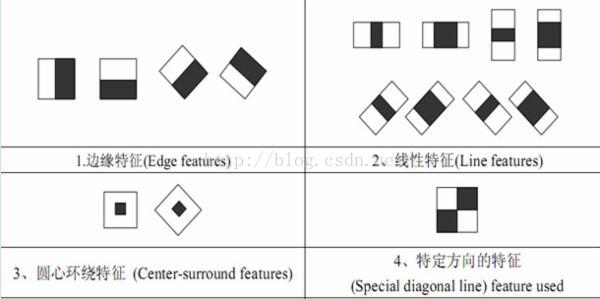
**一、AdaBoost算法原理**  
   AdaBoost算法是一种迭代的算法，对于一组训练集，通过改变其中每个样本的分布概率，而得到不同的训练集Si，对于每一个Si进行训练从而得到一个弱分类器Hi，再将这些若分类器根据不同的权值组合起来，就得到了强分类器。  
　第一次的时候，每个样本都是均匀分布，通过训练得到分类器H0，在该训练集中，分类正确的，就降低其分布概率；分类错误的，就提高其分布概率，这样得到的新的训练集S1就主要是针对不太好分类的样本了。再使用S1进行训练，得到分类器H1，依次迭代下去……，设迭代此外为T，则得到T个分类器。  
　　对于每个分类器的权值，其分类准确性越高，权值越高。  
  
  
**二、Haar特征**  
　**2.1 特征模板**  
　　就是一些矩形特征的模板，在viola&Jones的论文中，有下面这四种：



在opencv中的方法中，采用的是haar-like特征，有下面这14种，



  haar特征分成了四类：边缘特征、线性特征、中心特征和对角线特征，这些特征组成了特征模板。这些模板的特征值就是白色矩形区域像素值之和减去黑色矩形像素和。haar特征值反映了图像的灰度变化情况。

2.2 特征的个数  
　　对于一个给定的24X24的窗口，根据不同的位置，以及不同的缩放，可以产生超过160,000个特征。  
  
2.3 特征计算方法——积分图  
　　主要思想：有点类似于动态规划的思想，一次计算，多次使用。

**积分图构建算法：**

     1）用s(i,j)表示行方向的累加和，初始化s(i,-1)=0;

     2）用ii(i,j)表示一个积分图像，初始化ii(-1,i)=0；

     3）逐行扫描图像，递归计算每个像素(i,j)行方向的累加和s(i,j)和积分图像ii(i,j)的值。

        s(i,j)=s(i,j-1)+f(i,j)

        ii(i,j)=ii([i-1](https://www.baidu.com/s?wd=i-1&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd),j)+s(i,j)

     4）扫描图像一遍，当到达图像右下角像素时，积分图像ii就构造好了。

**三、选取弱分类器**  
　　一个弱分类器，实际上就是在这160,000+的特征中选取一个特征，用这个特征能够区分出人脸or非人脸，且错误率最低。  
  
　　现在有人脸样本2000张，非人脸样本4000张，这些样本都经过了归一化，大小都是24X24的图像。那么，对于160,000+中的任一特征fi，我们计算该特征在这2000人脸样本、4000非人脸样本上的值，这样就得到6000个特征值。将这些特征值排序，然后选取一个最佳的特征值，在该特征值下，对于特征fi来说，样本的加权错误率最低。选择160,000+个特征中，错误率最低的特征，用来判断人脸，这就是一个弱分类器，同时用此分类器对样本进行分类，并更新样本的权重。  
**具体过程如下：**  
  
　　1. 归一化权重：　　  
  
　　2、对于每一个特征f，训练一个弱分类器h;计算所以特征的加权错误率εf，　　　　　　　　  
  
　　3、选取具有最小错误率εf的弱分类器hi  
  
　　4、调整权重

**四、级联成强分类器**

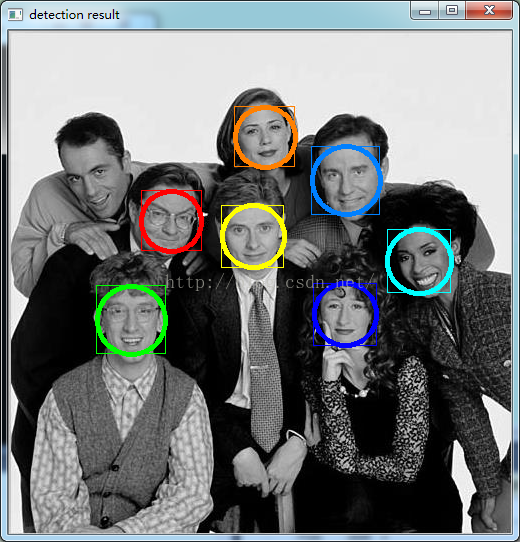
**五、检测**

检测过程中，通过不断的调整检测窗口的位置、比例，来找到人脸。

**六、OpenCV算法实现**

1. void detect\_and\_draw(IplImage\* img)
2. {
3. double scale = 1.2; //缩放因子
4. static CvScalar colors[] = {
5. { { 0, 0, 255 } }, { { 0, 128, 255 } }, { { 0, 255, 255 } }, { { 0, 255, 0 } },
6. { { 255, 128, 0 } }, { { 255, 255, 0 } }, { { 255, 0, 0 } }, { { 255, 0, 255 } }
7. };//Just some pretty colors to draw with
8. //Image Preparation
9. //
10. IplImage\* gray = cvCreateImage(cvSize(img->width, img->height), 8, 1);//构造灰度图像
11. IplImage\* small\_img = cvCreateImage(cvSize(cvRound(img->width / scale), cvRound(img->height / scale)), 8, 1);
12. cvCvtColor(img, gray, CV\_BGR2GRAY); //将待检测彩色图像转化成灰度图像
13. cvResize(gray, small\_img, CV\_INTER\_LINEAR);//缩放灰度图像到small\_img
14. cvEqualizeHist(small\_img, small\_img); //直方图均衡
15. //Detect objects if any
16. //
17. cvClearMemStorage(storage);
18. double t = (double)cvGetTickCount(); //opencv中测量运行时间的函数
19. CvSeq\* objects = cvHaarDetectObjects(small\_img, //检测人脸的核心函数
20. cascade,
21. storage,
22. 1.1,
23. 3,
24. 0/\*CV\_HAAR\_DO\_CANNY\_PRUNING\*/,
25. cvSize(30, 30),
26. cvSize(200,200));
27. t = (double)cvGetTickCount() - t;
28. printf("detection time = %gms\n", t / ((double)cvGetTickFrequency()\*1000.));
29. //Loop through found objects and draw boxes around them
30. for (int i = 0; i < (objects ? objects->total : 0); ++i) //画矩形
31. {
32. CvRect\* r = (CvRect\*)cvGetSeqElem(objects, i); //返回检测到的序列中元素指针，并将指针转换成序列中实际存储的元素的类型
33. cvRectangle(img, cvPoint(r->x\*scale, r->y\*scale), cvPoint((r->x + r->width)\*scale, (r->y + r->height)\*scale), colors[i % 8]);
34. }
35. for (int i = 0; i < (objects ? objects->total : 0); i++) //画圆
36. {
37. CvRect\* r = (CvRect\*)cvGetSeqElem(objects, i);
38. CvPoint center;//圆心
39. int radius;//半径
40. center.x = cvRound((r->x + r->width\*0.5)\*scale);
41. center.y = cvRound((r->y + r->height\*0.5)\*scale);
42. radius = cvRound((r->width + r->height)\*0.25\*scale);
43. cvCircle(img, center, radius, colors[i % 8], 3, 8, 0);
44. }
45. cvShowImage("result", img);
46. cvReleaseImage(&gray);
47. cvReleaseImage(&small\_img);
48. }

实验效果：

  
关键函数解析：

人脸检测的关键函数：

**CvHaarClassifier（Cascade\* cascade,  CvMemStorage\* storage,    double scale\_factor=1.1,  int min\_neighbors=3, int flags=0,  CvSize min\_size=cvSize(0,0) );**  
image 被检图像   
cascade harr 分类器级联的内部标识形式   
storage 用来存储检测到的一序列候选目标矩形框的内存区域。   
scale\_factor 在前后两次相继的扫描中，搜索窗口的比例系数。例如1.1指将搜索窗口依次扩大10%。   
min\_neighbors 构成检测目标的相邻矩形的最小个数(缺省－1)。如果组成检测目标的小矩形的个数和小于 min\_neighbors-1 都会被排除。

      如果min\_neighbors 为 0, 则函数不做任何操作就返回所有的被检候选矩形框，这种设定值一般用在用户自定义对检测结果的组合程序上。  
  flags 操作方式。当前唯一可以定义的操作方式是 CV\_HAAR\_DO\_CANNY\_PRUNING。如果被设定，函数利用Canny边缘检测器来排除一些边缘很少或者很多的图像区域，因为这样的区域一般不含被检目标。人脸检测中通过设定阈值使用了这种方法，并因此提高了检测速度。  
        min\_size 检测窗口的最小尺寸。缺省的情况下被设为分类器训练时采用的样本尺寸(人脸检测中缺省大小是~20×20)。

        函数 cvHaarDetectObjects 使用针对某目标物体训练的级联分类器在图像中找到包含目标物体的矩形区域，并且将这些区域作为一序列的矩形框返回。函数以不同比例大小的扫描窗口对图像进行几次搜索(察看cvSetImagesForHaarClassifierCascade)。 每次都要对图像中的这些重叠区域利用cvRunHaarClassifierCascade进行检测。 有时候也会利用某些继承（heuristics）技术以减少分析的候选区域，例如利用 Canny 裁减 （prunning）方法。 函数在处理和收集到候选的方框(全部通过级联分类器各层的区域)之后，接着对这些区域进行组合并且返回一系列各个足够大的组合中的平均矩形。

调节程序中的缺省参数(scale\_factor=1.1, min\_neighbors=3, flags=0)用于对目标进行更精确同时也是耗时较长的进一步检测。

为了能对视频图像进行更快的实时检测，参数设置通常是：

scale\_factor=1.2, min\_neighbors=2, flags=CV\_HAAR\_DO\_CANNY\_PRUNING, min\_size=<minimum possible face size>