.