

**《图像处理设计》**

**课程设计**

**题目： 基于Opencv的人脸识别系统**

数字媒体 学院 数字媒体技术 专业

学 号 1030514106

班 级 数媒1401

学生姓名 王晶晶

指导教师 狄 岚

二〇一七年六月

[1目的与要求 1](#_Toc486086580)

[1.1目的 1](#_Toc486086581)

[1.2任务及要求 1](#_Toc486086582)

[2设计内容 1](#_Toc486086583)

[2.1主要功能 1](#_Toc486086584)

[2.2原理方法 1](#_Toc486086585)

[2.2.1 Haar+Adaboost进行人脸检测 1](#_Toc486086586)

[2.2.1.1 Haar特征模板 1](#_Toc486086587)

[2.2.1.2 Haar特征计算 2](#_Toc486086588)

[2.2.1.3 Adaboost算法简介 2](#_Toc486086589)

[2.2.2 ASM人脸特征点识别算法 3](#_Toc486086590)

[2.2.3 LBPH（局部二值模式直方图）算法 4](#_Toc486086591)

[2.2.3.1 LBP编码 4](#_Toc486086592)

[2.2.3.2 LBPH算法 4](#_Toc486086593)

[3.总体方案设计 5](#_Toc486086594)

[3.1功能模块划分 5](#_Toc486086595)

[3.1.1 UI界面设计 5](#_Toc486086596)

[3.1.2 系统功能设计 5](#_Toc486086597)

[3.2系统流程图 5](#_Toc486086598)

[3.3系统环境简介及开发环境选择 5](#_Toc486086599)

[4功能模块 6](#_Toc486086600)

[4.1人脸检测 6](#_Toc486086601)

[4.2人脸特征点识别 7](#_Toc486086602)

[4.3人脸识别 7](#_Toc486086603)

[5总结与体会 10](#_Toc486086604)

[6参考文献 11](#_Toc486086605)

## 1目的与要求

### 1.1目的

**利用图像处理的相关知识实现一个人脸识别的系统。巩固加强图像处理相关知识结构体系。**

### 1.2任务及要求

自选开发工具设计一个人脸识别的系统，要求可以检测出现有图片中的人脸是人脸库中的那个人。

## 2设计内容

### 2.1主要功能

1. **实现图片以及摄像头视频的实时人脸识别**
2. **实现图片以及摄像头视频的实时人脸特征点显示**
3. **实现摄像头拍摄的图片是否存在于人脸库中**

### 2.2原理方法

#### 2.2.1 Haar+Adaboost进行人脸检测

Adaboost算法的速度与精准性与特征模板的选取有很大的关系[3]。在本实验过程中，作者选取的是Haar特征。由于使用到的工具opencv里面对Haar模板已经采集好了，所以在使用的时候只需要导入文件haarcascade\_frontalface\_alt2.xml即可。在这里简单介绍一下Haar特征模板[3]。

##### 2.2.1.1 Haar特征模板

|  |  |  |  |
| --- | --- | --- | --- |
| 左右型 |  | 上中下型 |  |
| 上下型 |  | 田字型 |  |
| 左中右型 |  |  |  |

以上是原始的Haar特征，后期Haar-like特征被提出，主要就行在原始特征的基础上依据边缘特征、线特征、中心环绕、对角线等特征对原始特征进行了扩展[2]。使用Haar特征的目的就是希望能够将人脸特征量化，以区分人脸和非人脸。

##### 2.2.1.2 Haar特征计算

Haar特征计算主要是通过对每个像素进行计算，得到图像的积分图。使用积分图，用相同的时间可以计算尺度大小不同的矩形特征值，以便提升检测的速度[4]。

主要原理如下：

对于图像内一点A(x,y)，定义起积分图ii(x,y)为[2]:



其中i(x’,y’)为点(x’,y’)处的像素值；

构建积分图的算法：

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ii(i,j)的值

s(i,j) = s(i,j-1)+f(i,j);

ii(i,j) = ii(i,j)+s(i,j);

④一直扫描值图像右下角的像素时，扫描结束

##### 2.2.1.3 Adaboost算法简介

Adaboost算法是一个分类算法，需要大量的图片进行训练，之后所需检测图片的准确率才能有所保证。在这里作者选用的opencv工具里面的xml文件就已经是经过大量图片的训练得出来的Haar特征了。下面简单的介绍一下Adaboost算法的流程及原理。Adaboost算法的前身就是机器学习里面的PCA（主成分分析）模型。在人脸检测过程中运用Adaboost算法主要就是为了能够形成一个分类器，该分类器能够完美的区分人脸部分以及非人脸部分。在这里就不得不说一下决策树了，Adaboost算法也可以说成是一个二叉决策树，整个过程可以分为分类与回归[1]。由于强分类器是由弱分类器发展而来，主要原理还是弱分类器的原理，所以在这里注意介绍一下弱分类器的算法。

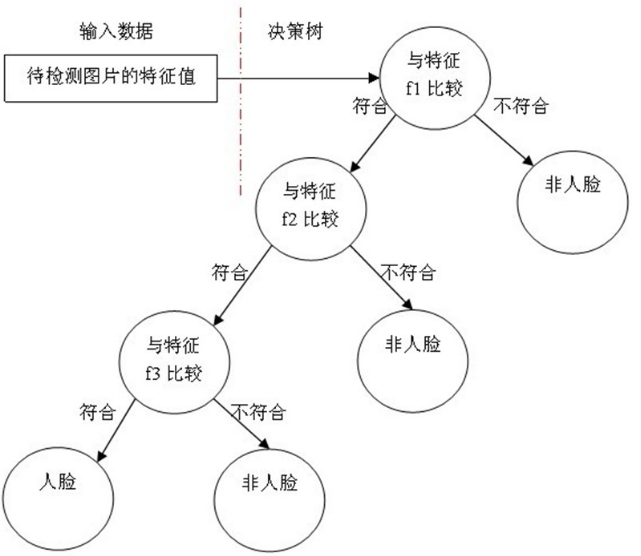
弱分类器的具体操作过程[1]：

①对于每个特征f，计算所以训练样本特征值，并将其排序

扫描以便排好序的特征值，对排好序的表中元素计算全部人脸样本权重和t1、全部非人脸样本的权重和t0,、在当前元素前的人脸样本权重和s1、在当前元素前的非人脸样本权重和s0

②求得每个元素的分类误差r = min((s1+(t0-s0)),(s0+(t1-s1)))

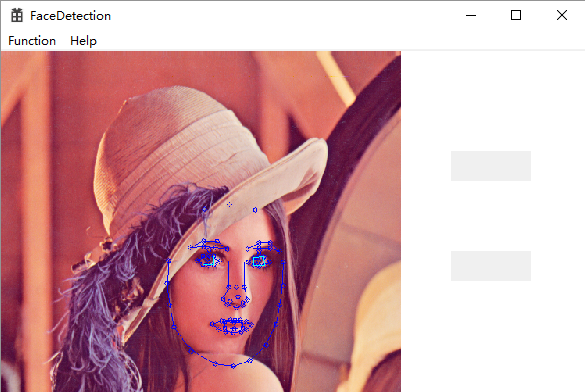
强分类器主要就是经过多轮迭代，将每轮得出的弱分类器的结果保存后，最终就行组成一个强分类器。



弱分类器进行分类的主要流程

#### 2.2.2 ASM人脸特征点识别算法

ASM（active shape model），主动形状模型。在本实验中，作者选择了使用68个关键特征点来标记人脸中的关键特征点，68个点分布情况如下。



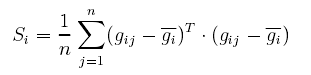
ASM的算法主要分成2个阶段：

1. 训练

首先得建立形状模型，搜集多的训练样本，训练样本的图片需要保证军事含有人脸的图片。对于训练的样本里面的人脸的特征点先进行手动的标记68个关键特征点，如上图。

然后依据标记出来的68个关键特征点构成训练集的形状向量。为了使这些点分布的更加合理有效，需要先进行Procrustes方法对点分布进行对齐操作[9]。

将对齐后的形状向量进行PCA处理，得出平均形状向量后计算所以样本与平均形状向量之间的协方差矩阵，然后对每个特征点进行定位[9]。



新的特征与训练好的特征之间的相似性为[8]：



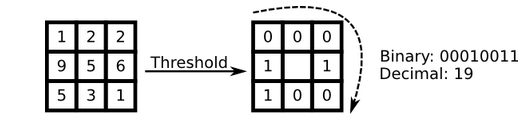
1. 搜索

通过样本集训练得到ASM模型后，主要通过调整各个权重使得当前特征点的位置与对应的新位置更加接近，档迭代的次数达到一定要求或者是达到指定的阈值时迭代结束，结束搜索的过程。

#### 2.2.3 LBPH（局部二值模式直方图）算法

##### 2.2.3.1 LBP编码

首先需要以每个像素为中心，判断该像素与其周围像素灰度大小关系。假设是定义在像素3\*3领域内，将相邻8个像素的灰度值与中心处的像素进行比较，如果周围像素大于中心像素，则该像素点的位置被标记为1，否则为0[5]。这样就能得到LBP的编码。



##### 2.2.3.2 LBPH算法

首先需要加载数据集，训练模型。对数据集里面的所以样本先进行LBP编码后求其直方图。直方图求解的公式[10]如下：



准备好模型之后，直接预测该人脸图片属于数据库图片里面的哪一类。在这里主要是通过欧氏距离来计算两个直方图之间的相似度[10]：



在预测的时候，主要思想就是找出数据库中与当前图片作为相似的分类，即当前图片与哪一类图片的相似度最高，那么模型就会显示当前图片属于哪一类。

由于LBP算法对于光照的鲁棒性较强，所以在使用的时候受光照的影响较小。

## 3.总体方案设计

### 3.1功能模块划分

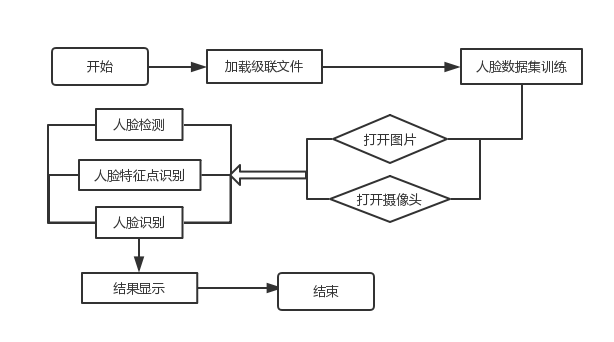
#### 3.1.1 UI界面设计

由于实验的要求是设计一个含有GUI的人脸识别的系统，这就要求对GUI有一定的设计。因为在实验中，选择的开发工具是VS2013，所以就直接选用了Windows应用程序来对整个实验的界面进行设计，界面的图片下文可见。

#### 系统功能设计

1. 打开文件功能
2. 对选取的照片进行人脸检测
3. 对摄像头内抓取的视频流中的人脸进行检测
4. 对选取的照片进行人脸特征点定位
5. 对摄像头内抓取的视频流中的人脸特征点进行定位
6. 对选取的照片进行人脸识别

### 3.2系统流程图



### 3.3系统环境简介及开发环境选择

系统运行的环境：VS2013+WIN10

选用的开发工具：Opencv + C++

## 4功能模块

### 4.1人脸检测

在这一部分主要使用Adaboost+Haar特征算法实现人脸检测的功能。主要使用了opencv里面封装好的Haar特征值来进行人脸的检测。

实现该功能的主要函数为FaceDetection.h里面的detectAndDisplay(Mat)函数

函数详情如下：

//级联分类器加载

string face\_cascade\_name = "data/haarcascade\_frontalface\_alt2.xml";

CascadeClassifier face\_cascade;

//加载联机分类器

void Init(){

if (!face\_cascade.load(face\_cascade\_name)){

return;

}

}

//框出人脸

void detectAndDisplay(Mat frame){

vector<cv::Rect> faces;

Mat frame\_gray;

cvtColor(frame, frame\_gray, CV\_BGR2GRAY);

equalizeHist(frame\_gray, frame\_gray);

face\_cascade.detectMultiScale(frame\_gray, faces, 1.1, 2, 0 | CV\_HAAR\_SCALE\_IMAGE, cv::Size(30, 30));

for (int i = 0; i < faces.size(); i++){

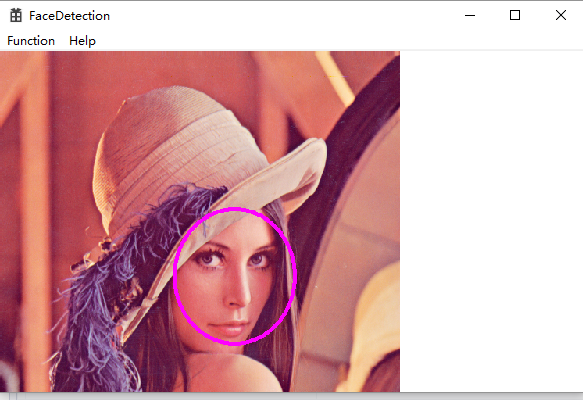
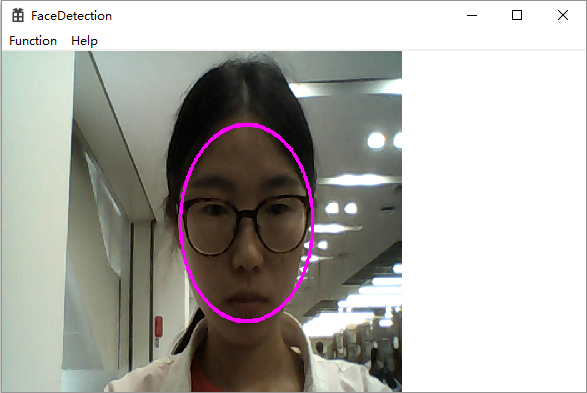
cv::Point center(faces[i].x + faces[i].width\*0.5, faces[i].y + faces[i].height\*0.5);

ellipse(frame, center, cv::Size(faces[i].width\*0.45, faces[i].height\*0.5), 0, 0, 360, cv::Scalar(255, 0, 255), 4, 8, 0);

//rectangle(frame, faces[i], Scalar(255, 0, 255), 4, 8, 0);

}

}

### 4.2人脸特征点识别

由于ASM的整个算法系统都特别麻烦，所以在编码的时候直接运用封装好的特征点标记，这里的特征点标记是实现通过程序对相关数据集进行手动的面部特征点采集所得。这一部分的代码主要封装在stasm\_lib.h和stasm\_landmrks.h文件里面由于篇幅问题这里没有贴出来。

使用上面提到的两个.h文件的功能核心代码在FaceDetection.h里面的detectFeatures(Mat)里面，这里主要贴出核心代码：

cvtColor(img2, img, CV\_BGR2GRAY, 0);

float landmarks[2 \* stasm\_NLANDMARKS] = { 0 }; // x,y coords (note the 2)

if (!stasm\_search\_single(&foundface, landmarks, (const char\*)img.data, img.cols, img.rows, path, "../data"))

{

//MessageBox(NULL, \_T("no face"), 0, MB\_OK);

return;

}

然后直接通过绘制特征点在图片上，主要方法就是遍历landmarks数组。

下面粘贴处绘制下巴的几个点的方式：

//下巴轮廓[0~15)共15个点;

int start = 0; int end = 15;

for (int i = start + 1; i <end; i++)

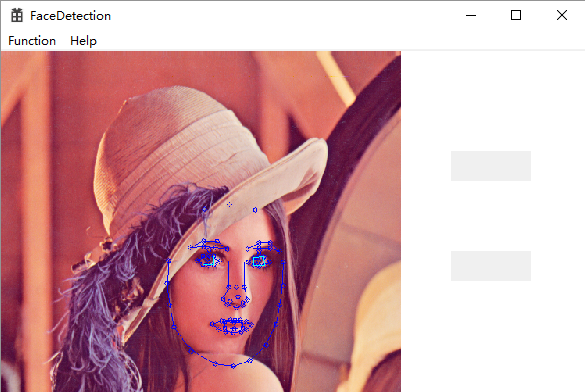
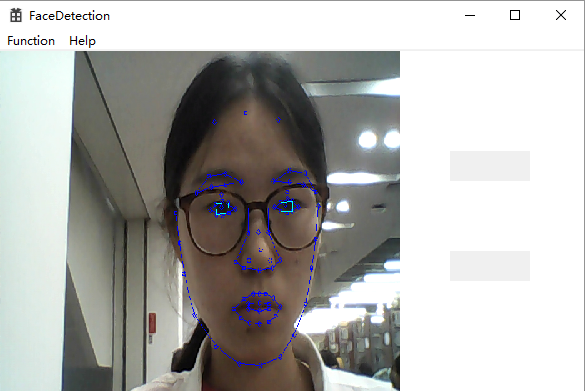
{

line(img2, Point(cvRound(landmarks[i \* 2]), cvRound(landmarks[i \* 2 + 1])), Point(cvRound(landmarks[i \* 2 - 2]),

cvRound(landmarks[i \* 2 - 1])), CV\_RGB(0, 0, 255), 1, 8, 0);

feature[i] = Point(cvRound(landmarks[i \* 2 - 2]), cvRound(landmarks[i \* 2 - 1]));

}

### 4.3人脸识别

依据上面的介绍可知，本文中对人脸识别采用的算法时LBPH算法，那么第一步就是实现LBP编码，在这里采用的是旋转一致LBP算子。

//UniformLBP

Mat UniformLBP(Mat img)

{

uchar UTable[256]; memset(UTable, 0, 256 \* sizeof(uchar));

uchar temp = 1;

for (int i = 0; i<256; i++)

{

if (getHopCount(i) <= 2)

{

UTable[i] = temp;

++temp;

}

}

Mat result;

result.create(img.rows - 2, img.cols - 2, img.type()); result.setTo(0);

for (int i = 1; i<img.rows - 1; i++)

{

for (int j = 1; j<img.cols - 1; j++)

{

uchar center = img.at<uchar>(i, j); uchar code = 0;

code |= (img.at<uchar>(i - 1, j - 1) >= center) << 7;

code |= (img.at<uchar>(i - 1, j) >= center) << 6;

code |= (img.at<uchar>(i - 1, j + 1) >= center) << 5;

code |= (img.at<uchar>(i, j + 1) >= center) << 4;

code |= (img.at<uchar>(i + 1, j + 1) >= center) << 3;

code |= (img.at<uchar>(i + 1, j) >= center) << 2;

code |= (img.at<uchar>(i + 1, j - 1) >= center) << 1;

code |= (img.at<uchar>(i, j - 1) >= center) << 0;

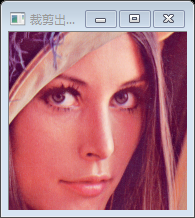
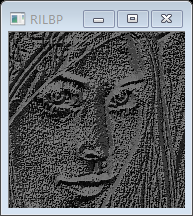
result.at<uchar>(i - 1, j - 1) = UTable[code];

}

}

return result;

}

在做实验的过程中，由于没有对算法提出什么新的改进的方法，以及时间上面的问题，所以在进行人脸识别的时候，主要就是运用opencv里面封装好的FaceRecongizer类来实现LBPH算法的训练以及预测的过程。

实现该功能的主要函数为FaceDetection.h里面的ReadPic、PicTrain、detectFace三个函数。

//redpic

void ReadPic(vector<string> filenames){

for (int i = 0; i < filenames.size(); i++){

Mat temp = imread(filenames[i]);

cvtColor(temp, temp, CV\_BGR2GRAY);

if (!temp.empty()){

images.push\_back(temp);

}

}

}

//识别表情类别

void PicTrain(){

string path = "F:\\att\_faces";

GetAllPic(path, filenames);

ReadPic(filenames);

//开始训练

Model = cv::face::createLBPHFaceRecognizer();

Model->train(images, labels);

}

//人脸识别

void detectFace(){

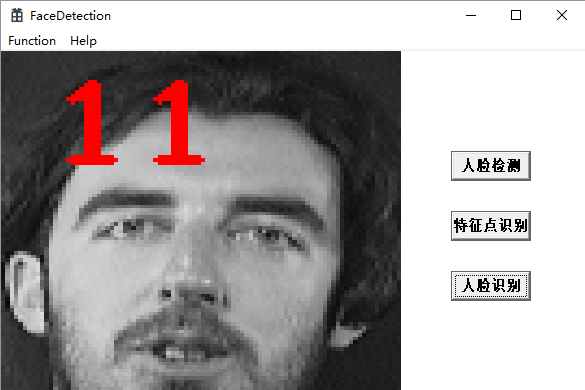
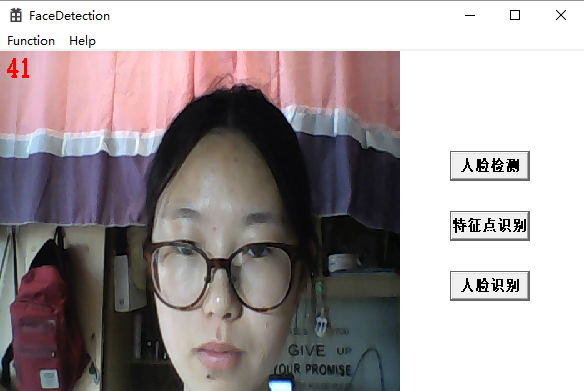
Mat im;

cvtColor(CurrentImg, im, CV\_BGR2GRAY);

int predit = Model->predict(im);

putText(CurrentImg, to\_string(predit), Point(10, 30), CV\_FONT\_HERSHEY\_COMPLEX, 1, Scalar(0,0,255),2);

}

## 5总结与体会

1. **对于人脸识别的一些步骤流程有了足够的了解，但是对于算法的细节了解的还是不是很清楚，尤其是有关ASM的算法没有理解透彻。**
2. **在实验的过程中，对人脸进行识别时，使用的是opencv里面封装好的contrib库，但是这个库用起来真的超级不容易，在配置的时候屡屡出错，版本的限制问题很严重。**
3. **在整个实验的过程中，由于选用的是C++的Windows应用程序，在实验的过程中，肯定会使用到windows.h这个头文件，但是opencv库与Windows.h中都有ACCESS\_MASK这个参数，导致了怎么运行程序，程序都不能通过。在这种百度的情况下，依旧没能解决问题，最后选择了修改opencv库函数里面的mat.hpp，将ACCESS\_MASK改成了\_FIX\_ACCESS\_MASK。最终才将问题解决了。**
4. **对于人脸识别的算法，经过本次课程设计对LBPH的相关算法，有了一定的了解。首先准备好人脸数据集，想找出该数据集下面的所有人脸的LBP特征，并记录每张图片的标签（相同标签代表同一个人）。然后经过训练的过程，让数据库认识这个人，最后再通过对当前图片的预测，在库里找出与这张人脸最相似的人脸。**
5. **在实验的过程中，大量使用了opencv相关的函数，但是由于在使用的过程不够细心以及知识量积累不够的问题，代码在调试的过程中屡屡出错，尤其是在使用FaceRecongizer类的时候，关于labels标签的使用出错，以及图片未进行灰度化进行直接运用等等一系列问题。通过本次实验不仅仅对人脸识别的相关算法有了一定的了解，同时还对opencv的使用更加得心应手，自己的编程能力以及算法的思考能力都有了突破和提高。**

## 6参考文献

[1] <http://blog.csdn.net/zouxy09/article/details/7922923>

[2] <http://blog.csdn.net/zouxy09/article/details/7929570/>

[3] 龙伶敏. 基于Adaboost的人脸检测方法及眼睛定位算法研究[D].电子科技大学,2008.

[4] 齐兴敏. 基于PCA的人脸识别技术的研究[D].武汉理工大学,2007.

[5] 王宪,张彦,慕鑫,张方生. 基于改进的LBP人脸识别算法[J]. 光电工程,2012,(07):109-114.

[6] 仲澄,冯涛. 复杂背景下的人脸定位识别方法[J]. 计算机工程与应用,2012,(01):205-207+211.

[7] 宋鹏飞. 基于LBP特征的人脸识别算法改进研究[D].东北大学,2014.

[8] 范玉华,马建伟. ASM及其改进的人脸面部特征定位算法[J]. 计算机辅助设计与图形学学报,2007,(11):1411-1415.

[9] <http://blog.csdn.net/carson2005/article/details/8194317>

[10] <http://blog.csdn.net/sinat_25885063/article/details/43704005>