**目 录**

[1. 课程设计内容 3](#_Toc7030)

[2. 相关技术简介 3](#_Toc22098)

[3. 需求分析 5](#_Toc15015)

[4. 总体设计 7](#_Toc5719)

[5. 详细设计 8](#_Toc21229)

[5.1. 功能模块1：打开相机并录入人脸 8](#_Toc12590)

[5.2. 功能模块2：对录入人脸进行保存 10](#_Toc6515)

[5.3. 功能模块3：获得128D人脸数据 10](#_Toc30566)

[5.4. 功能模块4： 人脸检测器...................................................................................................10](#_Toc30566)

[6. 设计结果及分析 11](#_Toc27285)

[7. 问题及心得体会 12](#_Toc24610)

[8. 对本设计过程及方法、手段的改进建议 16](#_Toc28576)

[9. 参考文献 16](#_Toc6245)

**可多人脸同时识别的人脸识别系统**

# 课程设计内容

本项目是提取 128D 人脸特征，然后计算 摄像头人脸特征 和 预设的特征脸的欧式距离，进行比对

基于常用摄像头，通过调用摄像头对人脸进行拍照，存入制定文件夹,再通过代码将图片进行统一识别处理，将之前捕获到的人脸图像文件，提取出 128D 特征，然后计算出某人人脸数据的特征均值存入 CSV 中，方便之后识别时候进行比对；

利用 numpy.mean() 计算特征均值，生成一个存储所有录入人脸数据的数据库，在后期识别框架中，通过摄像机抓取与数据库中比较，如果欧式距离比较近的话，就可以认为是同一张人脸。并在交互界面上显示名称，并且支持多张人脸同时识别

# 相关技术简介

Python由荷兰数学和计算机科学研究学会的Guido van Rossum 于1990 年代初设计，作为一门叫做ABC语言的替代品。 Python提供了高效的高级数据结构，还能简单有效地面向对象编程。Python语法和动态类型，以及解释型语言的本质，使它成为多数平台上写脚本和快速开发应用的编程语言， 随着版本的不断更新和语言新功能的添加，逐渐被用于独立的、大型项目的开发。

Python解释器易于扩展，可以使用C或C++（或者其他可以通过C调用的语言）扩展新的功能和数据类型。 Python 也可用于可定制化软件中的扩展程序语言。Python丰富的标准库，提供了适用于各个主要系统平台的源码或机器码。

2021年10月，语言流行指数的编译器Tiobe将Python加冕为最受欢迎的编程语言，20年来首次将其置于Java、C和JavaScript之上。

dlib库简介

dlib是一个机器学习的开源库，包含了机器学习的很多算法，使用起来很方便，直接包含头文件即可，并且不依赖于其他库（自带图像编解码库源码）。Dlib可以帮助您创建很多复杂的机器学习方面的软件来帮助解决实际问题。目前Dlib已经被广泛的用在行业和学术领域,包括机器人,嵌入式设备,移动电话和大型高性能计算环境。

Dlib是一个使用现代C++技术编写的跨平台的通用库，遵守Boost Software licence. 主要特点如下：

● 可移植代码：代码符合ISO C++标准，不需要第三方库支持，支持win32、Linux、Mac OS X、Solaris、HPUX、BSDs 和 POSIX 系统 。

● 线程支持：提供简单的可移植的线程API 。

● 网络支持：提供简单的可移植的Socket API和一个简单的Http服务器 。

● 图形用户界面：提供线程安全的GUI API 。

● 数值算法：矩阵、大整数、随机数运算等 。

● 机器学习算法

● 图形模型算法

● 图像处理：支持读写Windows BMP文件，不同类型色彩转换

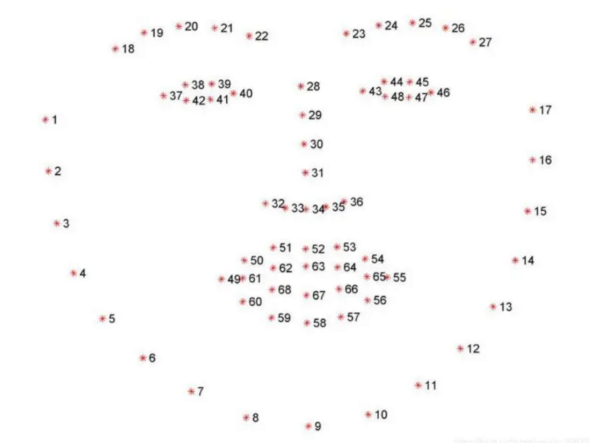
● 数据压缩和完整性算法：CRC32、Md5、不同形式的PPM算法

● 测试：线程安全的日志类和模块化的单元测试框架以及各种测试assert支持

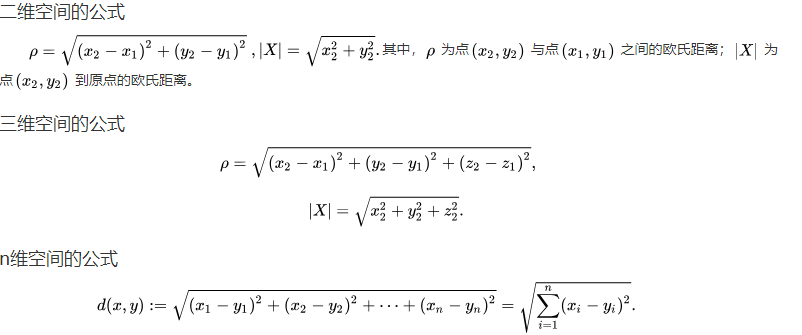
● 一般工具：XML解析、内存管理、类型安全的big/little endian转换、序列化支持和容器类

二 人脸识别

1、dlib库采用68点位置标志人脸重要部位，比如18-22点标志右眉毛，51-68标志嘴巴。

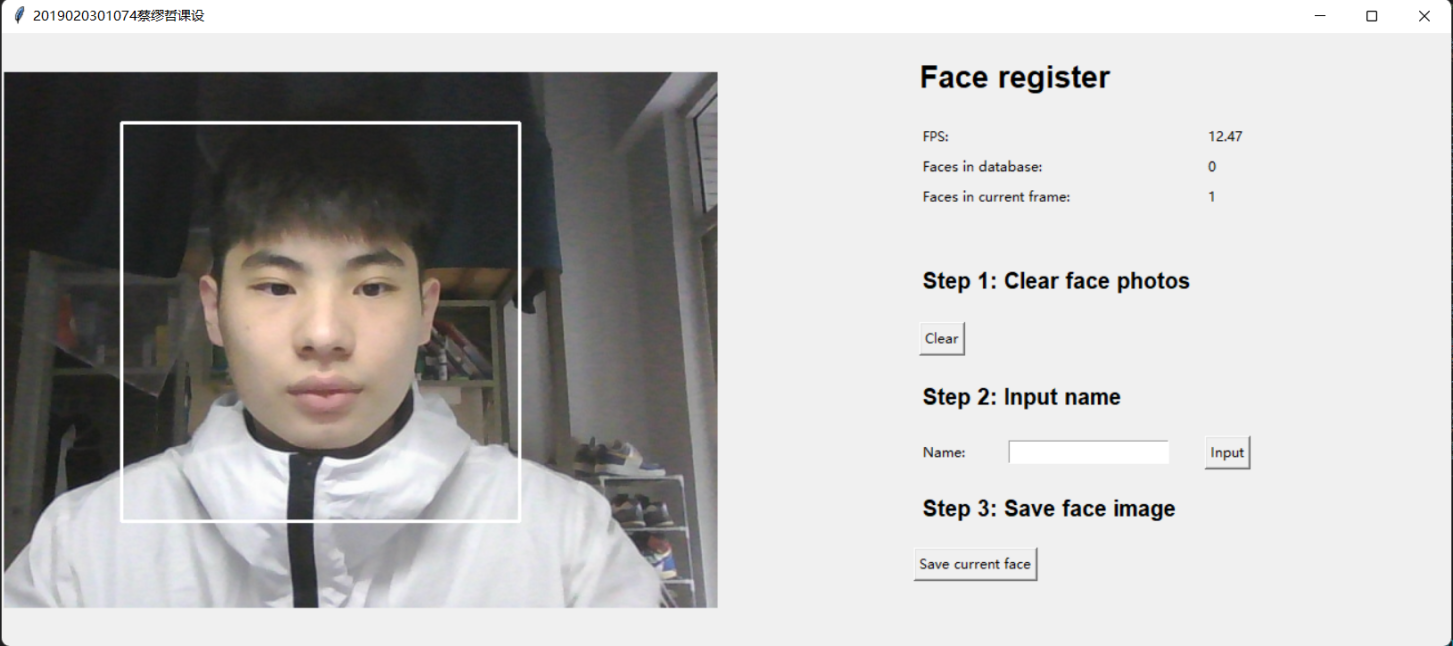


欧几里得度量（euclidean metric）（也称欧氏距离）是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。



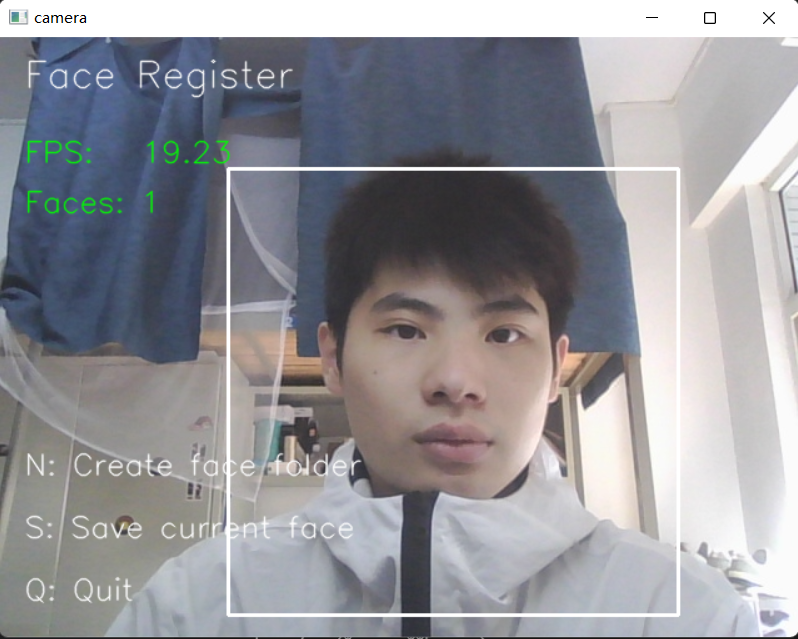
# 需求分析

1. 人脸录入界面, 支持录入时设置 (中文) 姓名



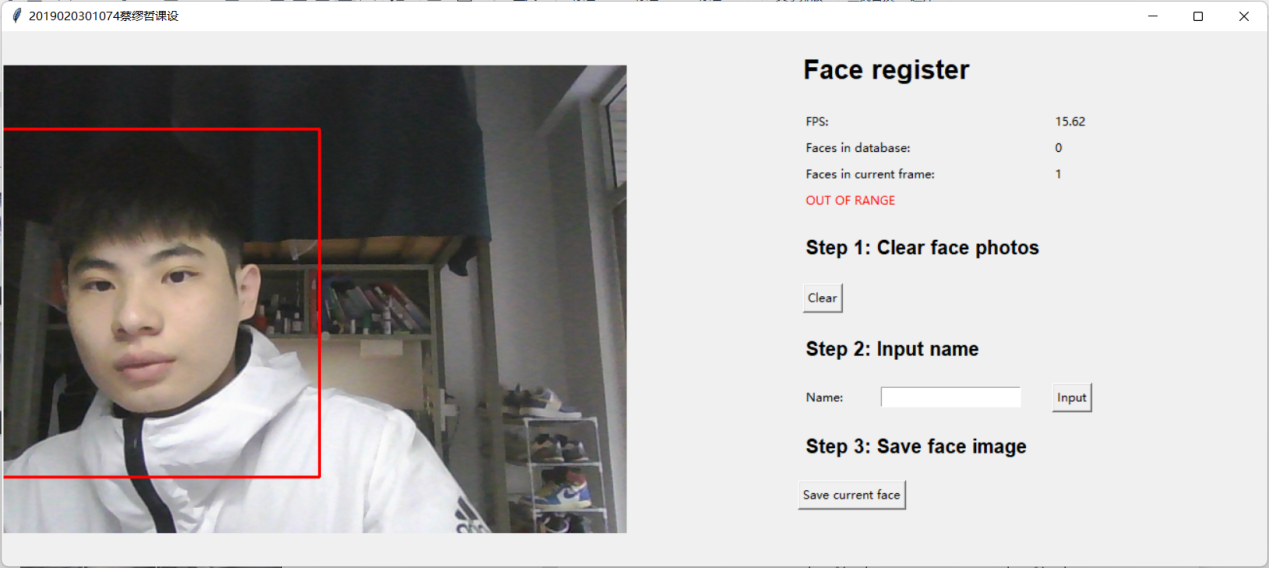
1. 简单的 OpenCV 摄像头人脸录入界面。可以显示人脸数量，刷新帧率

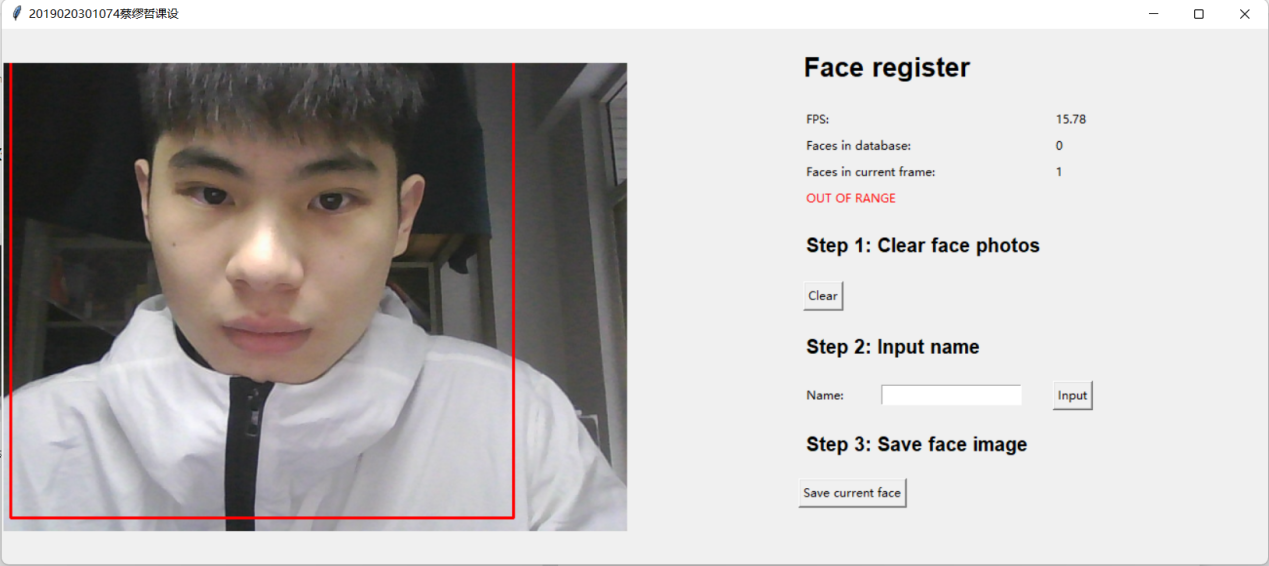
对图像进行保存，和界面退出。



1. 离摄像头过近或人脸超出摄像头范围时, 会有 "OUT OF RANGE" 提醒



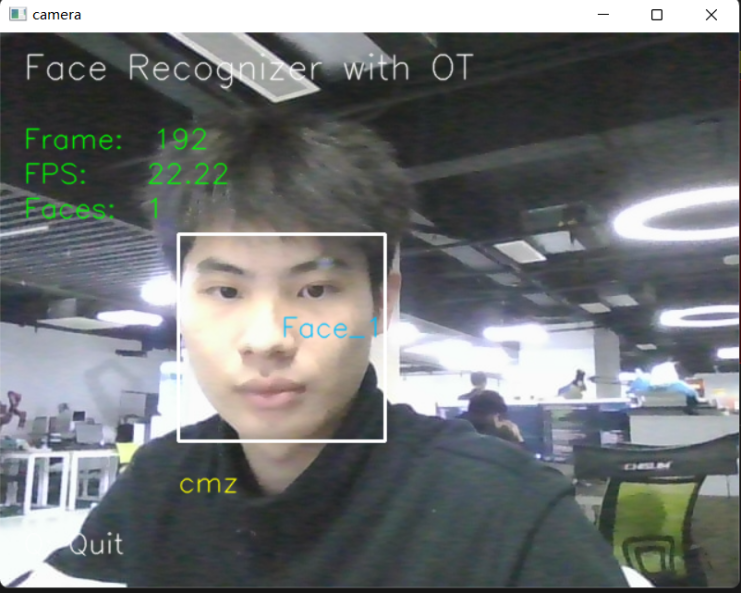




4.提取特征建立人脸数据库利用摄像头进行人脸识别 对于每一帧都做检测识别

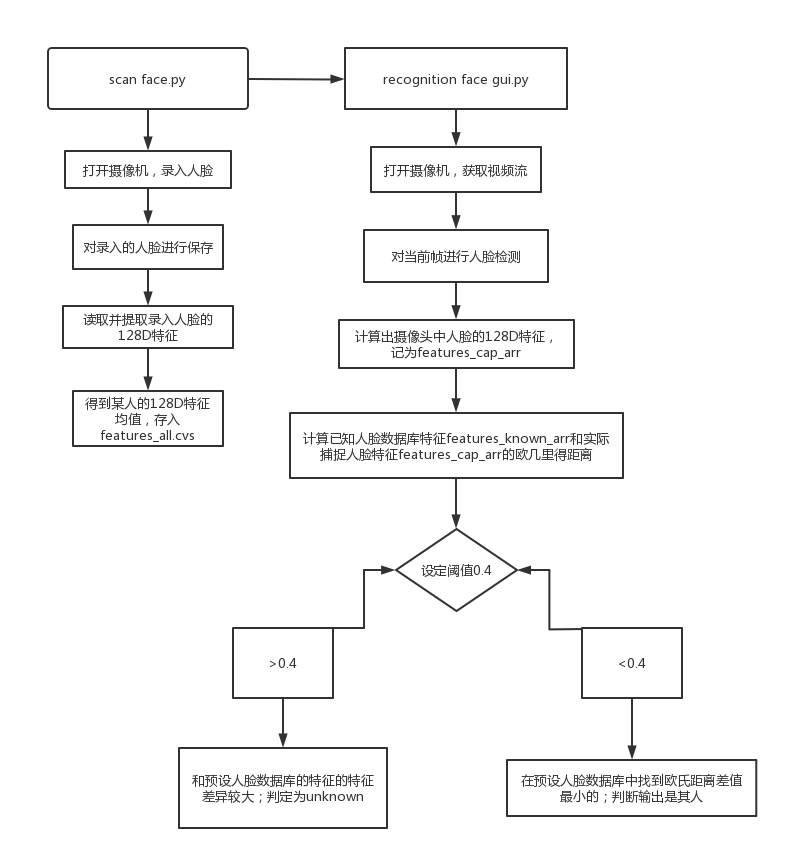


1. Gui界面定制显示名字



# 总体设计

以组织结构图或者流程图的形式给出课程设计作品的功能模块划分，并对各模块做简要解释说明。



# 详细设计

## 功能模块1：打开相机并录入人脸

# Step 1: 清理旧数据

tk.Label(self.frame\_right\_info,

font=self.font\_step\_title,

text="Step 1: Clear face photos").grid(row=5, column=0, columnspan=2, sticky=tk.W, padx=5, pady=20)

tk.Button(self.frame\_right\_info,

text='Clear',

command=self.GUI\_clear\_data).grid(row=6, column=0, columnspan=3, sticky=tk.W, padx=5, pady=2)

# Step 2: 输入要添加人员的名称和创建文件夹

tk.Label(self.frame\_right\_info,

font=self.font\_step\_title,

text="Step 2: Input name").grid(row=7, column=0, columnspan=2, sticky=tk.W, padx=5, pady=20)

tk.Label(self.frame\_right\_info, text="Name: ").grid(row=8, column=0, sticky=tk.W, padx=5, pady=0)

self.input\_name.grid(row=8, column=1, sticky=tk.W, padx=0, pady=2)

tk.Button(self.frame\_right\_info,

text='Input',

command=self.GUI\_get\_input\_name).grid(row=8, column=2, padx=5)

# Step 3: 在帧中保存当前的面部

tk.Label(self.frame\_right\_info,

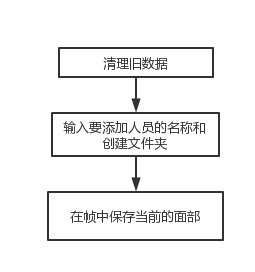
font=self.font\_step\_title,

text="Step 3: Save face image").grid(row=9, column=0, columnspan=2, sticky=tk.W, padx=5, pady=20)

tk.Button(self.frame\_right\_info,

text='Save current face',

command=self.save\_current\_face).grid(row=10, column=0, columnspan=3, sticky=tk.W)



## 功能模块2：对录入人脸进行保存

def pre\_work\_mkdir(self):

# 新建文件夹 / Create folders to save face images and csv

if os.path.isdir(self.path\_photos\_from\_camera):

pass

else:

os.mkdir(self.path\_photos\_from\_camera)

## 功能模块3：获得128D的人脸数据

# Input: path\_img <class 'str'>输入数据库的的图片

# Output: face\_descriptor <class 'dlib.vector'>输出数据

def return\_128d\_features(path\_img):

img\_rd = cv2.imread(path\_img)

faces = detector(img\_rd, 1)

logging.info("%-40s %-20s", "检测到人脸的图像 / Image with faces detected:", path\_img)

**5.4功能模块4：人脸检测器**

首先获取人脸

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor('data/data\_dlib/shape\_predictor\_68\_face\_landmarks.dat')

facerec = dlib.face\_recognition\_model\_v1("data/data\_dlib/dlib\_face\_recognition\_resnet\_model\_v1.dat")

faces = detector(img\_gray, 0)

1. 如果检测到人脸，则 遍历所有检测到的人脸

if len(faces) != 0:

for i in range(len(faces)):

2. 提取当前帧人脸的特征描述子

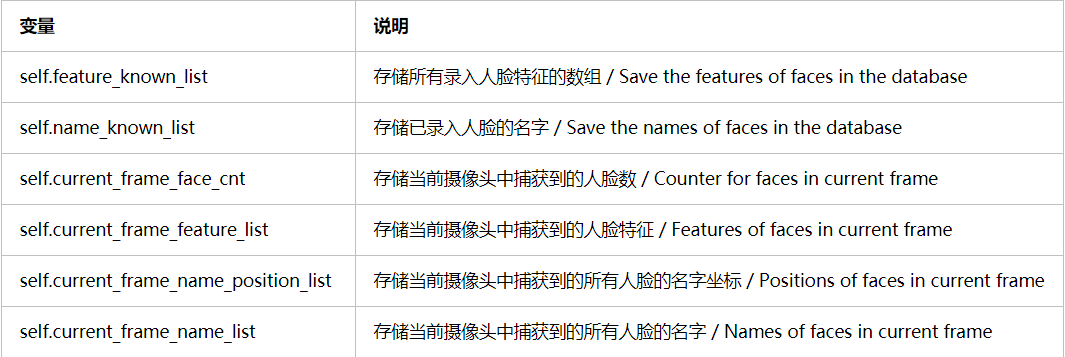
shape = predictor(img\_rd, faces[i])

facerec.compute\_face\_descriptor(img\_rd, shape)

3. 将当前帧人脸特征描述子和数据库的特征描述子进行对比

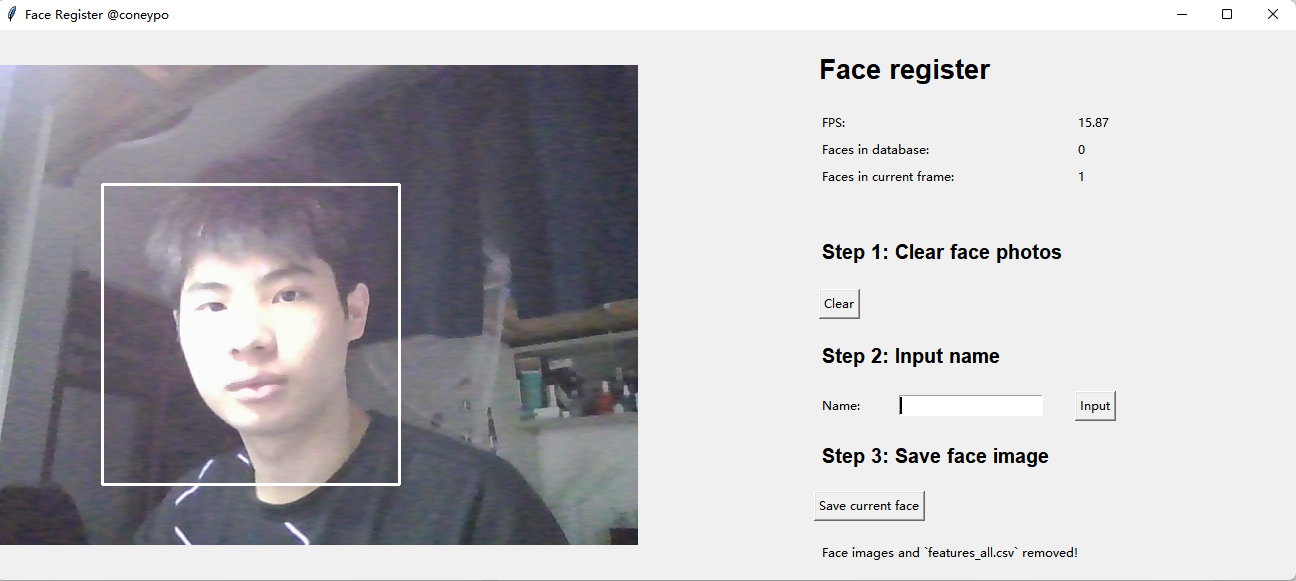
for i in range(len(self.features\_known\_list)):

e\_distance\_tmp = self.return\_euclidean\_distance(self.features\_camera\_list[k], self.features\_known\_list[i])

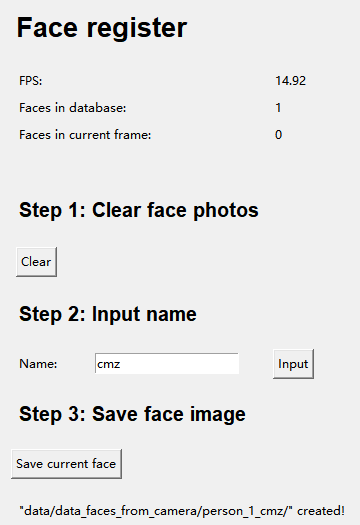


# 设计结果及分析

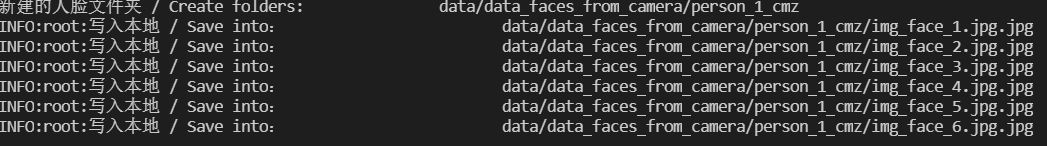
**1.可以先清理原数据库所有数据，如下图**

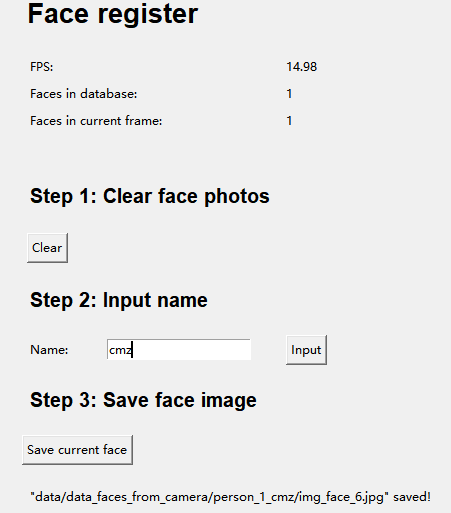


1. **输入被识别人的名称，创建文件夹，如图**

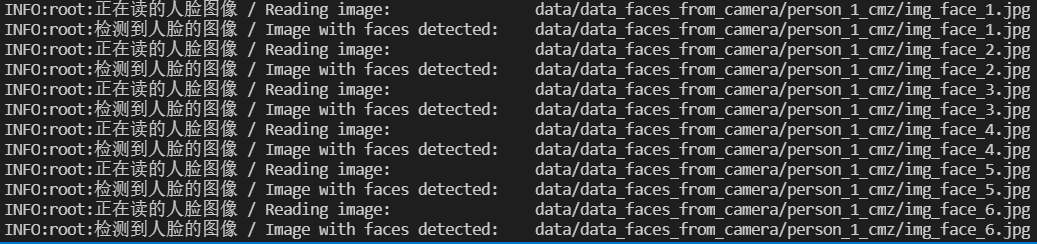


**3对面部进行拍照，并保存到文件夹，如图**



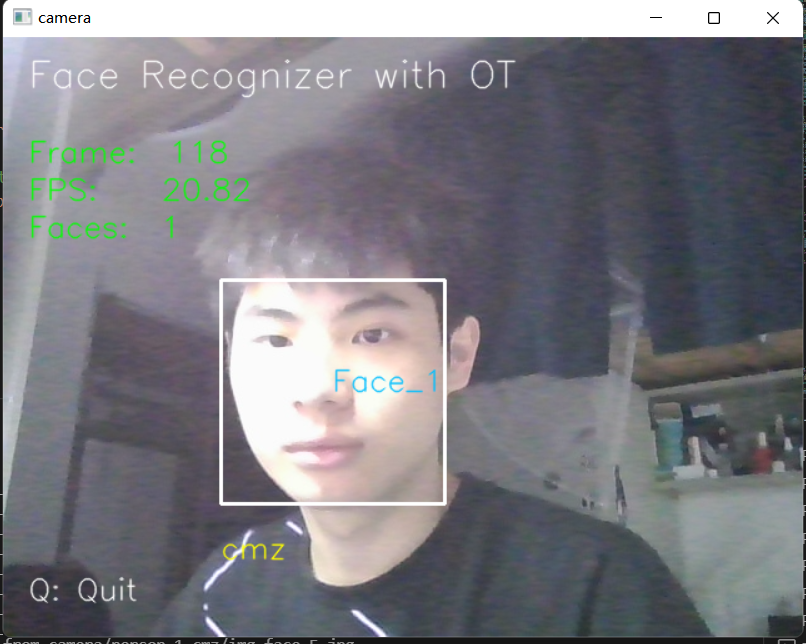


**4.对保存图像进行处理获得128D数据，记录在csv文件中**

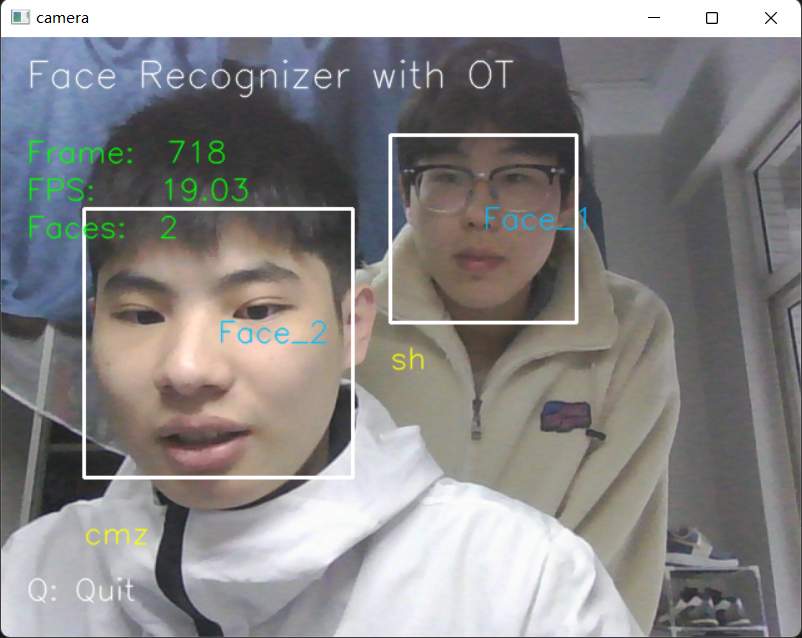




**5.得到结果，识别出我是cmz**



**2）并且可以同时识别多张脸**



**7.问题及心得体会**

在做项目中经常会遇到调用文件打不开，和库调用的问题。就像一开始安装dlib库，由于多次用pip install dlib 报错，选择手动安装并配置环境。用anaconda

先安装boost ，在安装cmake，再设置虚拟环境，再虚拟环境中调用conda 代码设置并安装dlib库。才得以成功。

在软件编写这一块，开始时遇到了很大的难题，因为接近七百行的程序，哪怕有一个错误，修改起来也不是十分方便。时常没有心思去寻找和改正错误，所以软件这一块花了很大的一部分时间。几次调试后总算出现“0 warning”和“0 error”的字样。

除了学会很多东西以为，也发现了自己的不足。由于自己在这一方面只是的短缺，所以很多问题都需要查阅资料或者向别的同学请教。比如vscode的使用方法，都是看着别人操作几遍之后才慢慢学会。做了实物实现，遇到了多的问题，学习到了很多的实践知识。

庆幸设计人脸识别系统的结果是成功的，实现人脸识别功能，各个按键的功能也和预想时一致，看着数字的跳动自己信息也感到十分开心。。此次试验增长了自己的单片机知识外也让我知道实践和理论学习的差别。实践过程中学到的知识自己记得会更加牢固一些，不容易忘记。总的来说，此次人脸识别系统的制作让我收获不小。

# 8.对本设计过程及方法、手段的改进建议

1.在人脸识别的交互界面上，可在做美观一些。

2.后期制作中可以区分真正人脸和图片人脸的差距

# 9.参考文献

[1] OpenCV计算机视觉编程攻略（第三版），Robert Laganiere著，相初根译，人民邮电出版社，2017

[2] 计算机视觉——一种现代化方法（第二版），电子工业出版社，2017

[3] 计算机视觉——算法与应用，清华大学出版社，2012