

第 5 章 opencv 实时人脸识别应用开发.....	2
5.1 人脸检测.....	2
5.1.1 equalizeHist().....	2
5.1.2 级联分类器 CascadeClassifier.....	2
5.1.3 rectangle().....	3
5.1.3 VideoCapture 类.....	3
5.2 人脸识别.....	4
5.2.1 EigenFace 介绍.....	4
5.2.2 人脸识别算法对比.....	6
5.2.3 opencv 人脸识别案例分析.....	6
5.3 opencv 非特定目标检测之几何图形识别.....	8
5.3.1 准备训练样本素材.....	8
5.3.2 将图片进行灰度处理.....	8
5.3.3 生成 pos.txt 积极图片描述文件.....	9
5.3.4 生成 neg.txt 消极图片描述文件.....	9
5.3.5 生成 vec 文件.....	9
5.3.6 开始样本训练.....	9
5.3.6 对几何图进行识别.....	10

## 第 5 章 opencv 实时人脸识别应用开发

### 5.1 人脸检测

#### 5.1.1 equalizeHist()

```
void cvEqualizeHist( const CvArr* src, CvArr* dst );
```

功能：直方图均衡化处理

参数：

src：单通道灰度图

dst：得到对比度更强的输出图像

返回值：无

#### 5.1.2 级联分类器 CascadeClassifier

分类器是判别某个事物是否属于某种分类的器件，其结果要么是，要么不是，级联分类器，可以理解为将 N 个单类的分类器串联起来，如果一个事物能属于这一系列串联起来的所有分类器，则最终结果就成立，若有一项不符，则判定为不成立。比如人脸，它有很多属性，我们将每个属性做成一个分类器，如果一个模型符合了我们定义的人脸的所有属性，则我们人为这个模型就是一个人脸，比如人脸需要要有两条眉毛，两只眼睛，一个鼻子，一张嘴，一个大概 U 形状的下巴或者是轮廓等等。

常用方法：

一、从文件中加载级联分类器

```
CV_WRAP bool load( const String& filename );
```

二、检测级联分类器是否被加载

```
CV_WRAP bool load( const String& filename );
```

三、目标检测方法

```
CV_WRAP void detectMultiScale( InputArray image,
                               CV_OUT std::vector<Rect>& objects,
                               double scaleFactor = 1.1,
                               int minNeighbors = 3, int flags = 0,
                               Size minSize = Size(),
```

**做真实的自己，用良心做教育**

```
Size maxSize = Size() );
```

**功能：**检测输入图像中不同大小的对象。检测到的对象以矩形列表的形式保存

**参数：**

**image：**包含检测对象的图像的 CV\_8U 类型矩阵

**objects：**矩形的向量，其中每个矩形包含被检测的对象，矩形可以部分位于原始图像之外

**scaleFactor：**指定在每个图像缩放时的缩放比例

**minNeighbors：**指定每个候选矩形需要保留多少个相邻矩形

**flags：**含义与函数 cvHaarDetectObjects 中的旧级联相同，一般是 0，它不用于新的级联

**minSize：**对象最小大小，小于该值的对象被忽略

**maxSize：**最大可能的对象大小，大于这个值的对象被忽略

**返回值：**无

### 5.1.3 rectangle()

```
void rectangle(Mat& img, Point pt1, Point pt2,  
               const Scalar& color, int thickness=1,  
               int lineType=8, int shift=0)
```

**功能：**绘制简单、指定粗细或者带填充的矩形

**参数：**

**Img：**图像来源

**pt1：**矩形的一个顶点，一般是左上角

**pt2：**矩形对角线上的另一个顶点，一般是右下角

**color：**线条颜色 (BGR) 或亮度 (灰度图像) (grayscale image)

**Thickness：**组成矩形的线条的粗细程度，取负值时 (如 CV\_FILLED) 函数绘制填充了色彩的矩形

**line\_type：**线条的类型

**shift：**坐标点的小数点位数

**返回值：**无

### 5.1.3 VideoCapture 类

VideoCapture 既支持从视频文件读取，也支持直接从摄像机等监控器中读取，还可以读取 IP 视频流，要想获取视频需要先创建一个 VideoCapture 对象，对象创建以后，OpenCV 将会打开文件并做

**做真实的自己，用良心做教育**

好读取准备，如果打开成功，我们将可以开始读取视频的帧。

捕获本地视频：

```
VideoCapture cap( "video.mp4" );
```

捕获摄像机等监控器：

```
VideoCapture cap(-1);
```

捕获 IP 视频流：

```
VideoCapture cap( "http://youku.elecfans.com/video.flv" );
```

判断打开是否成功：

```
if (!cap.isOpened()) {  
    cout << "打开失败!!" << endl;  
    return -1;  
}
```

读取一帧图像：

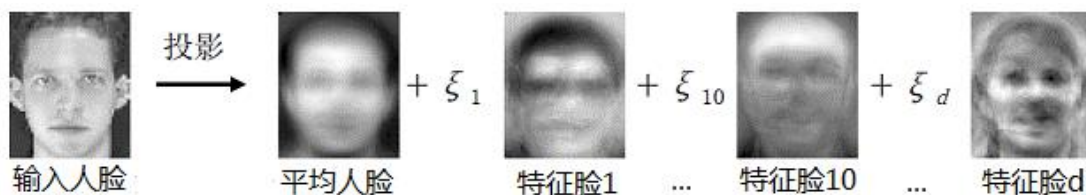
```
if (!cap.read(img)){  
    cap.set(CV_CAP_PROP_POS_FRAMES, 0);  
    continue;  
}
```

## 5.2 人脸识别

### 5.2.1 EigenFace 介绍

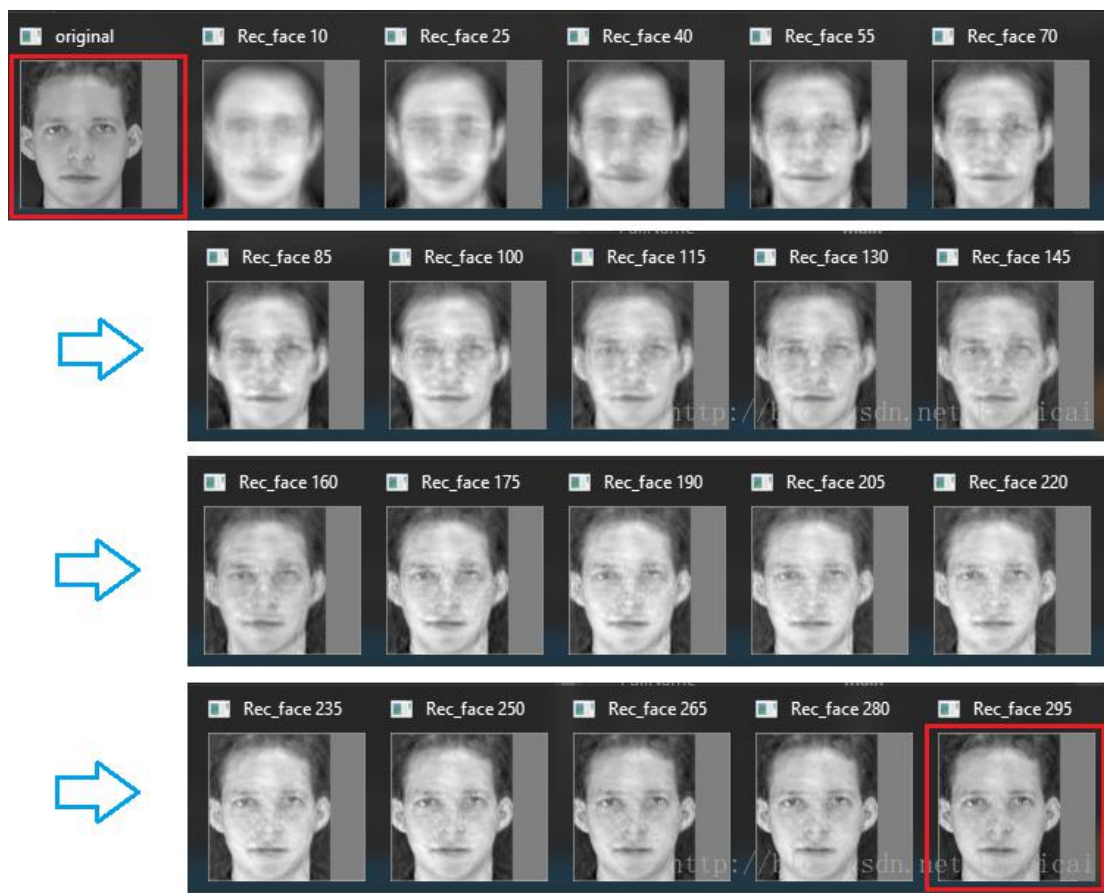
EigenFace 在人脸识别历史上应该是具有里程碑式意义的，其被认为是第一种有效的人脸识别算法。1987 年 Sirovich and Kirby 为了减少人脸图像的表达采用了 PCA（主成分分析）的方法进行降维，1991 年 Matthew Turk 和 Alex Pentland 首次将 PCA 应用于人脸识别，即将原始图像投影到特征空间，得到一系列降维图像，取其主成份表示人脸，因其主成份中就包含人脸的形状，估称 EigenFace 为“特征脸”。

EigenFace 是一种基于统计特征的方法，将人脸图像视为随机向量，并用统计方法辨别不同人脸特征模式。EigenFace 的基本思想是，从统计的观点，寻找人脸图像分布的基本元素，即人脸图像样本集协方差矩阵的特征向量，以此近似的表征人脸图像。具体过程如下：

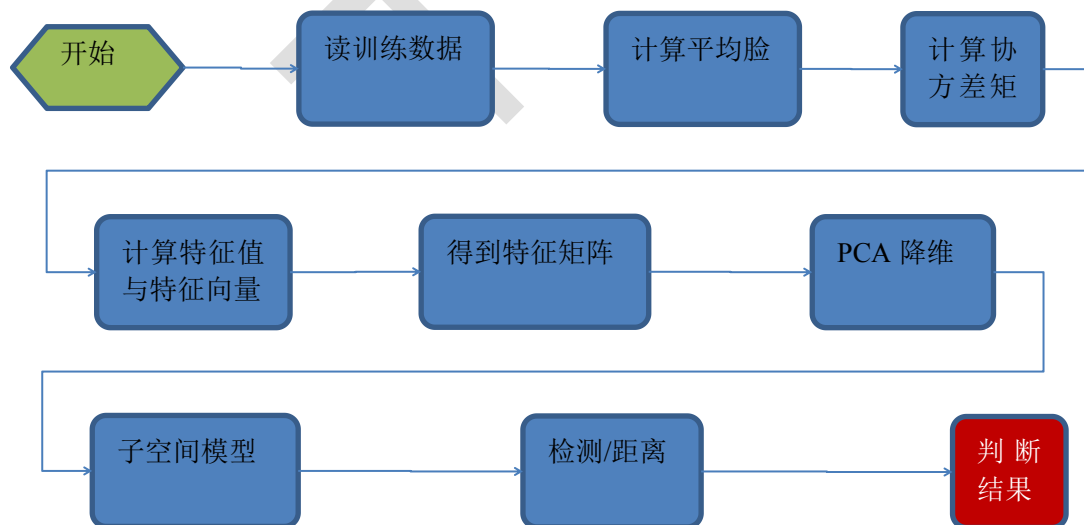


下面是从第一幅原始图像，原图和每隔 15 个特征向量进行重建后的效果图，很明显随着引入进

来的特征向量越来越多，重建后的效果也越来越接近原图：



EigenFace 的工作流程如下：



做真实的自己，用良心做教育

## 5.2.2 人脸识别算法对比

前一节主要介绍了 eigenFace，下面再简单介绍两种其它的识别算法：

### 一、FisherFace

FisherFace 是一种基于 LDA(全称 Linear Discriminant Analysis, 线性判别分析)的人脸识别算法，而 LDA 是 Ronald Fisher 于 1993 年提出来的，而 eigenFace 是基于 PCA 实现的，LDA 和 PCA 相同的地方是，都有利用特征值排序找到主成份的过程，但是不同的是 PCA 求的是协方差矩阵的特征值，而 LDA 是求的是一个更为复杂的矩阵的特征值。其中需要注意的是在求均值时，和 PCA 也是有所不同的，LDA 对每个类别样本求均值，而 PCA 是对所有样本数据求均值，来得到平均脸。采用 Fisherface 方法对人脸进行识别对光照、人脸姿态的变化更不敏感，有助于提高识别效果，但左右偏转脑袋，尽量不要超过  $15^{\circ}$ ，并对上下偏转比较敏感。

### 二、LBPH

LBPH 是利用局部二值模式直方图的人脸识别算法，LBP 是典型的二值特征描述子，所以相比前面 EigenFace 和 FisherFace，更多的是整数计算，而整数计算的优势是可以通过各种逻辑操作来进行优化，因此效率较高。另外通常光照对图中的物件带来的影响是全局的，也就是说照片中的物体明暗程度，是往同一个方向改变的，可能是全部变亮或全部变暗，因此 LBP 特征对光照具有比较好的鲁棒性。

## 5.2.3 opencv 人脸识别案例分析

### 5.2.3.1 准备样本

一、建议每个样本人物图像至少 8 张，拍摄样本图像时，注意头部尽量不要上下左右摆头。



二、编写样本描述文件 image.txt, 主要描述样本图像路径和样本图像对应的标签，格式如下：



```
image/face_1.jpg;17
image/face_2.jpg;17
image/face_3.jpg;17
image/face_4.jpg;17
image/face_5.jpg;17
image/face_6.jpg;17
image/face_7.jpg;17
image/face_8.jpg;17
image/face_11.jpg;18
image/face_22.jpg;18
image/face_33.jpg;18
image/face_44.jpg;18
image/face_55.jpg;18
image/face_66.jpg;18
image/face_77.jpg;18
image/face_88.jpg;18
```

### 三、实现流程

加载样本图像

```
vector<Mat> images;
vector<int> labels;
images.push_back(imread(path, 0));
labels.push_back(num);
```

训练样本，并给样本打上标签

```
Ptr<BasicFaceRecognizer> model = createEigenFaceRecognizer();
model->train(images, labels);
```

识别比对目标图像，将找到的样本对应标签返回

```
int label = model->predict(obj_img);
```

识别结果：



还可以通过上面最后介绍的两种识别算法进行测试，对比识别效果：

```
Ptr<BasicFaceRecognizer> model = createFisherFaceRecognizer();
```

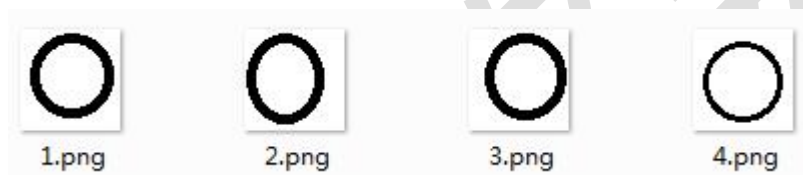
```
Ptr<LBPHFaceRecognizer> model = createLBPHFaceRecognizer();
```

## 5.3 opencv 非特定目标检测之几何图形识别

### 5.3.1 准备训练样本素材

#### 5.3.1.1 正样本

又叫积极样本，可以自己用画图板，多画几个图形统一放在 pos 目录，如下：



#### 5.3.1.2 负样本

又叫消极样本，你可自己从网上爬一些不相关的图片，也可以自己从电脑找一些，数量大概是正样本的 3 倍以上，尺寸是正样本的 8-12 倍，图片尺寸必须调整为统一大小，最后统一放在 neg 目录。

批量调整图片大小命令

```
find ./neg -name '*.jpg' -exec convert -resize 100X100! {} {} \;
```

### 5.3.2 将图片进行灰度处理

通过下面代码进行批量灰度处理：

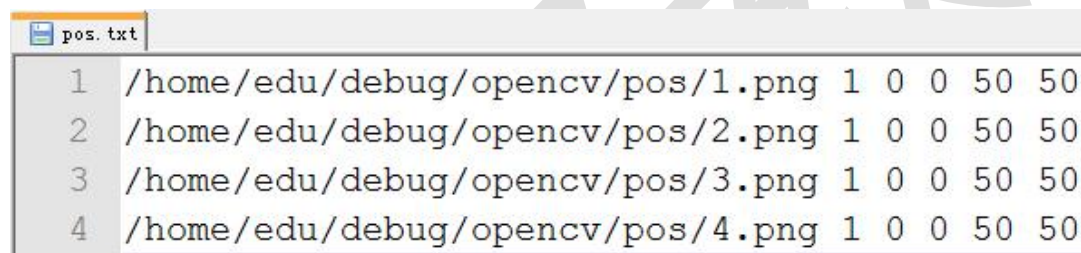
```
#include <fstream>
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
```

做真实的自己，用良心做教育



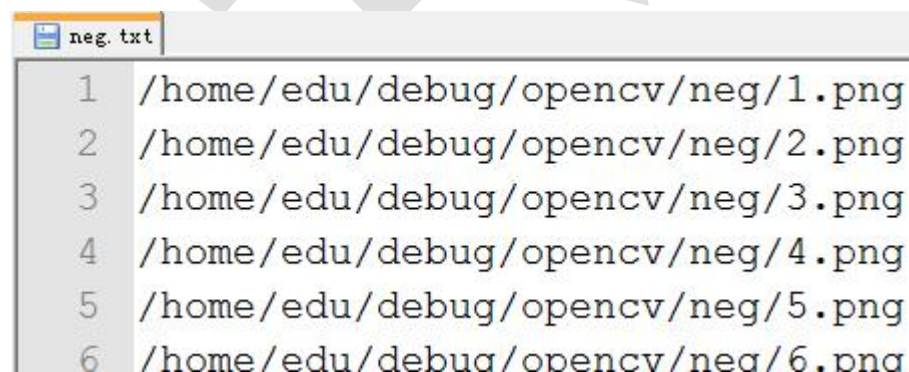
```
using namespace cv;
//g++ -o gray gray.cpp `pkg-config --cflags --libs opencv`
//./gray img 100
int main(int argc, char *argv[])
{
    char src [250];
    char obj [250];
    if(argc != 3){
        printf("./gray path 100(num)\n");
        return 0;
    }
    for(int i=1;i<atoi(argv[2])+1;i++){
        //将数字字母拼接在一起得到读取文件的路径
        sprintf(src, "%s/%d.jpg", argv[1], i);
        cout<<src<<endl;
        sprintf(obj, "%s/%d.jpg", argv[1], i);
        printf("%s\n", obj);
        //从指定路径 buffer 中读取图片
        Mat srcImage = imread(src), grayImage;
        cvtColor(srcImage, grayImage, CV_BGR2GRAY);
        imwrite(obj, grayImage);
    }
    cvWaitKey();
    return 0;
}
```

### 5.3.3 生成 pos.txt 积极图片描述文件



```
pos.txt
1 /home/edu/debug/opencv/pos/1.png 1 0 0 50 50
2 /home/edu/debug/opencv/pos/2.png 1 0 0 50 50
3 /home/edu/debug/opencv/pos/3.png 1 0 0 50 50
4 /home/edu/debug/opencv/pos/4.png 1 0 0 50 50
```

### 5.3.4 生成 neg.txt 消极图片描述文件



```
neg.txt
1 /home/edu/debug/opencv/neg/1.png
2 /home/edu/debug/opencv/neg/2.png
3 /home/edu/debug/opencv/neg/3.png
4 /home/edu/debug/opencv/neg/4.png
5 /home/edu/debug/opencv/neg/5.png
6 /home/edu/debug/opencv/neg/6.png
```

### 5.3.5 生成 vec 文件

```
opencv_createsamples -vec pos.vec -info pos.txt -num 4 -w 50 -h 50
```

### 5.3.6 开始样本训练

做真实的自己，用良心做教育

```
opencv_traincascade -data xml -vec pos.vec -bg neg.txt -numPos 4 -numNeg 30 -numStages 8  
-w 50 -h 50 -minHitRate 0.999 -maxFalseAlarmRate 0.2 -weightTrimRate 0.95 -featureType  
HAAR -mode ALL
```

上以过程因环境不同，需要反复修改参数来达到最终的训练效果，一般正常训练一个相对稳定的样本模型需要一两天时间。

### 5.3.6 对几何图进行识别

通过之前讲的人脸识别代码，采用刚刚训练好的 xml 分类器，加载一张几何图看看识别效果：

