**Haar Cascade**

Haar分类器 = Haar-like特征 + DP(积分图) + AdaBoost + 级联

具体来说：

1. 使用Haar-like特征做检测
2. DP/记忆化搜索 ⇒ 积分图(Integral Image)对Haar-like特征求值进行加速
3. AdaBoost训练区分人脸与非人脸的强分类器
4. 筛选式级联把强分类器级联到一起（决策树）

**一、Haar-like特征**

**1.1 矩阵特征**

四种模板：

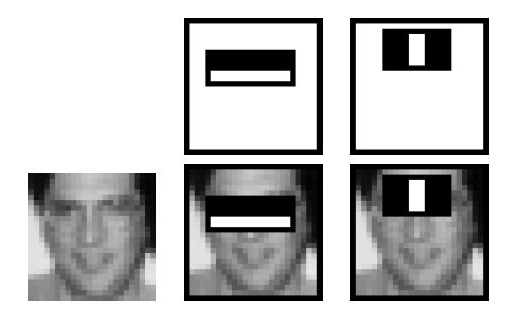
1. 边缘特征 A
2. 线性特征 B
3. 中心特征 C
4. 对角线特征 D



一个模板的特征值：白色矩形像素和减去黑色举行像素和。

注意对于A、B、D，都是∑白-∑黑，但是对于C来说，是∑白-2\*∑黑。因为C中的白色区域面积等于两倍的黑色区域面积，而其他的模型中的白色区域面积等于黑色区域面积

eg.（将上述四种模板应用到实际人脸图像上）

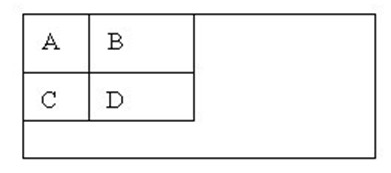


在一张图片（检测窗口）中，可能存在大量的子窗口的矩阵特征等待检测与计算，如何快速计算这么大批量的特征？

**1.2 积分图**

官方将积分图算法定性为了DP/记忆化搜索，其实也有点点像前缀和。一张图就可以直观说明了。

积分图试图将某一个矩形区域的特征之和保存起来，当需要计算一个子矩形区域的特征和时，只需要查询之前计算过的所有子区域的特征和，进行合理的加减即可。



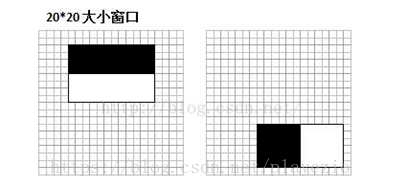
举例来说，假设D矩形区域的四个点（左上角，右下角，左下角，右上角）分别为α、β、γ和δ，假设ii(area)代表area区域的特征和，则

ii(D) = ii(α) + ii(β) - ii(γ) - ii(δ)

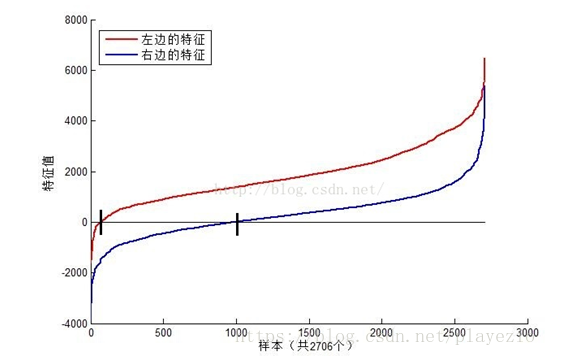
这个很显然了，第二个式子代表ii(A+B+C+D),所以我们要把A、B和C的全刨掉。但第三四项中各自都包括了一个A的特征和，那么就相当于多减了一次A，就要加回来。

**1.3 特征值的意义**

选取MIT人脸库中2706个大小为20\*20的人脸正样本图像(就是这个图像确实是张人脸），计算下图Haar特征。



左边对应人眼区域，右边没有具体意义



发现：

1. 左边特征值基本>0
2. 右边特征值均匀分布于人的两侧

则，左边对样本区分度大，右边对样本区分度小。

换句话说，计算特征后，计算机认为：

1. 对于所有样本来说，他们大都挺符合第一个模板的（包括模板的大小、位置，方向：黑在上还是白在上）
2. 对于所有样本来说，一部分有些趋向于不符合第二个模板，但一部分趋向于挺符合第二个模板

现在我们知道样本都是正样本（或者说所有样本在第一个模板的位置上确实都有一双眼睛）。因此计算机给出了一个很肯定的结果：这里有一双眼睛。但不确定在右下角上到底是什么：因为有些样本符合这个模板，有些不符合，这说明这个模板根本不是人脸的共性特征。

特征和越有区分度，越能说明某个区域很符合某个模板。

**二、级联分类器**

对AdaBoost了解较少，只能简单了解下了。

**2.1 分类器**

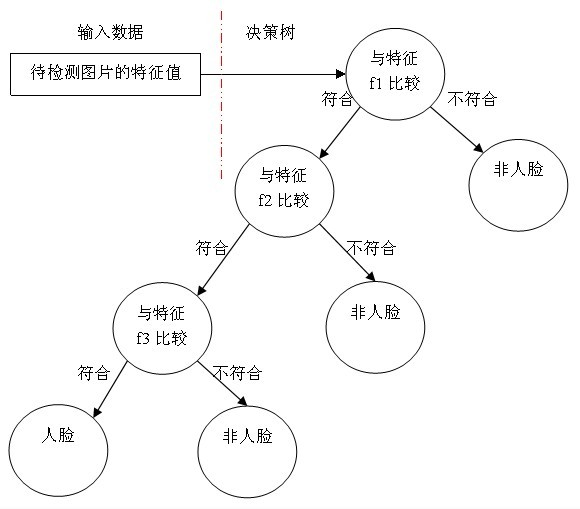
1. 强分类器：分类精度0.9以上
2. 弱分类器：分类准确率0.6-0.8（比随即预测好，但准确率也不太高）

西瓜书：弱学习器常指泛化性能略优于随即猜测的学习器（例如在二分类问题上精度略高于50%的分类器）

**2.2 级联分类器**

将弱分类器按一定的组合策略组合得到强分类器，然后级联起来，成为级联分类器。

个人理解：带短路的布尔表达式，但不够准确，因为决策树不仅仅解决二值型问题。



一个检测窗口需要通过所有的分类器才能被称为人脸。

**2.3 级联分类器检测**

训练级联分类器的目的就是为了检测的时候，更加准确，这涉及到Haar分类器的另一个体系，检测体系，检测体系是以现实中的一幅大图片作为输入，然后对图片中进行多区域，多尺度的检测，所谓多区域，是要对图片划分多块，对每个块进行检测，由于训练的时候用的照片一般都是20\*20左右的小图片，所以对于大的人脸，还需要进行多尺度的检测，多尺度检测机制一般有两种策略：

1. 一种是不改变搜索窗口的大小，而不断缩放图片，这种方法显然需要对每个缩放后的图片进行区域特征值的运算，效率不高；
2. 另一种方法，不断扩大搜索窗口，进行搜索，解决了第一种方法的弱势。

在区域放大的过程中会出现同一个人脸被多次检测，这需要进行区域的合并，这里不作探讨。

无论哪一种搜索方法，都会为输入图片输出大量的子窗口图像，这些子窗口图像经过筛选式级联分类器会不断地被每一个节点筛选，抛弃或通过。

**三、实例练习**

**3.1 作业目标与代码分析**

使用Haar+Cascade实现人脸检测（下图中的所有人脸）



目标：检测出图中的所有人脸，将检测出的人脸用红框标识出来。

我个人觉得，半张人脸其实是不算人脸的，即便Haar对我们来说是个黑盒，但根据上述算法，半张人脸的特征很有可能不足以判断该检测窗口是否是一张人脸，所以这里不像上一个作业一样将所有可能的检测目标都囊括进来，因为人脸和米粒的区别很大，后者只需要判断某一个检测窗口有没有实体，而前者还需要根据检测窗口的像素特征来判断这个实体是不是一张人脸。

所以修改目标：检测出图中所有的完整人脸，用红框对应标识出来。

首先装载训练好的模型：

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_alt2.xml")

这里我选择的是快速Haar分类器(alt2)，其实就是在几个能用的模型里面我挑了一个结果最好的模型。

转灰度图（其实转了和不转区别几乎没有，但我查了API说入参要是一张灰度图像，所以我就选择先转一次灰度图）

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow("Gray img", gray\_img)

调官方提供的人脸识别API，将结果(位置坐标：(x,y,w,h)，分别对应左上角x,y坐标以及窗口宽高）按行映射到矩阵中（n\*4)。

faces = faceCascade.detectMultiScale(gray\_img, 1.0185, 8)

1.0185指每次扫描搜索窗口的比例系数，在上文2.3的级联分类器检测中提到了，一张人脸的大小可能大于一个检测窗口，所以需要对一个区域进行多尺度的检测，有两种方式：

1. 放缩图片
2. 迭代增加搜索窗口大小

一般来说是第二种方便，所以官方API调用的就是第二种，因此我们要给一个扫描时搜索窗口的变化权重值。

8指成为合法目标的相邻矩阵的最小个数，这里直接用一张图来解释。



如果我们把这里的8改为4，那么将出现上图这样的结果，为什么呢？

一个红框代表一个最终的检测结果，也就是说分类器根据我们给定的参数进行计算，最终认为左边那个红框也是一张人脸。而我们给定的参数是minNeighbour，也就是刚才说的8/4。如果为4，那么左边红框代表的检测窗口中有不少于4个的搜索窗口的特征符合一张人脸，但该检测窗口中这样的搜索窗口个数不足8个。那么该参数为8时将不会有左边的红框。而右边的红框对应的合法搜索窗口个数大于等于8个，所以右侧红框将得到保留。

换句话说，minNeighbour越大，检测的标准越严格。

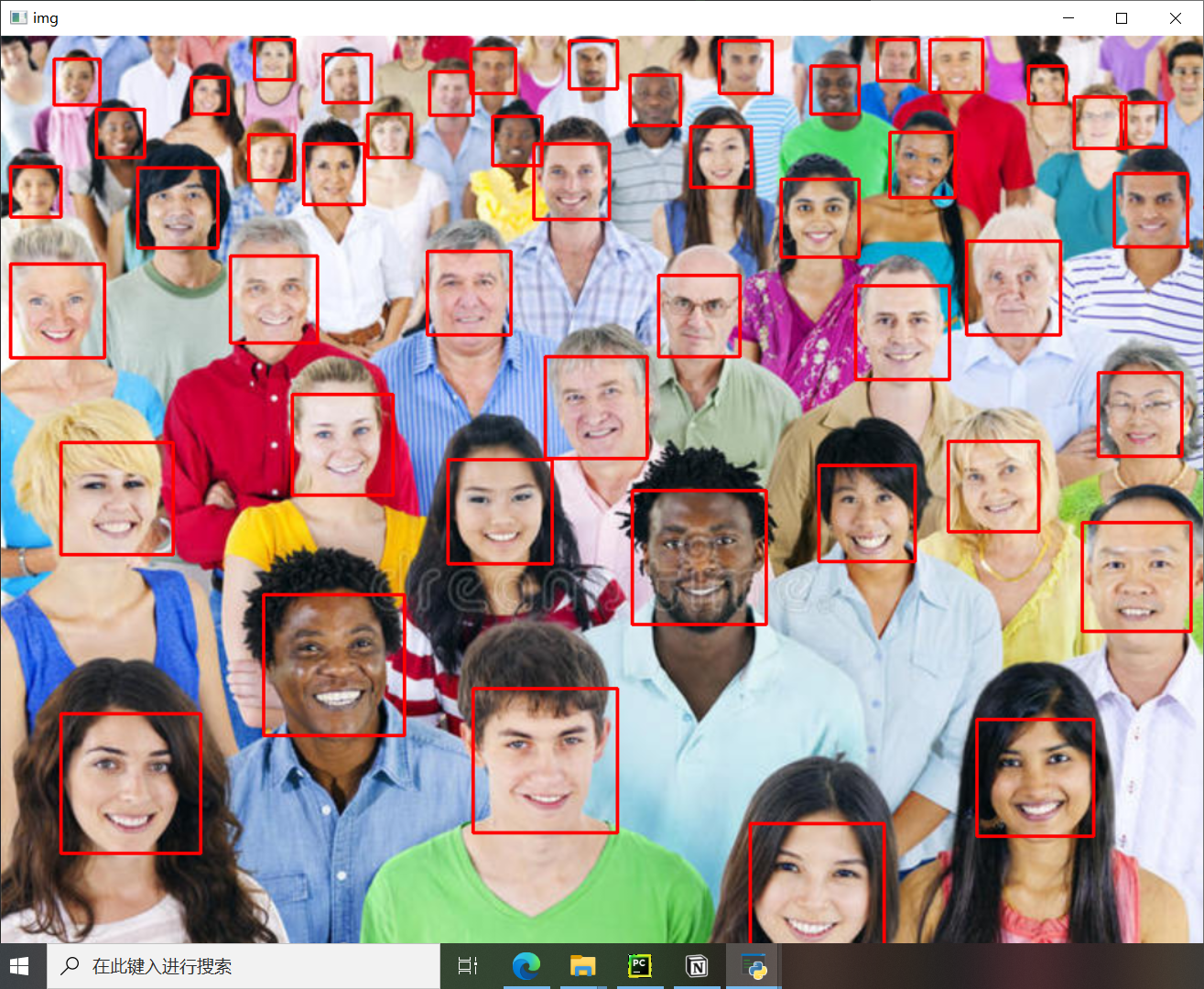
接下来，在途中绘制红框，标识人脸。

for (x, y, w, h) in faces:

# 将结果绘制到原图中

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

最终结果如下：



原图有点小，原图中标识的红框也比较小，为了便于查看，我使用resize函数将图片放大了。

完整代码如下：

import cv2

# 原图

img = cv2.imread("pic/faces2.jpg")

cv2.imshow("Original img", img)

img = cv2.resize(img, (0,0), fx=1.5, fy=1.5)

# 使用预训练摸型创建cascade分类器

# 默认Haar haarcascade\_frontalface\_default.xml

# 快速Haar haarcascade\_frontalface\_alt2.xml

# 快速LBP lbpcascade\_frontalface.xml

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_alt2.xml")

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow("Gray img", gray\_img)

ux,uy,uw,uh = 0,0,gray\_img.shape[0]//5,gray\_img.shape[1];

u\_area = gray\_img[ux:ux+uw,uy:uy+uh]

u\_faces = faceCascade.detectMultiScale(u\_area,1.01,6);

# 识别，将结果存储到faces变量中

faces = faceCascade.detectMultiScale(gray\_img, 1.0185, 8)

# 前后两次相继的扫描中搜索窗口的比例系数，默认为1.1 即每次搜索窗口扩大10%

# 构成检测目标的相邻矩形的最小个数

for (x, y, w, h) in faces:

# 将结果绘制到原图中

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.imshow("img", img)

cv2.waitKey(0)

**3.2 两个问题**

第一个问题是水印的问题。原图中有两处水印，可能会干扰到检测。但后来我在调参过程中发现，水印在参数合理的情况下不会被误检测为人脸，或者说，误检测根本不是水印导致的。比如上文中我为了举例说明minNeighbour的作用，截出来的误检测图片，那上面根本就没有水印。这说明误检测时时都有可能发生，并不是只要有水印就会出现误检测的。

同时，水印的特征也几乎不符合人脸的特征，所以水印本身也不会被当成人脸。当然最好是通过预处理把水印去掉，我有过这方面的尝试，但效果很差。（去水印的部分的人脸部分都被抹去了，无法通过插值法重新恢复，换句话说就是水印没了但人脸也没了）鉴于上述分析，我决定保留水印。

第二个问题，是检测不够精确。在结果图中，有一张几乎很完整的人脸没有被检测到。



右上角的人脸没检测出来。

我一开始猜测是整张图片中，由于近大远小的视觉规则，前面的人脸大，后面的人脸小，所以可能对于同一张图片的不同部分，同一组参数可能无法完全适配。于是我想再使用一次上一次作业中使用到的分治法，对于后面的图片，每次搜索窗口大小增加的小一点，相邻矩形数的要求再降低一点，但无论怎么调参，都无法检测到右上角这张人脸。



我对整张图片的上1/5部分进行局部处理，最好的一次结果是scale=1.01,minNeighbour=6，但即便是最好的一次，右上角那张人脸还是没检测出来。

后来我仔细看了下，右上角这张人脸，其实是不完整的。他的一部分不在图片内。

鉴于Haar级联分类器对我们来说其实是个黑盒，所以我们只能从流程和参数层面进行优化，没有办法对算法本身进行优化，因此我认为很有可能因为这不是一张完整的人脸，所以才检测不到。

## 心得与体会

本次作业我们使用Haar与级联分类器算法进行人脸识别。过程中需要选择恰当的模型（Haar,快速Haar，快速LBP等），传入恰当的参数进行检测。为了了解我调用的API的大致原理，我查阅了一些博客，根据实例整理了Haar级联分类器进行级联分类的笔记。对特征模板、检测窗口、积分图等基本概念有了大概的了解。但对AdaBoost算法的了解还不够多，对数学公式的推导掌握的也一般。这些地方需要加强。

实例练习中，我通过调整模型与参数，给出了一个大致完整且准确的人脸目标检测结果，不足之处在于右上角边缘处有一张大部分完整的人脸没有检测到，但由于算法本身是黑盒，因此我在尝试分治法进行局部化处理失败后，很难从算法层面进行调整优化，最终我认为该人脸并非一个完整的检测目标，无法检测到是因为缺少了一部分特征。