**浙江师范大学**

**研究生课程论文封面**

|  |  |
| --- | --- |
| 课程名称： | **模式识别** |
| 开课时间： | **1-16周** |
| 学 院 | 工学院 |
| 学科专业 | 智能交通技术 |
| 学 号 | 201820200784 |
| 姓 名 | 沈迦伟 |
| 任课教师 | 郑忠龙 |
| 交稿日期 | 6月22日 |
| 成 绩 |  |
| 评阅日期 |  |
| 评阅教师签名 |  |

**浙江师范大学研究生院制**

# 人脸识别程序

## 一、技术要点

传统目标检测算法——级联分类器Cascade

1. 加载级联分类器

CascadeClassifier face\_cascade;   
face\_cascade.load("haarcascade\_frontalface\_alt.xml"); //读取分类器训练好后的文件

1. 读取视频流
2. 将图像灰度化，并直方图均衡化化，对每一帧使用分类器。

   cascade.detectMultiScale( smallImg, faces,  
       1.1, 2, 0  
       //|CASCADE\_FIND\_BIGGEST\_OBJECT //只检测大物体  
       //|CASCADE\_DO\_ROUGH\_SEARCH //初略检验  
       |CASCADE\_SCALE\_IMAGE, //按比例检验  
       Size(30, 30) );

## 二、人脸识别内容

识别人脸

## 三、技术知识要点

目前人脸检测方法主要分为两大类，基于知识和基于统计。基于知识的人脸检测方法主要包括：模板匹配，人脸特征，形状与边缘，纹理特征，颜色特征。将人脸看成不同特征的特定组合，即通过人脸的眼睛、嘴巴、鼻子、耳朵等特征及其组合关系来检测人脸。基于统计的人脸检测方法包括：主成分分析和特征脸法，神经网络模型，隐马尔可夫模型，支持向量机，Adaboost等法。

本文采用opencv工程下sample文件作为人脸识别程序facedectective.cpp作为识别程序识别人脸。

## 四、实验环境准备

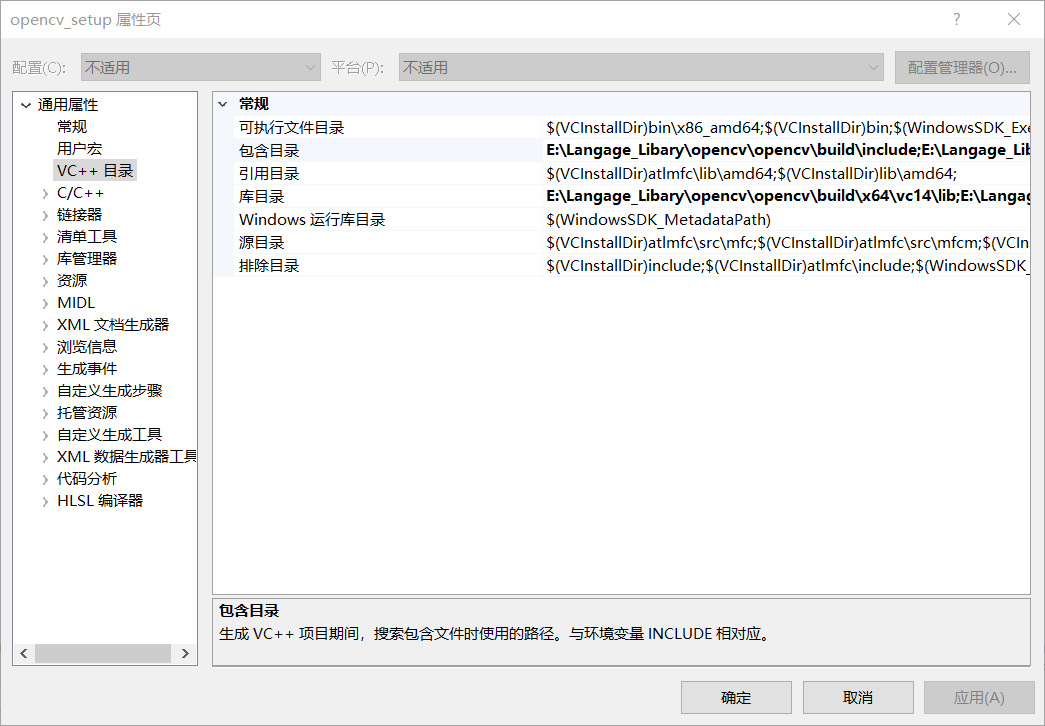
操作系统：win7 系统

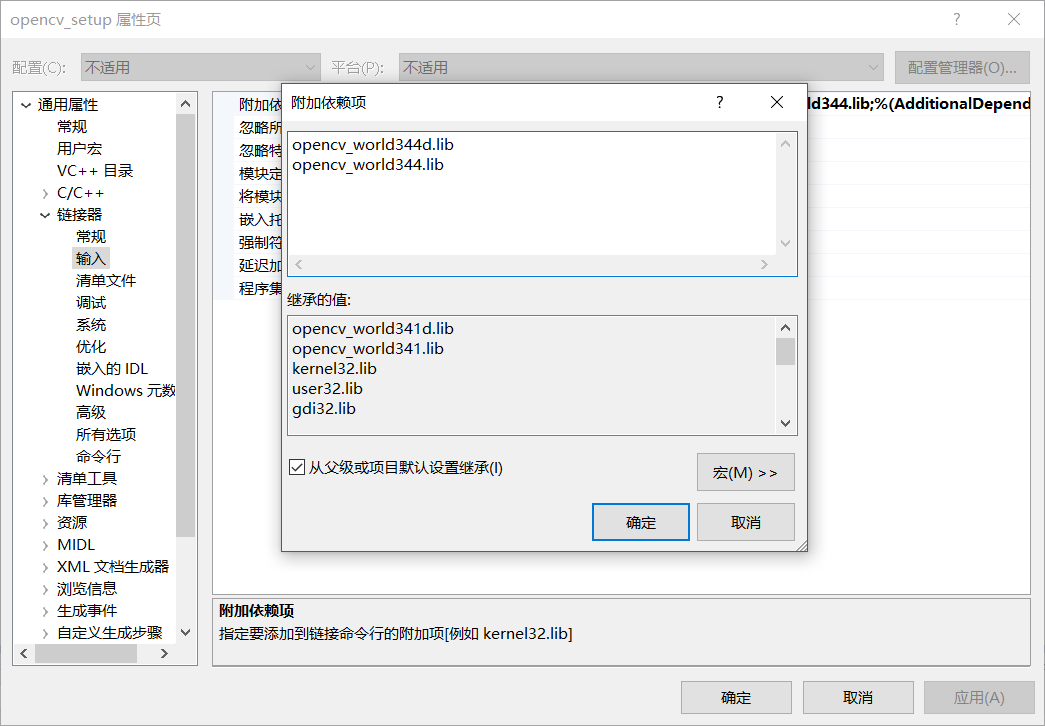
IDE：Visual Studio 2010

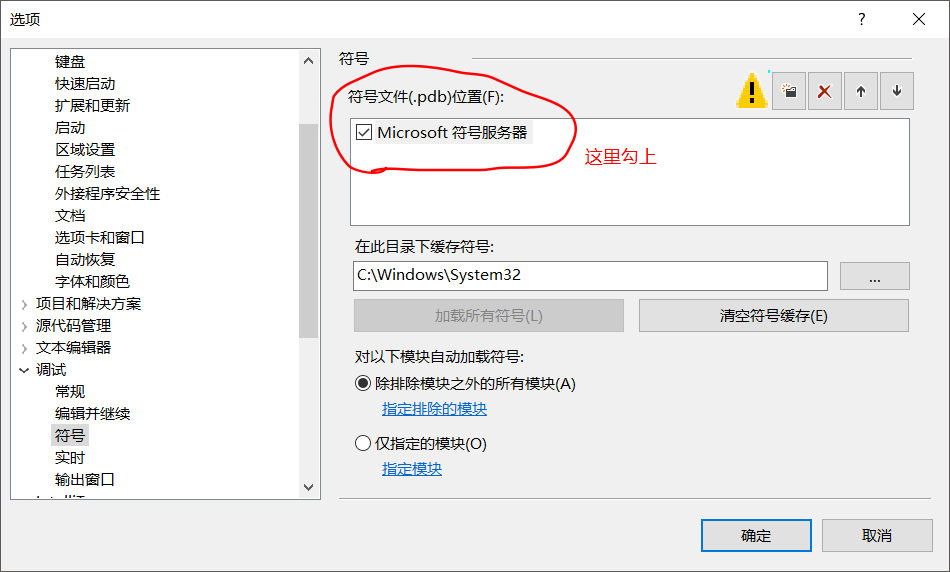
代码框架：opencv

## 五、实验步骤

1.根据网络博文<https://www.cnblogs.com/aiguona/p/9370433.html>配置Opencv环境。

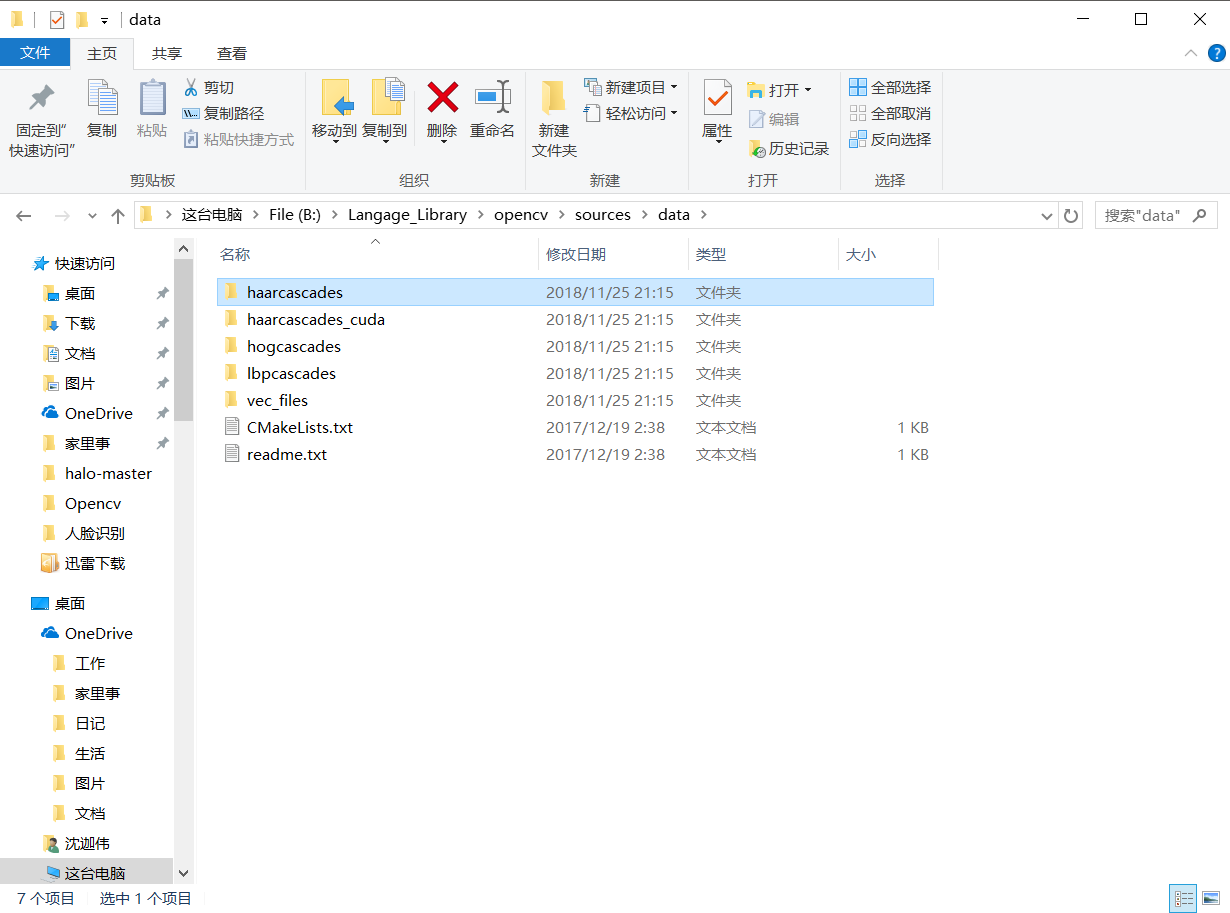






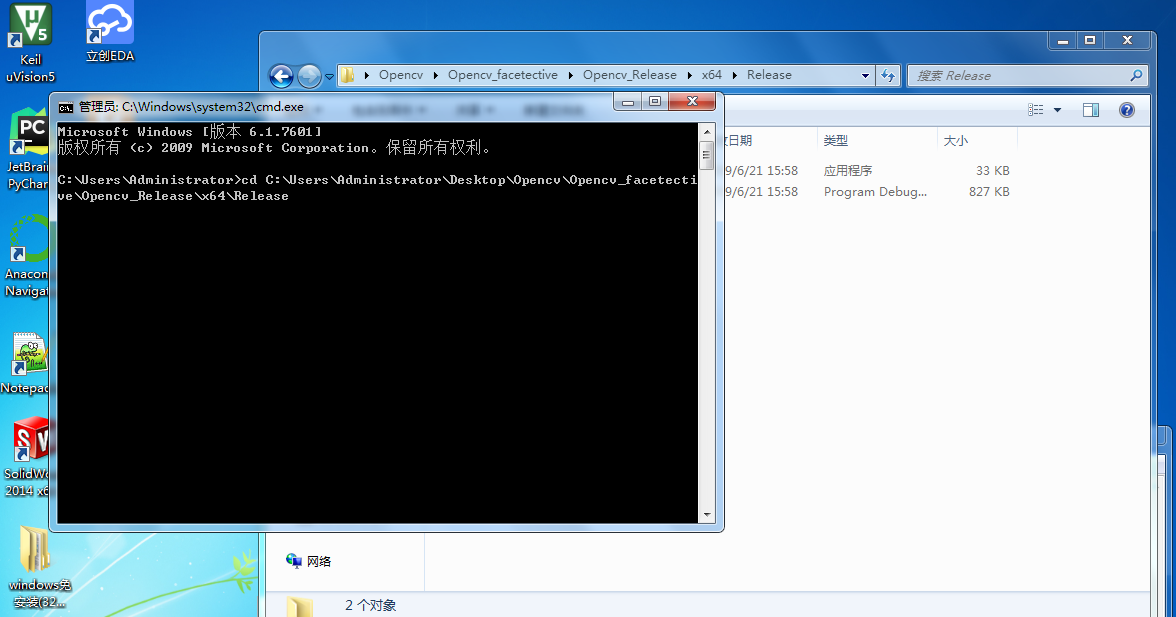
注：缺少DLL，可从网络下载。

2. 写入代码，将Opencv开源工程文件中data目录下haarcascades文件复制到工程目录下



3．打开windows下cmd命令行，移动到打包的exe文件目录下

$cd C:\Users\Administrator\Desktop\Opencv\Opencv\_facetective\Opencv\_Release\x64\Release



3．在命令行下输入下例语句，了解程序

$facetective --help

得到下列程序执行结果

Usage:  
./facedetect [--cascade=<cascade\_path> this is the primary trained classifier such as frontal face]  
  [--nested-cascade[=nested\_cascade\_path this an optional secondary classifier such as eyes]]  
  [--scale=<image scale greater or equal to 1, try 1.3 for example>]  
  [--try-flip]  
  [filename|camera\_index]  
​  
see facedetect.cmd for one call:  
./facedetect --cascade="data/haarcascades/haarcascade\_frontalface\_alt.xml" --nested-cascade="data/haarcascades/haarcascade\_eye\_tree\_eyeglasses.xml" --scale=1.3  
​  
During execution:  
Hit any key to quit.  
Using OpenCV version 3.4.4

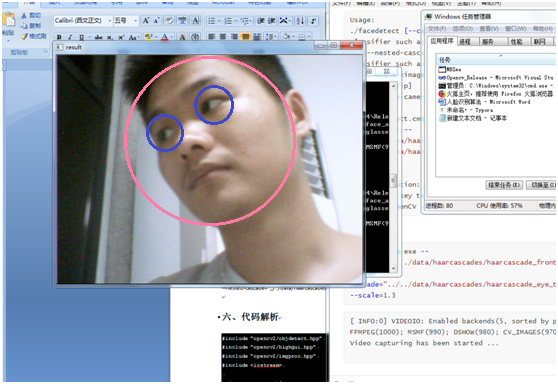
4．输入执行识别程序

命令行输入下列语句

$facetective.exe --cascade="../../data/haarcascades/haarcascade\_frontalface\_alt.xml" --  
nested-cascade="../../data/haarcascades/haarcascade\_eye\_tree\_eyeglasses.xml" --scale=1.3



[ INFO:0] VIDEOIO: Enabled backends(5, sorted by priority): FFMPEG(1000); MSMF(990); DSHOW(980); CV\_IMAGES(970); CV\_MJPEG(960)  
Video capturing has been started ...  
​detection time =678.218 ms



## 六、代码解析

#include "opencv2/objdetect.hpp"

#include "opencv2/highgui.hpp"

#include "opencv2/imgproc.hpp"

#include <iostream>

using namespace std;

using namespace cv;

//命令行帮助函数

static void help()

{

cout << "Usage:\n"

"./facedetect [--cascade=<cascade\_path> this is the primary trained classifier such as frontal face]\n"

" [--nested-cascade[=nested\_cascade\_path this an optional secondary classifier such as eyes]]\n"

" [--scale=<image scale greater or equal to 1, try 1.3 for example>]\n"

" [--try-flip]\n"

" [filename|camera\_index]\n\n"

"see facedetect.cmd for one call:\n"

"./facedetect --cascade=\"data/haarcascades/haarcascade\_frontalface\_alt.xml\" --nested-cascade=\"data/haarcascades/haarcascade\_eye\_tree\_eyeglasses.xml\" --scale=1.3\n\n"

"During execution:\n\tHit any key to quit.\n"

"\tUsing OpenCV version " << CV\_VERSION << "\n" << endl;

}

//人脸识别程序

void detectAndDraw( Mat& img, CascadeClassifier& cascade,

CascadeClassifier& nestedCascade,

double scale, bool tryflip );

//用户输入的分类器地址

string cascadeName;

string nestedCascadeName;

int main( int argc, const char\*\* argv )

{

//初始化

VideoCapture capture;

Mat frame, image;

string inputName;

bool tryflip;

CascadeClassifier cascade, nestedCascade;

double scale;

cv::CommandLineParser parser(argc, argv,

"{help h||}"

"{cascade|data/haarcascades/haarcascade\_frontalface\_alt.xml|}"

"{nested-cascade|data/haarcascades/haarcascade\_eye\_tree\_eyeglasses.xml|}"

"{scale|1|}{try-flip||}{@filename||}"

);

//命令行判断用户输入语句

//判断

if (parser.has("help"))

{

help();

return 0;

}

//命令行中得到cascade分类器地址

cascadeName = parser.get<string>("cascade");

nestedCascadeName = parser.get<string>("nested-cascade");

scale = parser.get<double>("scale");

//图像左右翻转标志与图像尺度

if (scale < 1)

scale = 1;

tryflip = parser.has("try-flip");

inputName = parser.get<string>("@filename");

//检查命令行语句是否有误

if (!parser.check())

{

parser.printErrors();

return 0;

}

//检查分类器能否加载

if (!nestedCascade.load(samples::findFileOrKeep(nestedCascadeName)))

cerr << "WARNING: Could not load classifier cascade for nested objects" << endl;

if (!cascade.load(samples::findFile(cascadeName)))

{

cerr << "ERROR: Could not load classifier cascade" << endl;

help();

return -1;

}

//检测是否以载入视频文件的方式

if( inputName.empty() || (isdigit(inputName[0]) && inputName.size() == 1) )

{

int camera = inputName.empty() ? 0 : inputName[0] - '0';

if(!capture.open(camera))

{

cout << "Capture from camera #" << camera << " didn't work" << endl;

return 1;

}

}

else if (!inputName.empty())

{

image = imread(samples::findFileOrKeep(inputName), IMREAD\_COLOR);

if (image.empty())

{

if (!capture.open(samples::findFileOrKeep(inputName)))

{

cout << "Could not read " << inputName << endl;

return 1;

}

}

}

else

//检查是否可以读入图像

{

image = imread(samples::findFile("lena.jpg"), IMREAD\_COLOR);

if (image.empty())

{

cout << "Couldn't read lena.jpg" << endl;

return 1;

}

}

//打开摄像头

if( capture.isOpened() )

{

cout << "Video capturing has been started ..." << endl;

//读摄像头

for(;;)

{

capture >> frame; //读入图像桢

if( frame.empty() )

break;

Mat frame1 = frame.clone();

detectAndDraw( frame1, cascade, nestedCascade, scale, tryflip ); //判断图

char c = (char)waitKey(10); //10ms内等待按键码

if( c == 27 || c == 'q' || c == 'Q' )

break;

}

}

else

{

cout << "Detecting face(s) in " << inputName << endl;

if( !image.empty() )

{

detectAndDraw( image, cascade, nestedCascade, scale, tryflip );

waitKey(0);

}

else if( !inputName.empty() )

{

/\* assume it is a text file containing the

list of the image filenames to be processed - one per line \*/

FILE\* f = fopen( inputName.c\_str(), "rt" );

if( f )

{

char buf[1000+1];

while( fgets( buf, 1000, f ) )

{

int len = (int)strlen(buf);

while( len > 0 && isspace(buf[len-1]) )

len--;

buf[len] = '\0';

cout << "file " << buf << endl;

image = imread( buf, 1 );

if( !image.empty() )

{

detectAndDraw( image, cascade, nestedCascade, scale, tryflip );

char c = (char)waitKey(0);

if( c == 27 || c == 'q' || c == 'Q' )

break;

}

else

{

cerr << "Aw snap, couldn't read image " << buf << endl;

}

}

fclose(f);

}

}

}

return 0;

}

//检测人脸函数

void detectAndDraw( Mat& img, CascadeClassifier& cascade,

CascadeClassifier& nestedCascade,

double scale, bool tryflip )

{

double t = 0;

vector<Rect> faces, faces2;

const static Scalar colors[] =

{

Scalar(255,0,0),

Scalar(255,128,0),

Scalar(255,255,0),

Scalar(0,255,0),

Scalar(0,128,255),

Scalar(0,255,255),

Scalar(0,0,255),

Scalar(255,0,255)

};

Mat gray, smallImg;

cvtColor( img, gray, COLOR\_BGR2GRAY );

double fx = 1 / scale;

resize( gray, smallImg, Size(), fx, fx, INTER\_LINEAR\_EXACT );

equalizeHist( smallImg, smallImg );

t = (double)getTickCount();

cascade.detectMultiScale( smallImg, faces,

1.1, 2, 0

//|CASCADE\_FIND\_BIGGEST\_OBJECT

//|CASCADE\_DO\_ROUGH\_SEARCH

|CASCADE\_SCALE\_IMAGE,

Size(30, 30) );

if( tryflip )

{

flip(smallImg, smallImg, 1);

cascade.detectMultiScale( smallImg, faces2,

1.1, 2, 0

//|CASCADE\_FIND\_BIGGEST\_OBJECT

//|CASCADE\_DO\_ROUGH\_SEARCH

|CASCADE\_SCALE\_IMAGE,

Size(30, 30) );

for( vector<Rect>::const\_iterator r = faces2.begin(); r != faces2.end(); ++r )

{

faces.push\_back(Rect(smallImg.cols - r->x - r->width, r->y, r->width, r->height));

}

}

t = (double)getTickCount() - t;

printf( "detection time = %g ms\n", t\*1000/getTickFrequency());

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

{

Rect r = faces[i];

Mat smallImgROI;

vector<Rect> nestedObjects;

Point center;

Scalar color = colors[i%8];

int radius;

double aspect\_ratio = (double)r.width/r.height;

if( 0.75 < aspect\_ratio && aspect\_ratio < 1.3 )

{

center.x = cvRound((r.x + r.width\*0.5)\*scale);

center.y = cvRound((r.y + r.height\*0.5)\*scale);

radius = cvRound((r.width + r.height)\*0.25\*scale);

circle( img, center, radius, color, 3, 8, 0 );

}

else

rectangle( img, Point(cvRound(r.x\*scale), cvRound(r.y\*scale)),

Point(cvRound((r.x + r.width-1)\*scale), cvRound((r.y + r.height-1)\*scale)),

color, 3, 8, 0);

if( nestedCascade.empty() )

continue;

smallImgROI = smallImg( r );

nestedCascade.detectMultiScale( smallImgROI, nestedObjects,

1.1, 2, 0

//|CASCADE\_FIND\_BIGGEST\_OBJECT

//|CASCADE\_DO\_ROUGH\_SEARCH

//|CASCADE\_DO\_CANNY\_PRUNING

|CASCADE\_SCALE\_IMAGE,

Size(30, 30) );

//在找到人脸特征处画圆

for ( size\_t j = 0; j < nestedObjects.size(); j++ )

{

Rect nr = nestedObjects[j];

center.x = cvRound((r.x + nr.x + nr.width\*0.5)\*scale);

center.y = cvRound((r.y + nr.y + nr.height\*0.5)\*scale);

radius = cvRound((nr.width + nr.height)\*0.25\*scale);

circle( img, center, radius, color, 3, 8, 0 );

}

}

imshow( "result", img );

}