1，人脸识别库的选择，超过99%的都可以，简单一些，特征点为68个点，标注及训练都快一些。

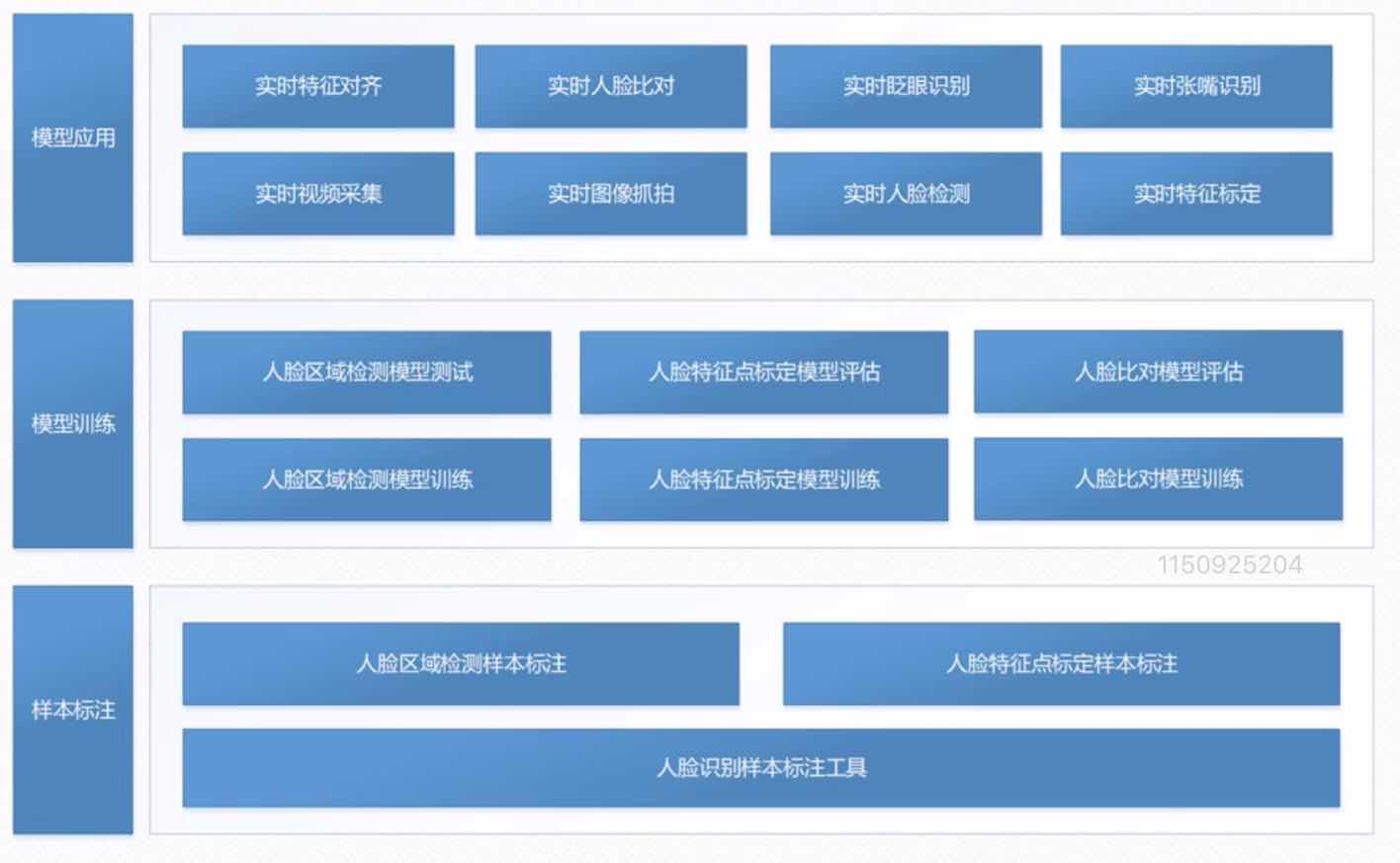
2，架构设计：



3，关键技术



4，项目需求：



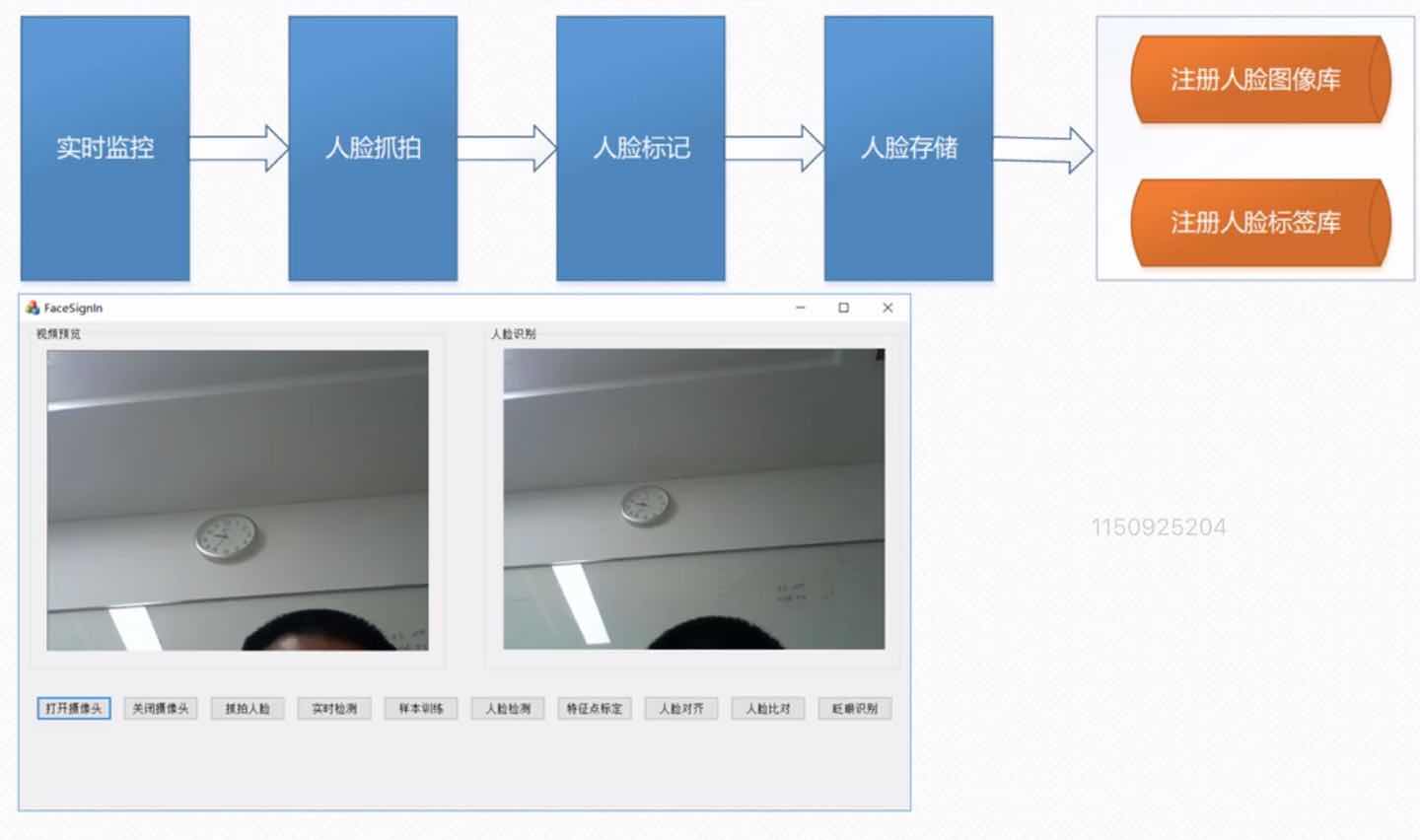
人脸检测

人脸对齐

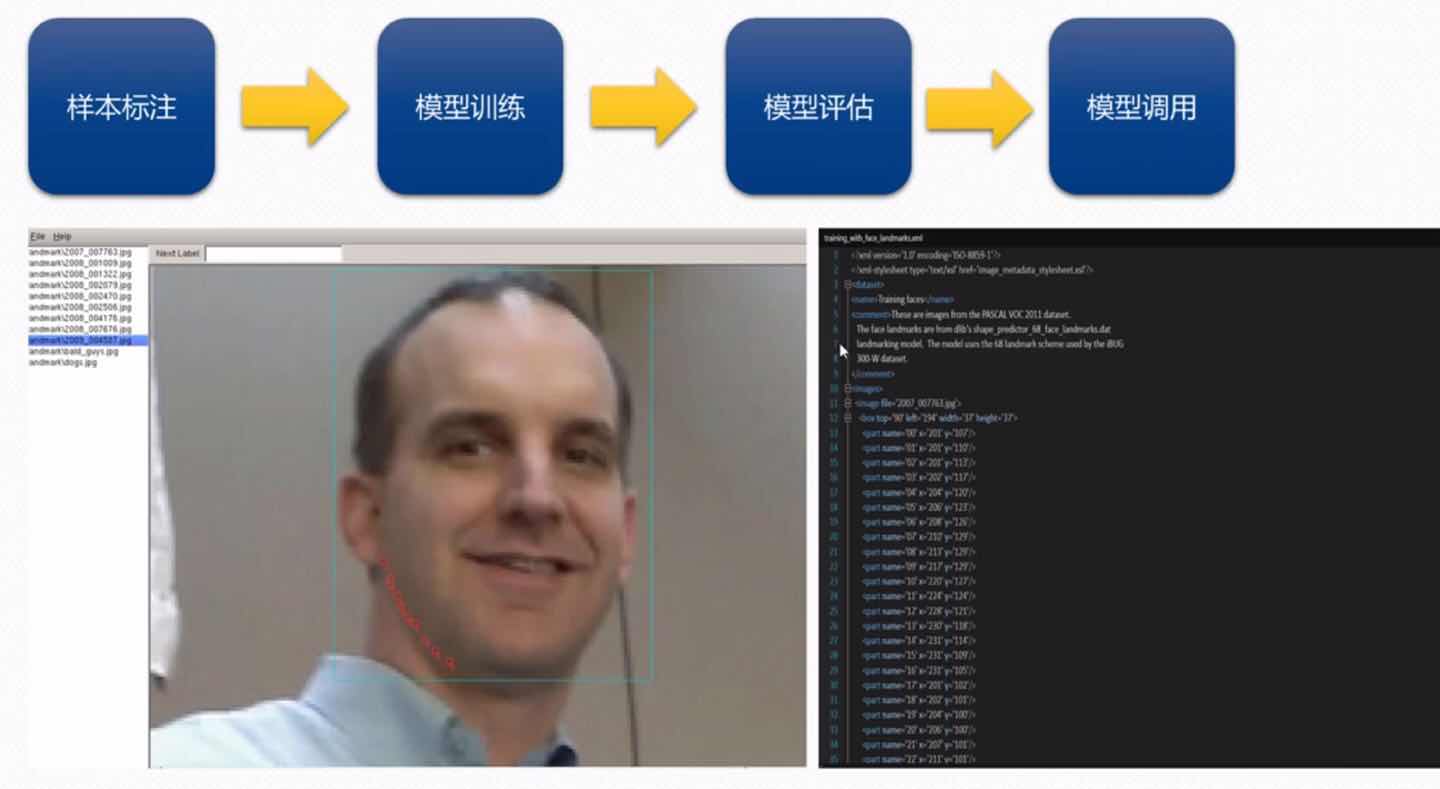
人脸验证：比对+活体检测

对于人脸签到系统，要有注册+实时识别

注册的功能类比与抓拍



5，实施流程



1，实时视频采集程序设计：(打开摄像头，关闭摄像头)

就是实时读取视频每一帧，再挂到左侧图像控件上面显示。

2，实时图像抓拍程序设计:（抓拍人脸）

将m\_dst全局变量的图片，存储在给定路径

再将其挂到右侧图像控件上显示

这里面有个坑，展示图像这个函数要重新定义

即定义了一个ShowImage(pDC, picpath, 0, 0);是在社区找的

3，人脸检测程序设计：（实时检测，人脸检测）

实时检测：基于实时视频流检测人脸

按钮事件就是保证m\_bdetect标志一直打开

人脸检测：检测，并且画矩形框，按钮事件仅仅是标识符声明的01置换 ,如m\_detect以及m\_brectangle;具体代码在打开摄像头里面，if（m\_bdetect）

总体来讲就是读取每一帧，存到cimg，再用detector检测，绘制矩形框，再显示出来。

4，特征点标定程序设计：（特征点标定）

按钮事件只是控制了m\_bdetect标识符

总体来讲：首先m\_capture获得每一帧写入frame，然后基于68个点的位姿模型（集成了检测与特征点标注）进行人脸检测，存到face对象里面（矩形），再将68个点存入到shapes，然后对于shapes进行特征点绘制cv::circle并且打上序号putText。

5，实时特征点对齐程序设计：（人脸对齐）（将歪的人脸矫正）

两种实现方式：一种是单张图片对齐，一种是基于视频流实时对齐。

总的来说：读取抓拍到的图片，初始化环境即68个点的位姿模型，人脸检测，68个特征点存入shapes，特征点标定，再进行人脸对齐的操作，用dlib::extract\_image\_chips对齐，存在face\_chips并且保存为图片，最后显示在控件上面。

6，实时目标跟踪程序设计：（目标跟踪）

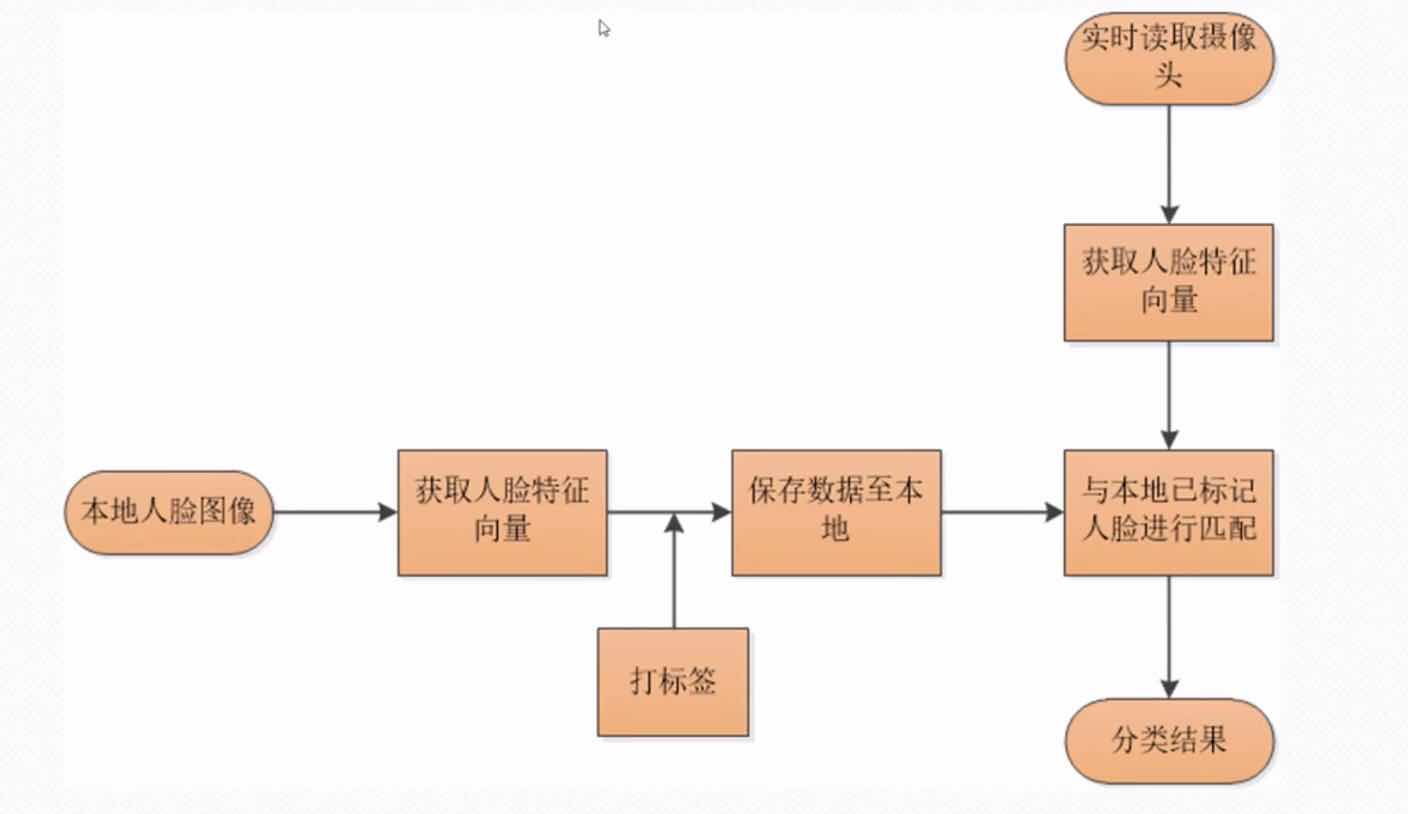
首先一个坑是实时目标跟踪，m\_tracker，这个要为public全局变量，不能放在函数里，否则可能被释放掉，还有一个全局变量是一个窗口有m\_win。

按钮事件只是控制了m\_btrack标识符

总的来说，标识位m\_track,首先开始跟踪，要确定第一帧当前的对象是谁，确定目标区域要跟踪的对象centerd\_rect，然后m\_tracker.update(cimg)，最后显示在窗口m\_win上面，更新overlay就是再绘制跟踪的矩形框。

程序调试时候，在打开摄像头代码里的目标跟踪代码，将m\_win注释掉，并且在.h文件里，将m\_win设置为全局变量。

7，实时人脸比对程序设计：（人脸比对）（一般阀值取0.6？？？）



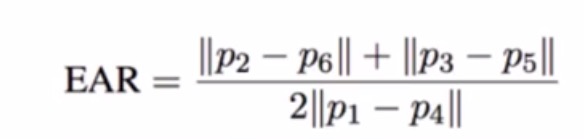
实时的按钮事件只是控制了m\_brecognition标识符

离线比对，总的来说，第一步初始化检测器和图片，导入68特征点标注模型与人脸特征表达模型；第二步，读取两张图片（可以一张是视频流的，一张是人脸库的）并且显示；第三步，人脸对齐（综合第二步，第三步可以理解为用 “68特征点模型”来人脸检测+对齐）；第四步，用“resnet模型”提取人脸特征；第五步，进行欧式距离计算，并显示在文本静态区及弹框显示，这里欧式距离为0.25可以理解为75%的相似度，工程上一般取70%就是同一个人。

7，活体检测之眨眼识别：（眨眼检测）

按钮事件只是控制了m\_beye标识符

总的来说，眨眼识别算法是根据EAR算法原理，即纵横比。公式就是



第一步，取shapes的37，38，40，41，36，39六个点

第二步，计算纵横比

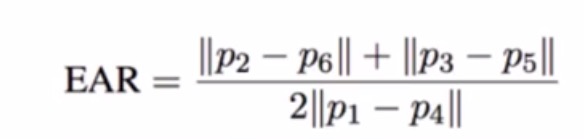
第三步，正则表达显示方式str.format，再显示在静态文本区。

调试流程：打开摄像头-----特征点标定----眨眼识别

9，活体检测之张嘴程序设计：（张嘴识别）

按钮事件只是控制了m\_bmouth标识符

总的来说，眨眼识别算法是根据EAR算法原理，即纵横比。公式就是



第一步，取shapes的共计六个点，纵向：50，52，58，56，横向：48，54

第二步，计算纵横比

第三步，正则表达显示方式str.format，再显示在静态文本区。

调试流程：打开摄像头-----特征点标定----张嘴识别