**实验名称：**人脸检测与识别

**实验目的：**通过建立自己的人脸检测识别系统，深入理解人机交互设计。

**实验要求：**能够调用摄像头实时抓取图像，如果图像中出现人脸，能够检测定位人脸，同时保存人脸，建立自己的人脸样本集；训练人脸识别模型；测试人脸识别模型。

**实验环境与工具：**WIN 10系统，Microsoft VS 2012, Opencv 2.4.9

**实验步骤**

**一、数据收集**

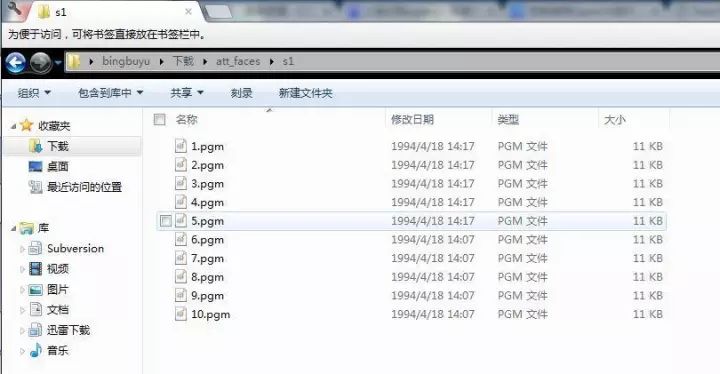
1、标准人脸数据集

本实验用的标准数据集是opencv给出的教程里面的第一个数据集：

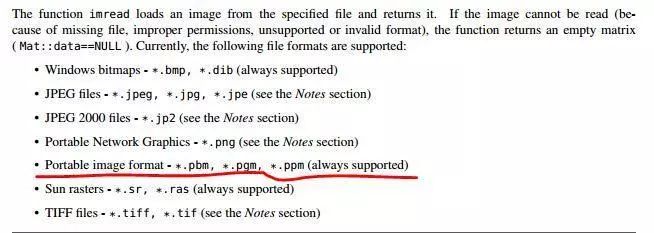
ORL Database of Faces（https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html）。又称ORL人脸数据库，40个人，每人10张照片。照片在不同时间、不同光照、不同表情(睁眼闭眼、笑或者不笑)、不同人脸细节(戴眼镜或者不戴眼镜)下采集。所有的图像都在一个黑暗均匀的背景下采集的，正面竖直人脸(有些有轻微旋转)。

下载下来之后是这样的：





**注：**每个人一个文件夹，每个文件夹下是这个人的十张照片，但是不是我们熟悉的BMP或者是PNG或者是JPEG格式的，而是PGM格式的。windows7自带的照片查看器和画图软件都不能打开这种格式的图片。不过opencv文档里有这样的描述：



也就是使用imread()可以读取PGM格式的图片。



**2、建立自己的人脸数据集**

**调取摄像头**



图1

样本收集界面设计（如图1）可以打开或者清空样本文件夹，还有就是进行人脸检测，点击**开始检测**会调用摄像头，捕获图像。

核心代码：

if (!PathIsDirectory("F:\\xin"))

{

CreateDirectory("F:\\xin",0);

}

IplImage\* pFrame = NULL;

CvCapture\* pCapture = cvCreateCameraCapture(0);

cvNamedWindow("video", 1);

HWND hWnd = (HWND)cvGetWindowHandle("video");

::SetWindowPos(hWnd,HWND\_NOTOPMOST,0,0,0,0,SWP\_NOMOVE|SWP\_NOSIZE|SWP\_SHOWWINDOW);

int nFrmNum = 0;

string face\_cascade\_name = "D:\\Program Files\\opencv249\\opencv\\sources\\data\\haarcascades\\haarcascade\_frontalface\_alt2.xml";

CascadeClassifier face\_cascade;

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

{

AfxMessageBox(\_T("级联分类器错误，可能未找到文件，拷贝该文件到工程目录下！"));

}

**检测人脸并保存人脸建立自己的人脸样本**

点击“开始检测”按钮，打开摄像头，并开始进行人脸检测，检测出人脸之后改变大小使之与ORL人脸数据库人脸大小一致，并保存为人脸样本。

核心代码如下：

while(pFrame = cvQueryFrame(pCapture))

{

//cvShowImage("video",pFrame);

//char c = (char)waitKey(10);

//if(c==27)

//break;

nFrmNum ++;

int face\_num=0;

vector<Rect> faces;

IplImage\* face\_gray;

face\_gray = cvCreateImage(cvSize(pFrame->width, pFrame->height), IPL\_DEPTH\_8U,1);

cvCvtColor( pFrame, face\_gray, CV\_BGR2GRAY ); //rgb类型转换为灰度类型

cvEqualizeHist( face\_gray, face\_gray ); //直方图均衡化

face\_cascade.detectMultiScale( face\_gray, faces, 1.1, 2, 0|CV\_HAAR\_SCALE\_IMAGE, Size(80, 80) );

face\_num=faces.size();

char faceName[30];

fi++;

sprintf(faceName,"F:\\xin\\%d.jpg",fi);

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

{

cvNamedWindow("face",1);

cvRectangle(pFrame,cvPoint(faces[i].x,faces[i].y),cvPoint(faces[i].x+faces[i].width,faces[i].y+faces[i].height), CV\_RGB(255,0,0), 1);

cvSetImageROI(pFrame,cvRect( faces[i].x, faces[i].y, faces[i].width, faces[i].height));

//在目标图像中截出人脸区域

IplImage\* face1 = cvCreateImage(cvSize(faces[i].width,faces[i].height),pFrame->depth,pFrame->nChannels);

IplImage\* face2 = cvCreateImage(cvSize(92, 112),pFrame->depth,pFrame->nChannels);

cvCopy(pFrame,face1);

//if (face1->width > 100)

//{

cvResize(face1, face2, CV\_INTER\_CUBIC);

//}

cvShowImage("Face",face2);

cvResetImageROI(pFrame);

char c = (char)waitKey(0);

if(c=='e') break;

switch(c)

{

case 'p':

cvSaveImage(faceName,face2);

break;

default:

;

}

cvDestroyWindow("face");

cvReleaseImage(&face1);

cvReleaseImage(&face2);

}

cvShowImage("video",pFrame);

cvReleaseImage(&face\_gray );

if(waitKey(0)==27) break;

}

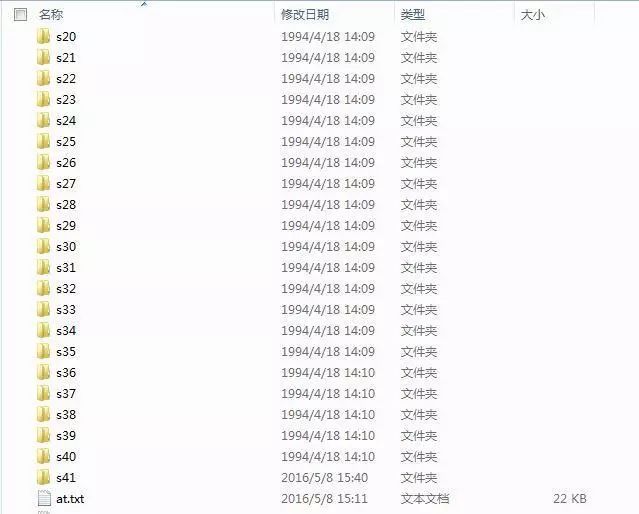
//cvWaitKey(300);

cvReleaseCapture(&pCapture);

cvDestroyWindow("video");

cvReleaseImage(&pFrame);

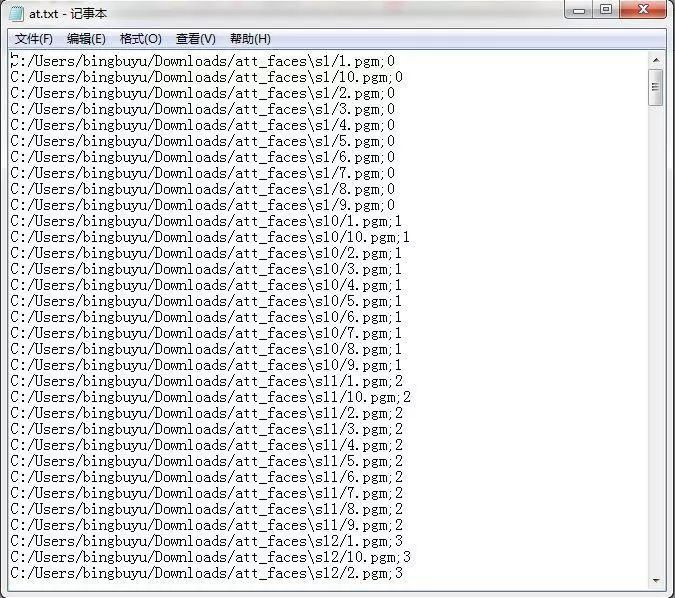
至此，我们就得到和ORL人脸数据库人脸大小一致的自己的人脸数据集。然后我们把自己的作为第41个人（在我们下载的人脸文件夹下建立一个s41的子文件夹，把自己的人脸数据放进去）。就成了这样下面这样，最后一个文件夹里面是自己的人脸数据集：



**二、训练模型**

生成CSV文件

当写人脸模型的训练程序的时候，需要读取人脸和人脸对应的标签。直接在数据库中读取显然是低效的，所以采用csv文件读取。csv文件中包含两方面的内容，一是每一张图片的位置所在，二是每一个人脸对应的标签，就是为每一个人编号。这个at.txt就是我们需要的csv文件。生成之后它里面是这个样子的：



**前面是图片的位置，后面是图片所属人脸的人的标签。**

要生成这样一个文件直接用手工的方式一个一个输入显然不可取的，毕竟这里有400多张图片。而且这种重复性的工作估计也没人想去做。所以我们可以用**命令行**的方式简化工作量；或者用opencv自带的**Python脚本**来自动生成。

**命令行方式：**

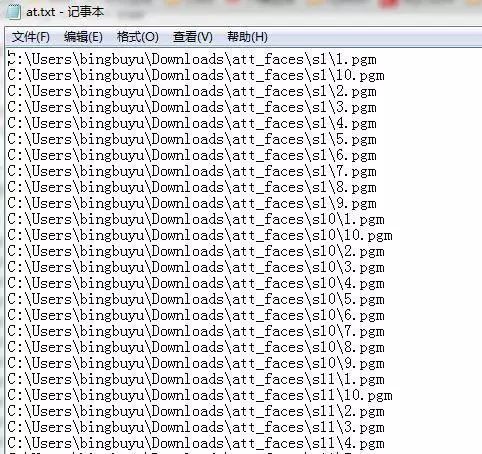
进入DOS命令，按如下步骤即可。

1. 打开cmd

2. 输入cd F:\Face DR\faces点回车进入数据集所在文件夹

3. 输入dir /b/s/p/w \*.pgm \*.jpg >at.txt

然后数据集文件夹下面就多出了一个at.txt文件，但是现在是只有路径没有标签的。像下面这样：



标签需要手动敲上去，也挺麻烦的。

**Python脚本方式：**

在opencv教程里面为我们提供了自动生成csv文件的脚本。路径：F:\opencv\sources\modules\contrib\doc\facerec\src\create\_csv.py。把代码里面的BASE\_PATH手动的改成自己的数据集路径，改完大致是这样：

#!/usr/bin/env python  
 import sys  
import os.path  
 # This is a tiny script to help you creating a CSV file from a face  
# database with a similar hierarchie:  
#  
#  philipp@mango:~/facerec/data/at$ tree  
#  .  
#  |-- README #  |-- s1  
#  |   |-- 1.pgm #  |   |-- ...  
#  |   |-- 10.pgm  
 #  |-- s2  
#  |   |-- 1.pgm  
#  |   |-- ...  
#  |   |-- 10.pgm  
#  ...  
#  |-- s40  
 #  |   |-- 1.pgm  
#  |   |-- ...  
#  |   |-- 10.pgm  
#  
 if \_\_name\_\_ == "\_\_main\_\_":

#if len(sys.argv) != 2:      
#    print "usage: create\_csv <base\_path>"      
#    sys.exit(1)      
#BASE\_PATH=sys.argv[1]      
BASE\_PATH="C:/Users/bingbuyu/Downloads/att\_faces"      
SEPARATOR=";"      
fh = open("../etc/at.txt",'w')      
label = 0      
for dirname, dirnames, filenames in os.walk(BASE\_PATH):          
for subdirname in dirnames:              
subject\_path = os.path.join(dirname, subdirname)              
for filename in os.listdir(subject\_path):                  
abs\_path = "%s/%s" % (subject\_path, filename)

print "%s%s%d" % (abs\_path, SEPARATOR, label)                 
 fh.write(abs\_path)                  
fh.write(SEPARATOR)                 
 fh.write(str(label))                 
 fh.write("\n")                    
label = label + 1  
 fh.close()

然后运行这个脚本就可以生成一个既有路径又有标签的at.txt了。

现在数据集、csv文件都已经准备好了。接下来要做的就是训练模型了。

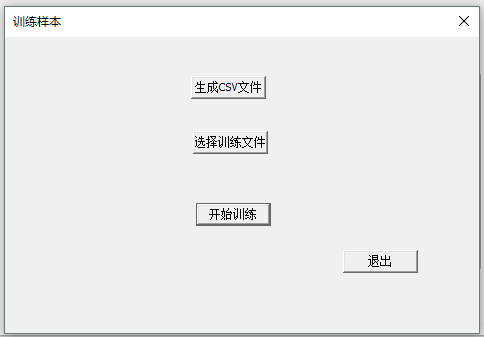


图2

点击“选择训练文件”，打开对话框，选择前面生成的at.txt文件。

代码：

void MyNDlg::OnBnClickedButton1()

{

// TODO: 在此添加控件通知处理程序代码

//CFileDialog dlg(TRUE);///TRUE为OPEN对话框，FALSE为SAVE AS对话框

//CFileDialog(TRUE,"","",OFN\_FILEMUSTEXIST |OFN\_HIDEREADONLY , "文件类型(\*.yourfiletype)|\*.yourfiletype||");//生成一个对话框

CFileDialog dlg(TRUE,"","",OFN\_FILEMUSTEXIST |OFN\_HIDEREADONLY,"文件类型(\*.txt)|\*.txt||");//生成一个对话框

if(dlg.DoModal()==IDOK)

// csvPath1=dlg.GetPathName();

csvPath1=dlg.GetPathName();

}

点击“开始训练”，后台自动训练人脸识别模型，并将模型存入文件保存。

void MyNDlg::OnBnClickedStart()

{

// TODO: 在此添加控件通知处理程序代码

//String csvPath = "F:\\人脸项目\\文件-图片\\people\\at.txt";

std::string csvPath((LPCTSTR)csvPath1);

//CString tmp;

//tmp=csvPath.c\_str();

//AfxMessageBox(tmp);

vector<Mat> images;

vector<int> labels;

char separator = ';';

ifstream file(csvPath.c\_str(),ifstream::in); //以读入的方式打开文件

String line,path,label;

while (getline(file,line)) //从文本文件中读取一行字符，未指定限定符默认限定符为“/n”

{

stringstream lines(line);

getline(lines,path,separator); //根据指定分割符进行分割，分为“路径+标号”

getline(lines,label);

if (!path.empty()&&!label.empty()) //如果读取成功，则将图片和对应标签压入对应容器中

{

images.push\_back(imread(path,0)); //读取训练样本

labels.push\_back(atoi(label.c\_str())); //读取训练样本标号

}

}

Ptr<FaceRecognizer> modelPCA1 = createEigenFaceRecognizer();

modelPCA1->train(images,labels);

modelPCA1->save("F:\\people\\PCA\_Model.xml");

Ptr<FaceRecognizer> modelFisher1 = createFisherFaceRecognizer();

modelFisher1->train(images,labels);

modelFisher1->save("F:\\people\\Fisher\_Model.xml");

Ptr<FaceRecognizer> modelLBP1 = createLBPHFaceRecognizer();

modelLBP1->train(images,labels);

modelLBP1->save("F:\\people\\LBP\_Model.xml");

AfxMessageBox(\_T("训练成功！"));

}

**三、测试**

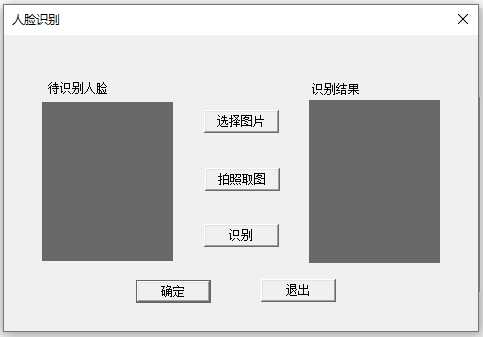


图3

人脸识别的程序布局图4所示，左侧为待识别人脸的图片区域，右侧为识别结果区域，中间是三个操作按钮。第一个操作按钮为选择图片，选取已经拍摄或保存下来的待识别人脸的图片放在左侧区域。第二个操作按钮是拍照取图，点击按钮会启用摄像头，进而捕捉待识别人脸并将图片放在左侧区域。**第一个操作按钮与第二个操作按钮都是选取待识别人脸，因此只能选择一个操作，二者不能同时选择。**第三个操作按钮为识别，点击按钮即可进行人脸识别。结果如图5和图6所示。

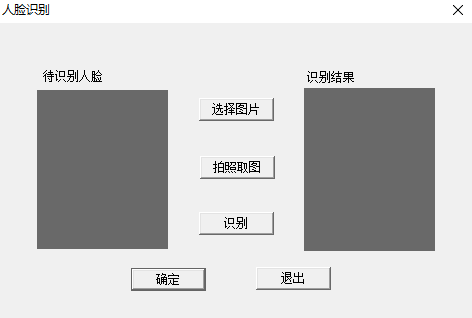


图4

void CMyNewDlg::OnBnClickedButton2()

{

IplImage \*image=NULL;

if(b\_Flag)

{

image=cvLoadImage(FilePathName3,0);

}

if(b\_Flag1)

{

image=cvLoadImage(faceName,0);

}

if(!PathFileExists("F:\\实验\\文件\\PCA\_Model.xml")||!PathFileExists("F:\\实验\\文件\\Fisher\_Model.xml")||!PathFileExists("F:\\实验\\文件\\LBP\_Model.xml"))

AfxMessageBox(\_T("请确保识别文件在G:\\毕业\\people文件夹中！"));

if(!image)

AfxMessageBox(\_T("请选择待识别图片！"));

if(image)

{

// TODO: 在此添加控件通知处理程序代码

Ptr<FaceRecognizer> modelPCA = createEigenFaceRecognizer();

modelPCA->load("F:\\实验\\文件\\PCA\_Model.xml");

Ptr<FaceRecognizer> modelFisher = createFisherFaceRecognizer();

modelFisher->load("F:\\实验\\文件\\Fisher\_Model.xml");

Ptr<FaceRecognizer> modelLBP = createLBPHFaceRecognizer();

modelLBP->load("F:\\实验\\文件\\LBP\_Model.xml");

Mat testImage(image,0);

//Mat testImage(image);

Mat MyFace;

if (testImage.cols!=92||testImage.rows!=112)

{

resize(testImage, MyFace, Size(92,112));

int predictPCA = modelPCA->predict(MyFace);

int predictLBP = modelLBP->predict(MyFace);

int predictFisher = modelFisher->predict(MyFace);

if(predictPCA ==predictLBP||predictLBP==predictFisher||predictFisher==predictPCA)

{

if(predictPCA ==35||predictLBP==35||predictFisher==35)

{

Information\_Box->SetWindowText(\_T("\r\n田海蓉"));

AfxMessageBox(\_T("识别成功"),MB\_ICONINFORMATION);

}

//MessageBox("田海蓉",\_T("程序运行结果"),MB\_OK);

Invalidate();

}

else

{

AfxMessageBox(\_T("查无此人"),MB\_ICONERROR);

Invalidate();

}

}

else{

int predictPCA = modelPCA->predict(testImage);

int predictLBP = modelLBP->predict(testImage);

int predictFisher = modelFisher->predict(testImage);

if(predictPCA ==predictLBP||predictLBP==predictFisher||predictFisher==predictPCA)

{

if(predictPCA ==35||predictLBP==35||predictFisher==35)

{

Information\_Box->SetWindowText(\_T("\r\n田海蓉"));

AfxMessageBox(\_T("识别成功"),MB\_ICONINFORMATION);

}

Invalidate();

}

else

{

AfxMessageBox(\_T("查无此人"),MB\_ICONERROR);

Invalidate();

}

}

}

}

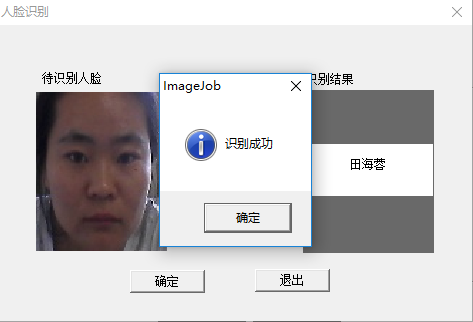


图5



图6

另可参考以下材料进行实验：

**数据收集和预处理**

https://mp.weixin.qq.com/s?\_\_biz=MzI2OTc5NTUwNg==&mid=2247484742&idx=2&sn=f38fd7520f5228caf9d1971095e33a3b&chksm=eadbac9bddac258d68bd106cc73e66c816293fe80c50195aeea390e8b94839dac8295021d517&scene=21#wechat\_redirect

**模型训练 与测试**

https://mp.weixin.qq.com/s?\_\_biz=MzI2OTc5NTUwNg==&mid=2247484752&idx=3&sn=6ea3a7fa18a58d8b53e3436252491df5&chksm=eadbac8dddac259bf840e794fd2cfe6c92c7c849db9d02cd000f70a78b65375d8b8bcdd64cca&scene=21#wechat\_redirect