# 项目名称：Expression

**项目成员：** 彭天祥 201730683314

谢晓民 20173068

申浩 20173068

吴岚锋 20173068

张利丹 20173068

**项目摘要：**

为实现将输入的人脸图片进行检测，并将其面部五官翻转，输出翻转后的图片

**项目设计：**

原理说明：

基本想法是要将人脸与周遭的像素差距凸显出来，因此需要提取人脸的特征，制成弱分类器，但是单个弱分类器的正确率较低（在实验样本数量下基本单个最低错误率在17~26%），因此将多个弱分类器同时使用，赋予其各自的权重系数，错误率越低，权重越高。

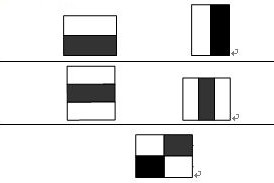
采用Haar分类器，总体组成是：5个Harr特征+积分图方法+AdaBoost算法。

仅引用了部分opencv的基本函数，如图片读入输出以及矩阵的基本功能,并未直接调用其实现好的人脸识别函数，而是（基本）全部自行实现。

代码分为训练版本以及使用版本，由FaceID.h中的宏控制

训练版本说明：

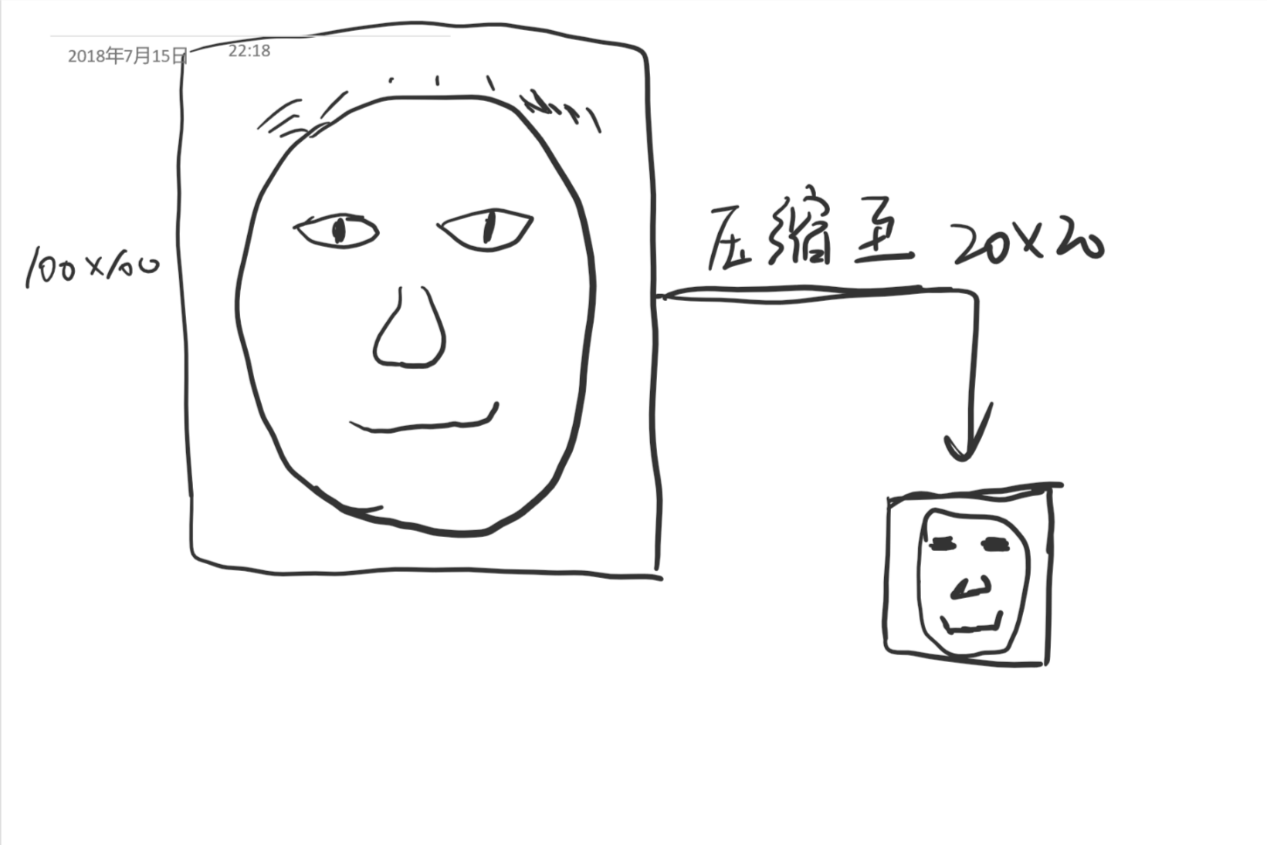
使用了五个Haar特征模型，分别如下



在FaceID.h文件中也有描述

关于这些特征的作用，请看以下。

假如给定一张人脸，原图分辨率为100\*100



为了降低数据量的数量级，一律压缩至20\*20处理，在经过压缩后，图像的很多细节都被略去了，但是，由于眼睛等器官颜色较深，所以信息仍被保留，我们怎样获得这些信息呢？

如果能有一些框框住眼睛，如下：

就可以将特征提取出来，因此就有了Haar特征，我们选取了其中五个模型

由于不同图片眼睛鼻子大小，所以原则上这些框应该可大可小，而又因为不同图片脸的位置不同，这些框的位置又应该可变。

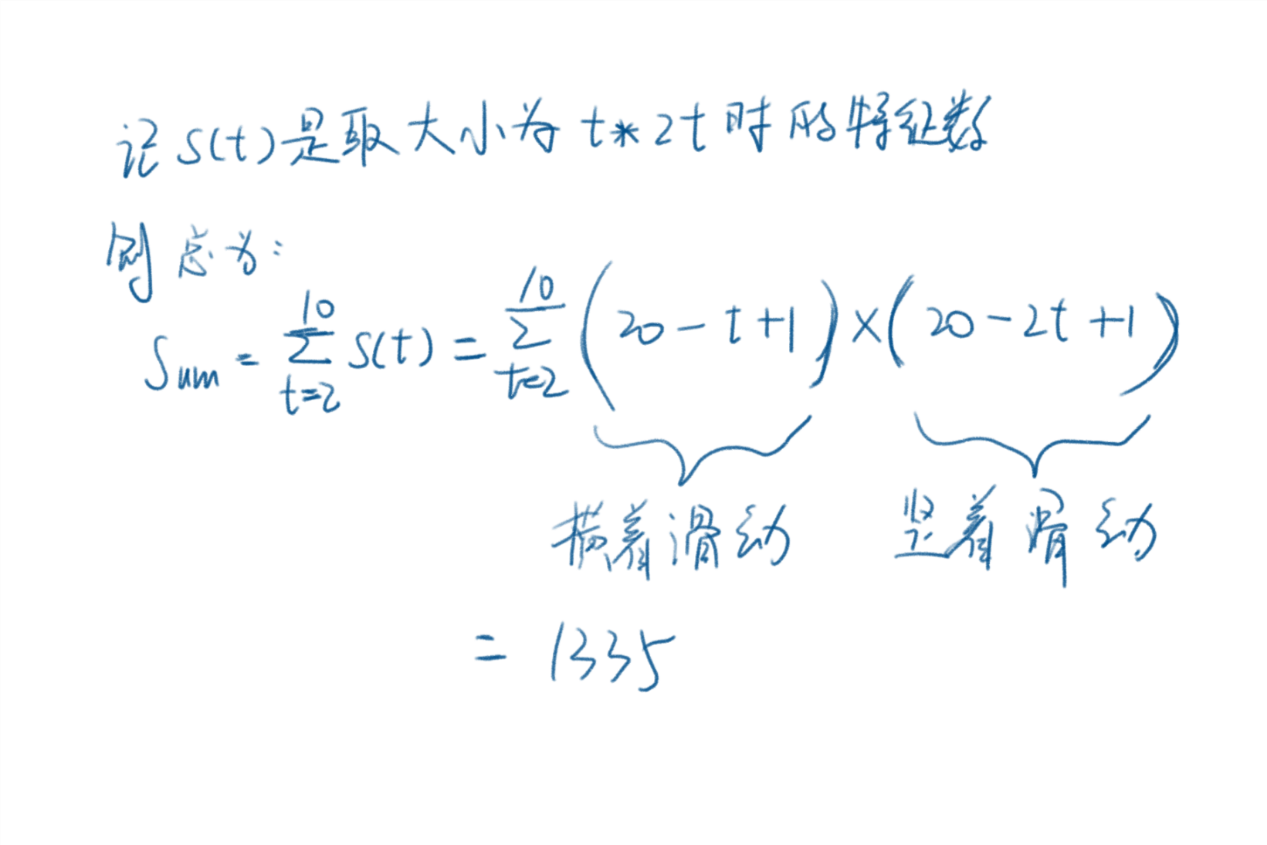
基于以上两个猜测，每个模型就会衍生出许多大小、位置不同，但相似的特征，这正是我们提取特征的依据。

那么应该怎样计算特征值呢？根据Haar特征的文献，是取阴影区域的像素减去空白区域的像素，我们也正是这么做的。

那么这样应该会有多少个特征呢？我们可以简单地计算一下第一种模型在20\*20的图片大小下的特征数

模型的横纵比为1：2，那么特征的大小可以取 2\*4 ,3\*6…….10\*20

此处不取1\*2是为了避免过拟合，对其他模型也是如此，那么如下：



对第一行中的两个模型设置最小面积为3，第二行为4，第三行为5

实际上这样取得的特征数大约有5200多个

值得注意的是，我们也尝试使用100\*100的图像来提取特征，希望能提高精度，但是那样提取出的特征数高达78万个，训练时间过长，而且由于样本数只有约2000个，得到的结果过拟合十分严重，综合考虑，仍然使用20\*20的大

基本思想是AdaBoost的迭代算法，进行50次迭代，每次迭代从所有特征中选取错误率最低的弱分类器，将其选出，并调高被该分类器分错样本的权重，进行下一次迭代，通过50次迭代，选出50个最优的弱分类器（特征），根据它们的错误率分配权重，组成一个强分类器，供使用版本侦测人脸。

代码逻辑如下：

1. 获取样本，正样本1000个，负样本1000个，由FaceID.h中的宏控制
2. 计算每个样本的积分图
3. 用20\*20的尺寸根据5个模型生成特征（约5200+）
4. 初始化一部分变量
5. 训练

训练的基本过程如下：

样本数为m

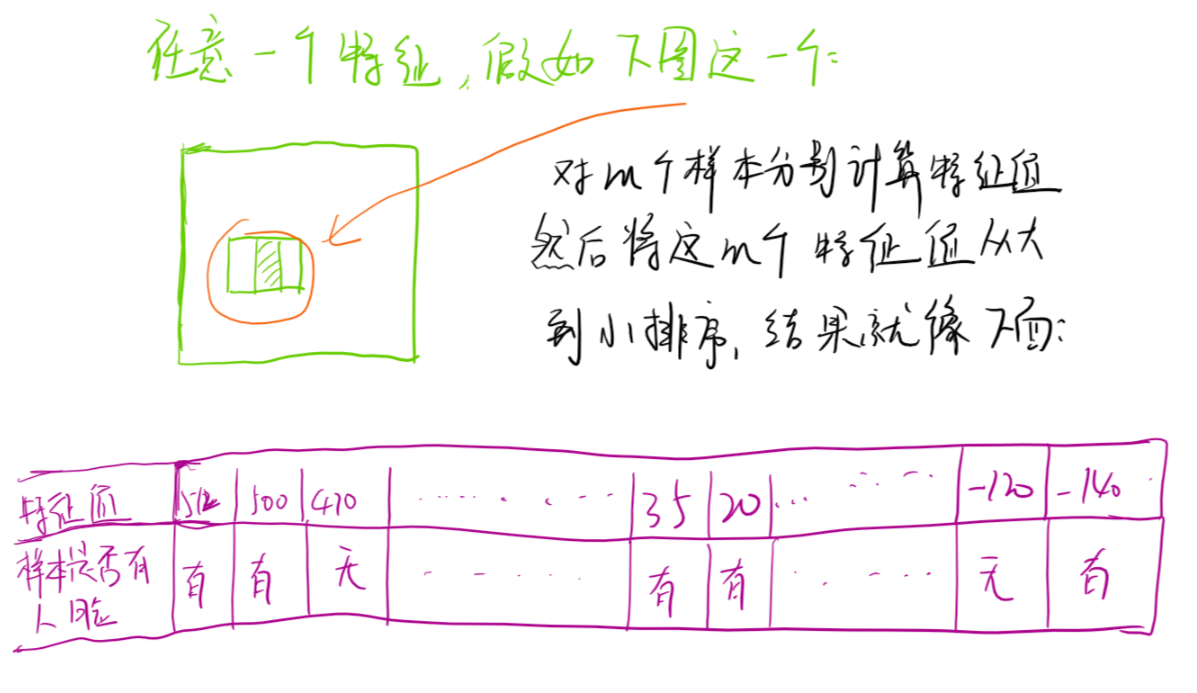
初始化所有样本的权重为1/m（由宏控制）（即平权）

进行k(设为50)次迭代，每次迭代进行如下步骤

1. 计算每个特征的最低错误率，并记录该值

（此处说明最低错误率的计算方式：

1. 计算每个样本在该特征下的特征值
2. 对这m个样本的特征值根据从高到低排序



1. 初始化minWrong=1，对这些根据特征值排好序的样本从第一个到第m个进行一次遍历

遍历过程中进行如下计算，对第i个样本

1. 更新在该样本之前所有有人脸样本（正样本）的权重和\_\_SP以及所有非人脸样本（负样本）的权重和\_\_SN
2. 更新在该样本之前所有正样本的权重和\_\_SN

计算如下值

curWrong=Min(\_\_SN+\_\_TP-\_\_SP, \_\_SP+\_\_TN-\_\_SN)，如果curWrong小于minWrong就更新minWrong并记录这个位置

（此处进行说明，我们通过遍历到底想得到什么？

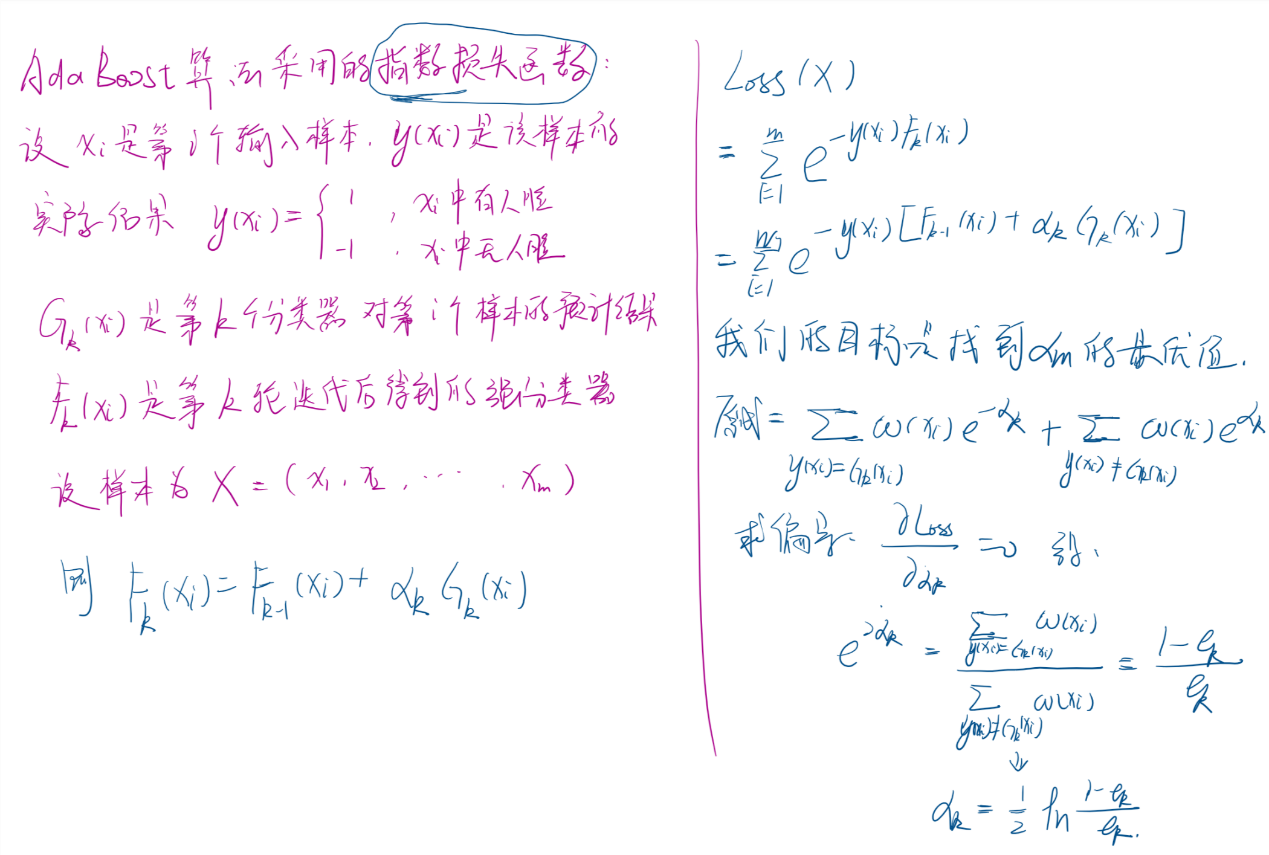
其实就是一个阈值，我们要找到这个特征的最佳阈值，我们可以说，特征值大于这个阈值的判定为人脸或者特征值小于这个阈值的是人脸。但不管怎么划，都会有一部分样本被划错，这也是为什么我们这个阶段选取的叫做弱分类器的原因了，因为它其实比较粗糙，只是根据数据拟合出了在这些数据样本下的一个最佳阈值点，拿它去判断一张新的图片。）而我们在遍历过程中计算的这个Min是以当前特征值为阈值时的错误率，我们遍历就是为了找到取得最小值的位置以及最小出错率

4. 取得最低错误率以及阈值，记录不等号方向，用p控制

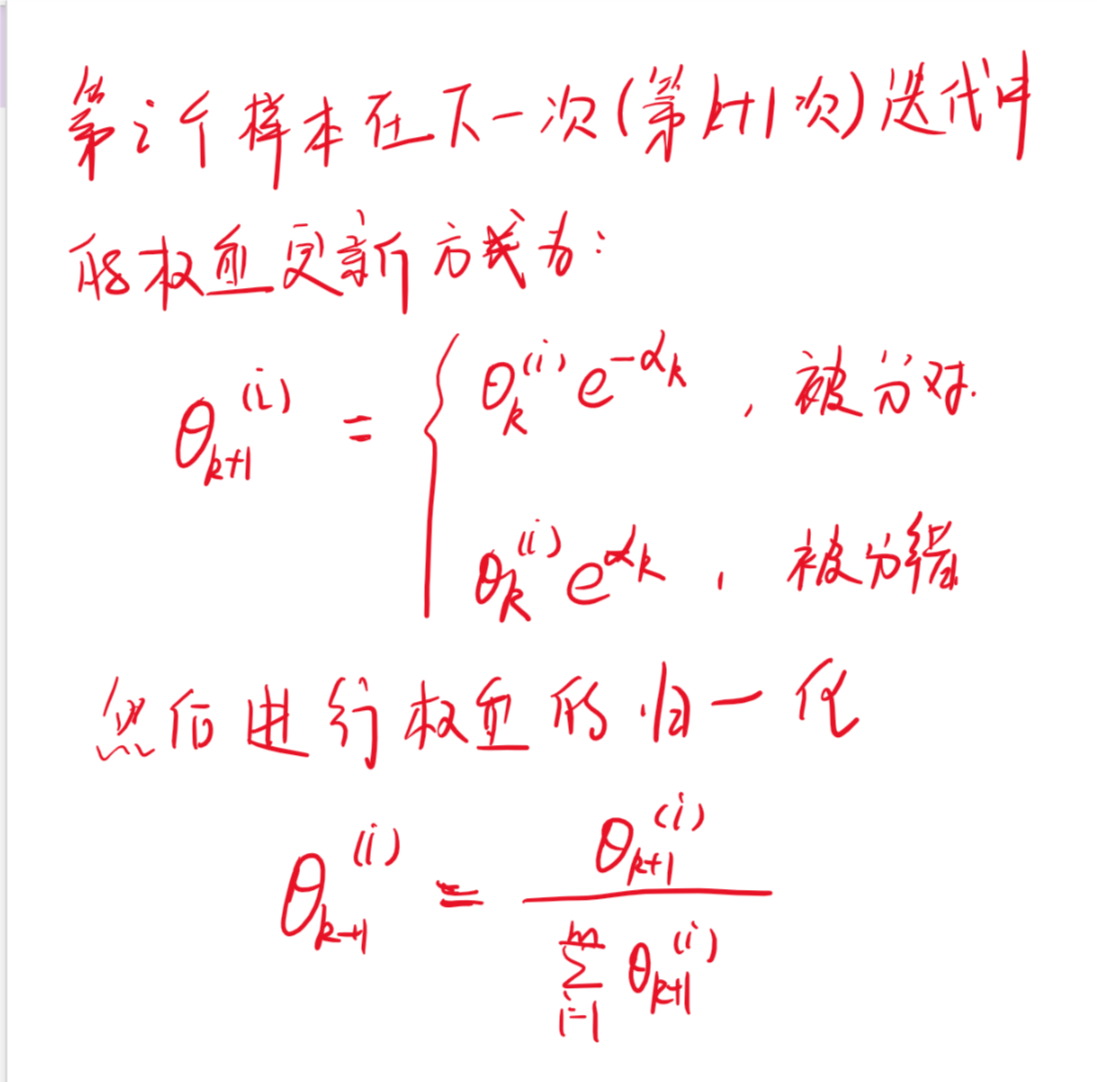
2. 对所有特征的错误率进行排序，找到最小的那个，就是这次迭代选择出来的最优弱分类器（当然，要先确认是不是已经选择过了）

然后，设这个弱分类器在最终的强分类器中的权重为

具体的推导如下：



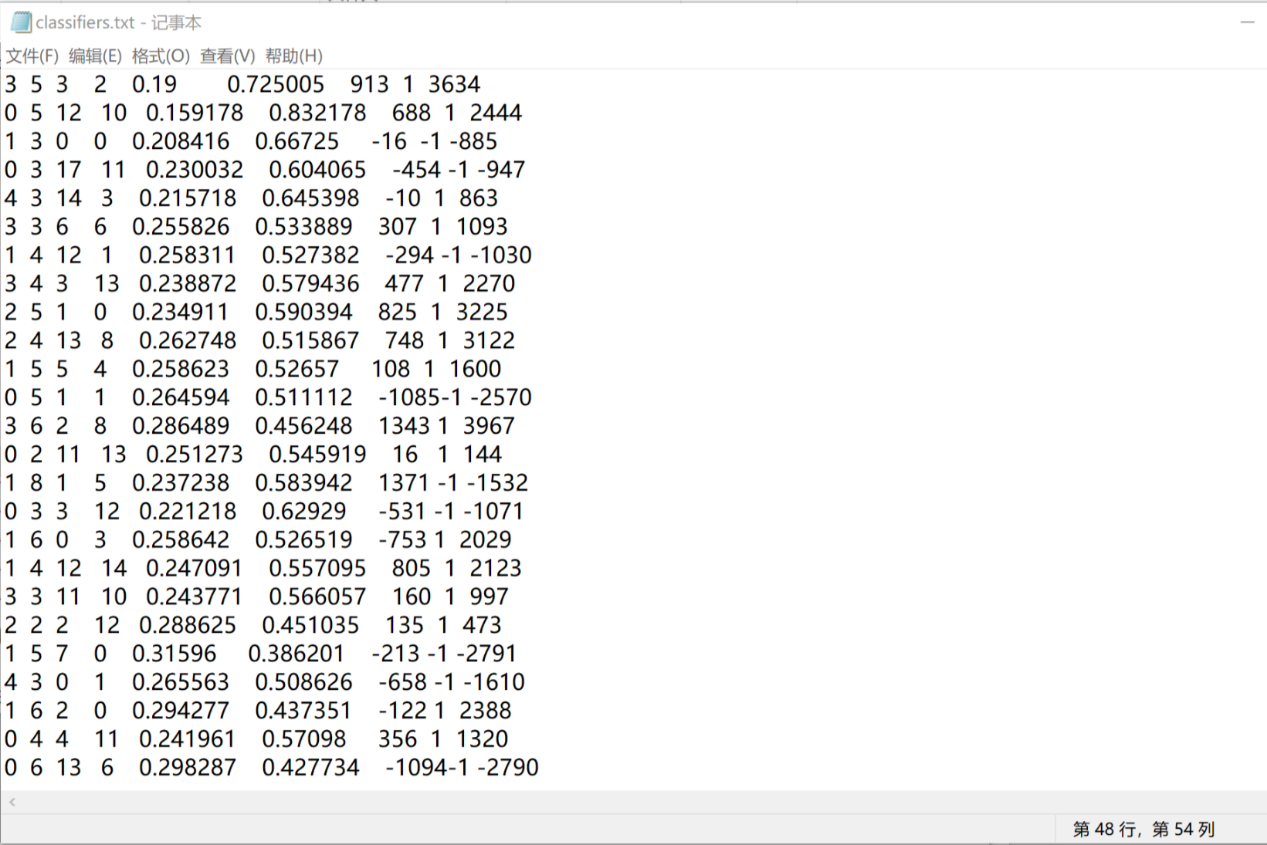
而在下一次迭代开始之前，要进行一次权重的调控，具体的原则就是，这次没分对的下一次要着重针对，于是，这一次被分错的样本的权重要提升，被分对的样本的权重要降低，具体操作如下



这样，经过k次迭代，我们就得到了k个弱分类器组成的强分类器，这也是我们训练版本的主要任务。得到一个较可靠的强分类器。

使用版本说明：

在进行过训练后，训练版本会将得到的弱分类器全部存储在名为classifier.txt的文件中，使用版本会从中读取所有的弱分类器，classifier文件内容大致如下：



从左到右依次是

模板类型/放大倍数/左上角的X坐标/左上角的Y坐标/错误率/权重/阈值/不等号方向/所有人脸样本中特征值最大样本的特征值

读取了这些弱分类器后，会对它们的权重进行归一化。

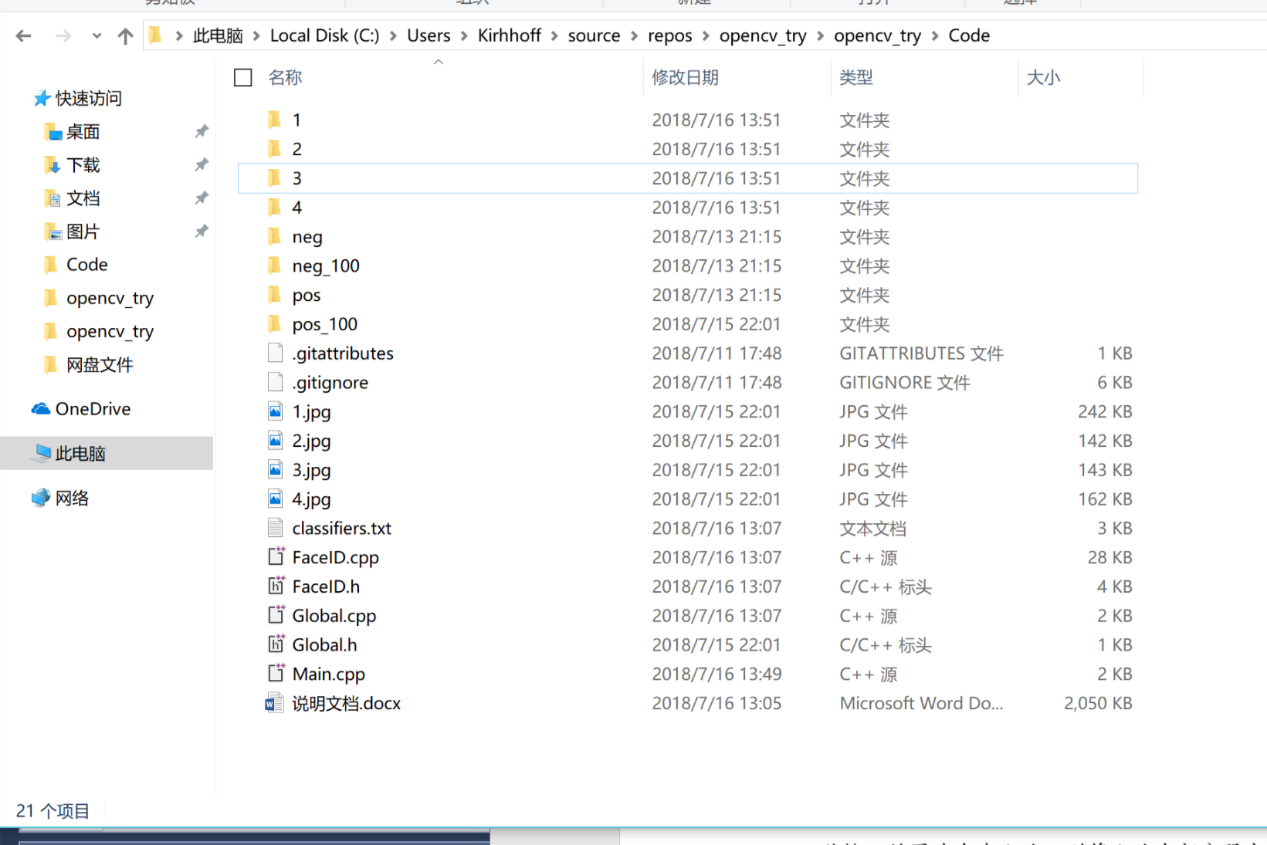
输入一张图片后，会先计算它对每个弱分类器的特征值value

如果p\*value>p\*threshold（此时p控制不等号方向的作用就体现出来了）那么这个弱分类器判定图中有人脸，总概率就会加上这个弱分类器的权重，根据最终的总概率来判定是否有人脸。

值得一提的是，我们的程序在接受到输入图片时会将其转化为20\*20的灰度图，处理结束后再按压缩比率重新恢复图片，但不免总是会使原图的尺寸发生变化。

输出的图片会保存在Code目录下，可同时输入多张图片，按输入顺序会生成名为1,2,3……的文件夹，每个文件夹内是将判定为有人脸的分类器一一翻转后的图像，但是真正看起来合理的目前只有15~20%.

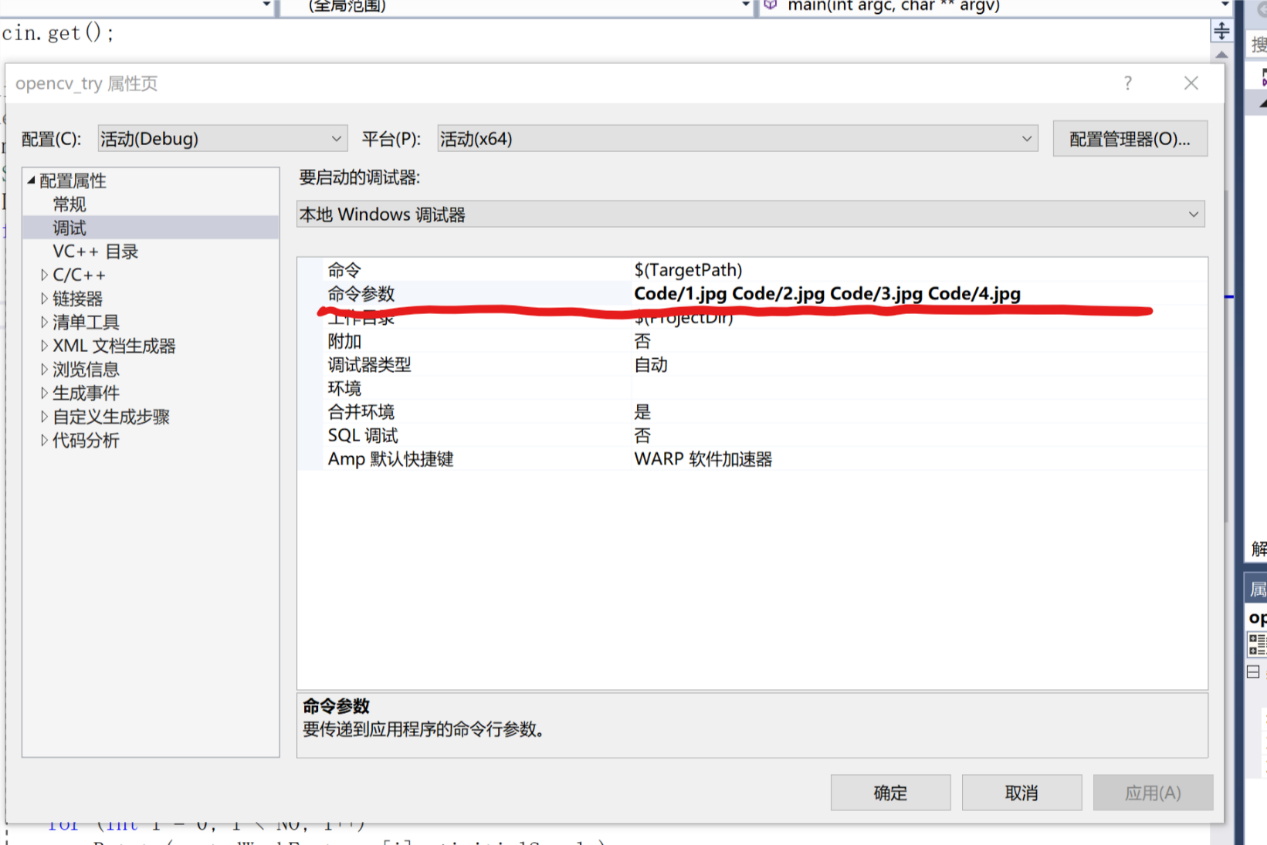
因为是在windows环境下的编程，所以并没有做出命令行，而且出于sdk版本的不匹配问题，无法直接将整个工程上传到github,github中只有我们程序的.cpp文件,.h文件。除此之外，还有我们的训练样本以及一些测试用的我的照片.整体看起来应该是这样：



pos\_100和neg\_100中是我们找的的素材图片，它们都是些100\*100的图片，（大部分是队员们归一化的功劳），我们把它们转化成20\*20的分别存储在pos和neg中，而那些1.jpg什么的是用来测试的我的照片，1，2，3，4文件夹就是对应的输出结果图片，当然，效果目前还并不是很好，但也可以看出来，有些特征是可以找到人脸的器官的。

使用说明：

由于SDK可能不匹配，我们并没有将整个项目添加到github，因此需要自己配置文件。应该先在vs上创建一个工程，然后将Code这个项目克隆到项目目录下，例如我在这里是创建了一个名为opencv\_try的项目，将Code克隆到了opencv\_try的目录下。然后进入工程，将这五个文件添加到工程中即可.而命令行参数要在vs的项目配置中给出例如：



给出图片相对工程的目录。生成的文件会在Code目录下，例如本例中的1,2,3,4文件夹。

**项目功能列表：**

1. 识别一张图片中是否有人脸（以训练样本探测则基本可以识别，但输入其他人脸暂时并不可行，主要是因为训练样本较少，过拟合较严重）
2. 若输入的图片中有人脸，则将人脸中部分器官反转并输出（正确抓取率目前只有约15%）

**代码说明：**

代码文件一共有五个，分别是FaceID.h,FaceID.cpp,Global.h,Global.cpp,Main.cpp

FaceID.h中主要是存储的结构体声明，宏以及函数原型

FaceID.cpp中主要是函数的定义

Global.cpp中存储了所有用到的全局变量

Global.h中定义了全局变量的extern

Main.cpp中是调试程序以及主程序入口

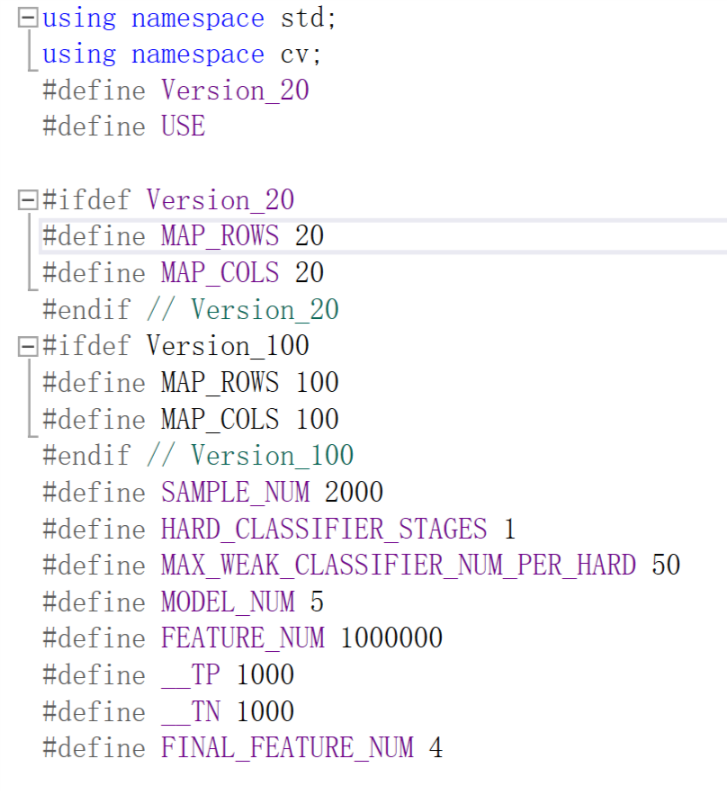
FaceID.h中的宏控制训练版本与使用版本,训练版本为TRAIN 使用版本为USE



定义了四个结构体

1. Feature是生成的特征
2. Sample
3. Key\_Value用于对样本遍历寻取阈值时的排序
4. ER\_Number用于对所有特征的错误率进行排序，选取最优弱分类器

对宏进行说明：



Version\_20是控制用于计算的图片大小的，我们采取的是20\*20

我们也曾尝试过直接使用100\*100的图片，但是有一些缺陷

1. 数量级问题。假如图片是n\*n的，而用于训练的样本有m个

那么训练这一过程的复杂大约是O(n^2\*mlog(m)),如果我们扩大到100\*100，则预计特征数会扩大两个数量级，实际上结果是约75万个，在这么多的特征中进行一次迭代的时间需要约80分钟，显然，要得到50个特征的时间太长了

1. 过拟合问题。如果仅仅是数量级，其实还可以接受，我们尝试进行过一次迭代，但是看到选出的特征我们瞬间意识到一个问题，就是过拟合。我们只有2000个样本，但是却有75万个特征，这就使得结果对数据的过拟合十分严重(实际上几乎无法使用，迭代的权重更新率为0)，因此我们也不能使用100\*100的图片

综合各方面考虑，Version\_100是不合适的，我们采用Version\_20

SAMPLE\_NUM是样本总数

\_\_TP是正样本数量

\_\_TN是负样本数量

HARD\_CLASSIFIER\_STAGES 是强分类器级数，我们本来打算使用级联的多个强分类器，但是由于两者效果差距不大，而运行时间也没有很长，我们最终就只使用了一级强分类器

MAX\_WEAK\_CLASSIFIER\_NUM\_PER\_HARD 可以理解为强分类器中的弱分类器数目

MODEL\_NUM 模型数量

FEATURE\_NUM 特征最大数量

FINAL\_FEATURE\_NUM 已没用

其他部分在代码中已经有详细的对应注释

**项目分工：**

彭天祥：整体逻辑编写，接口分配，算法资料查询，后期调试，文档说明

谢晓民：样本收集，样本大小转化，旋转处理，画框处理，opencv配置

申浩：积分图实现，图片数据存储类型设定，opencv配置

吴岚锋：素材的批处理，windows接口处理，图片读取

张力丹：图片存储格式查询，样本收集