## 需求分析

选题“员工刷脸考勤”，要求采用python语言开发，可以通过摄像头添加员工面部信息，这里就涉及到两个具体的个问题，一个是应该以什么样的数据来标识每一个员工的面部信息，二是持久化地保存这些信息到数据库中去。更细地，还涉及表的设计；另一个基本要求是通过摄像头识别员工面部信息来完成考勤，这个问题基本可以通过遍历数据库里的员工面部数据与当前摄像头里的员工面部数据的比对来实现，但有一个问题就是假如摄像头里有多张人脸改怎么处理。扩展要求是导出每日的考勤表，可以拆分为两个部分，一个是存储考勤信息，一个是展示考勤信息。

我们希望达到的目标是：

（1）仿照通用型软件界面设计的原则，所有的操作都在菜单栏里实现，一部分区域用于展示摄像头实时读取并由程序加工后的视频流信息，另一部分区域做控制台输出，打印相关信息，比如提示员工面部信息添加成功、添加失败及其原因，提示员工打卡成功、打卡失败及其原因；添加面部信息时人是必须和程序进行交互的，比如输入一些相关的信息，这个时候程序是阻塞的；但是在打卡的时候，程序是不阻塞的，如果不点击关闭打卡，它会一直在打卡的模式，等待并识别每一个前来打卡的员工，这比较符合现实的使用场景。

（2）建表来存储员工信息和考勤信息，每次新建录入员工面部信息时，要求输入工号、姓名，并查无重后方可录入，录入时只取距离屏幕最近的员工的面部信息，这是考虑到实际打卡都是依次进行而不是一群人一群人地打卡，录入时有两种模式可供选择，自动模式：一旦识别到人脸就自动捕获截图，连续截图达到10张就结束录入；手动模式：点击菜单结束录入，不一定要得到10张。结束录入后就开始从刚才捕获的员工面部提取人脸特征数据并连同之前输入的员工姓名等数据作为一行记录保存到数据库中，如果没有捕获到员工面部或者刚才捕获的面部信息不是同一个人，这行记录就被丢弃。

考勤时，须满足三个条件：面部信息已录入、在打卡时间段内、未重复打卡，只有打卡成功，打卡人姓名及工号、打卡日期及时间才会被当成一行记录保存到数据库并在控制台输出打卡成功信息，否则会在控制台输出失败及其原因信息。

总而言之：我们的设计目标是规范化、人性化。

## 总体设计

为了完成上述目标一，程序的界面初始化分为三部分，第一部分初始化菜单栏，第二部分初始化左边控制台，第三部分初始化右边展示面板，使这三部分相互独立；数据逻辑部分的初始化分为两部分，第一部分是数据库部分的初始化，如果数据库/表不存在就新建，存在则加载相关数据，第二部分是初始化一些需要循环使用的变量，比如新建录入时的员工姓名、工号、截图数目计数器等,每当完成录入时这些数据都应该被重置成初始化以待下一次录入，把这些初始化语句写成一个函数可以提高代码复用度。

上述目标二主要是一些限制性条件，可以通过添加判断语句来实现，比如对输入id的合法性检验：

while self.id == ID\_WORKER\_UNAVIABLE:

self.id = wx.GetNumberFromUser(message="请输入您的工号(-1不可用)",prompt="工号", caption="温馨提示", value=ID\_WORKER\_UNAVIABLE, parent=self.bmp,max=100000000,min=ID\_WORKER\_UNAVIABLE)

for knew\_id in self.knew\_id:

if knew\_id == self.id:

self.id = ID\_WORKER\_UNAVIABLE

wx.MessageBox(message="工号已存在，请重新输入", caption="警告")

其中ID\_WORKER\_UNAVIABLE是id的初始化值-1，不可用，self.knew.id是从数据库里加载出来的id列表，如果id非法（已重复或者不在0~100000000，就会一直有新的弹窗来提示输入id。

再比如对拒绝多张人脸时、只处理距离屏幕最近的员工的面部信息：

if len(dets) != 0:

biggest\_face = dets[0]

#取占比最大的脸

maxArea = 0

for det in dets:

w = det.right() - det.left()

h = det.top()-det.bottom()

if w\*h > maxArea:

biggest\_face = det

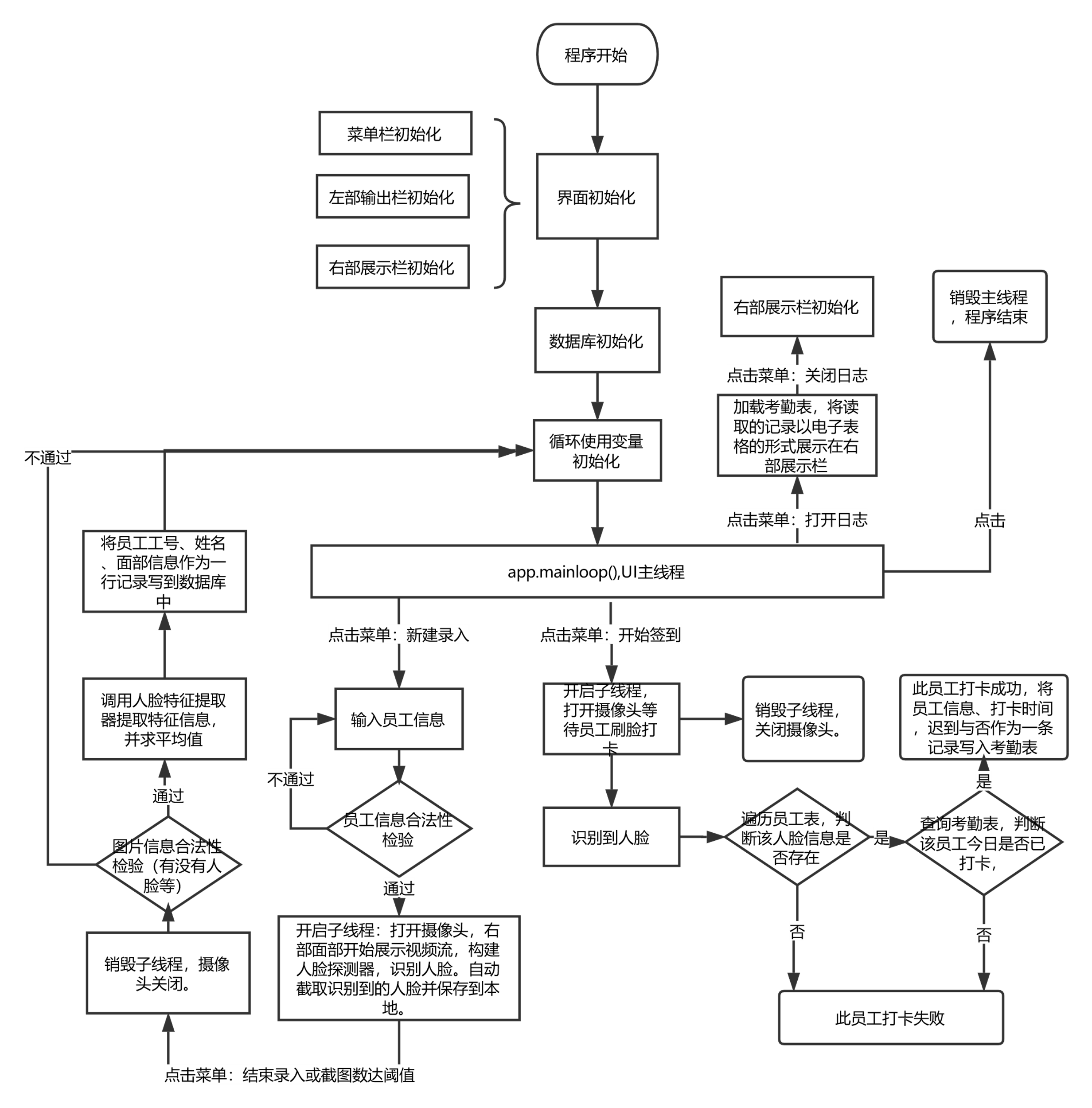
maxArea = w\*h

dets是侦测到的所有面部数组，biggest\_face是距离屏幕最近的面部。

本次课设的完整过程如下：选了员工刷脸考勤系统以后，我和小组成员刘佳璇查阅了大量资料，了解到dlib这个库，于是我们利用dlib库自带的人脸预测器、特征提取器，然后计算128d特征来完成人脸识别，再加上我们之前构想的逻辑，大致完成了课设，由于完成得早，我们大致在课设安排时间的一半时就给老师检查，虽然基本要求和拓展要求都已经实现，但是还有许多可以完善的地方，老师给出了许多宝贵的意见：

1. 把数据保存的地方由csv文件改成数据库。
2. 减少弹出式界面，用frame替换的方式解决之。
3. 刷脸打卡的界面不需要打卡者交互，全自动打卡。

最后检查的时候这些问题都被解决了。

程序框图：

注：图片如看不清也可在线预览：<https://www.processon.com/view/link/5bbcc953e4b08faf8c7324a1>

本程序的设计思想大致可分为以下几个方面

1. 面向对象的原则，整个程序的主体就是一个WAS(WorkAttendanceSystem)类,所有的实现都围绕这个类展开。
2. 界面和数据逻辑分离的原则，WAS类的初始化过程包括界面的初始化和数据初始化，两者相互独立。
3. 代码封装原则，多次调用的语句集写成接口供调用，没有冗余的代码。
4. 接口隔离原则:使用多个专门的接口,而不是使用单一的总接口。

## 详细设计

函数清单

注：所有类内的函数的第一个参数为self，表明该函数属于该类，后面不再赘述

1. def \_\_init\_\_(self)

WAS类的构造函数，主要是完成一些初始化操作，如初始化菜单、信息打印面板、主展示面板以及初始化加载数据库、初始化循环使用的变量。

1. def initMenu(self):

完成菜单的初始化显示，点击事件绑定。

1. def initInfoText(self):

完成左边信息提示面板的初始化显示。

1. def initGallery(self):

完成右边主展示面板的初始化显示。

1. def initDatabase(self):

数据库的初始化，建立数据库连接（如果数据库inspurer.db不存在则先新建），如果数据库中不存在员工信息worker\_info和考勤logcat这两个表，则依次创建。

1. def loadDataBase(self,type):

该模块函数完成从数据库读取数据的操作，包括读取员工信息和考勤信息，第二个参数type用于标识是加载员工信息还是考勤信息，一方面，可以统一接口，打开数据库和得到游标、关闭连接是一样的，将两个读取接口合二为一，提高代码复用度；另一方面，可以减少加载的工作量，减少IO,提高程序运行速度；最后，因为读取信息前对上一次读取的信息列表做了清空处理，用type标识可以避免读取一个表时对另一个表造成的误操作。

1. def insertARow(self,Row,type):

该模块函数完成写数据库操作，第二个参数为准备写的一条记录，第三个参数type表示要对哪一个表进行写操作。

1. def adapt\_array(self,arr):

将提取的人脸特征信息（列表）压缩，入口参数就是待压缩的数据，出口参数是压缩后的数据，用于写入数据库。

1. def convert\_array(self,text):

将读取出来的数据解压缩成人脸特征信息，入口参数是待解压得数据，出口参数是解压后的数据。

1. def return\_euclidean\_distance(feature\_1, feature\_2):

计算两个人脸的欧式距离，入口参数是两个人脸的特征数据，出口参数是判定的结果，欧式距离大于0.4判为不同，不大于判为相同。

1. def OnNewRegisterClicked(self,event):

见名知义，菜单新建录入的监听事件，参数event为事件信息，其他几个菜单的(OnFinishRegisterClicked,OnStartPunchCardClicked, OnEndPunchCardClicked, OnOpenLogcatClicked,OnCloseLogcatClicked)类似，在此不再赘述。

1. def getDateAndTime(self):

得到当前日期和时间，并组装成特定格式作为出口参数返回。

函数调用关系:箭头指向被调用者

