V with Practical Computer Vision Projects >

转载 easthu 于 2014-05-05 16:59:24 发布 13940 收藏 12
分类专栏： OpenCV

代码来源于<Mastering OpenCV with Practical Computer Vision Projects >

她的另外几篇文章，也翻译的很好

http://blog.csdn.net/raby_gyl/article/details/12611861

http://blog.csdn.net/raby_gyl/article/details/12623539

http://blog.csdn.net/raby_gyl/article/details/12338371

我感觉下面的程序——对人眼定位，人脸矫正，人脸尺寸化，对于初学人脸识别，做人脸的预处理非常有帮助~

程序的思路是：首先通过人脸检测级联器检测到人脸区域，对于人脸区域我们采用经验理论（即不同的人眼检测器在不同的人脸搜索区域具有最优性），也即人眼在人脸区域中的位置，得到人眼的大体位置，采用opencv的人眼级联检测器检测人眼，获取每一个人眼的中心位置，两个人眼的连线与水平位置的夹角来确定人脸旋转、矫正的角度。同时通过我们想要得到的目标图像来计算得到仿射矩阵的scale尺度因子，即图像缩放的比例，以及通过平移来计算得到的距离，进而实现定位人眼在目的图像的位置。

代码如下：

```
#include "stdafx.h"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/opencv.hpp"
#include<iostream>
#include<vector>
using namespace std;
using namespace cv;

const double DESIRED_LEFT_EYE_X = 0.16;     // 控制处理后人脸的多少部分是可见的
const double DESIRED_LEFT_EYE_Y = 0.14;
const double FACE_ELLIPSE_CY = 0.40;
const double FACE_ELLIPSE_W = 0.50;         // 应当至少为0.5
const double FACE_ELLIPSE_H = 0.80;         //控制人脸掩码的高度

/*-----------------------------------目标检测----------------------------------*/
void detectObjectsCustom(const Mat &img, CascadeClassifier &cascade, vector<Rect> &objects, int scaledWidth, int flags, Size minFeatureSize, float searchScaleFactor, int minNeighbors);
void detectLargestObject(const Mat &img, CascadeClassifier &cascade, Rect &largestObject, int scaledWidth);
void detectManyObjects(const Mat &img, CascadeClassifier &cascade, vector<Rect> &objects, int scaledWidth);
/*----------------------------------- end----------------------------------------*/

void detectBothEyes(const Mat &face, CascadeClassifier &eyeCascade1, CascadeClassifier &eyeCascade2, Point &leftEye, Point &rightEye, Rect *searchedLeftEye, Rect *searchedRightEye);
Mat getPreprocessedFace(Mat &srcImg, int desiredFaceWidth, CascadeClassifier &faceCascade, CascadeClassifier &eyeCascade1, CascadeClassifier &eyeCascade2, bool doLeftAndRightSeparately, Rect *storeFaceRect, Point *storeLeftEye, Point *storeRightEye, Rect *searchedLeftEye, Rect *searchedRightEye);

int main(int argc,char **argv)
{
    CascadeClassifier faceDetector;
    CascadeClassifier  eyeDetector1;
    CascadeClassifier  eyeDetector2;//未初始化不用

    try{
        //faceDetector.load("E:\\OpenCV-2.3.0\\data\\haarcascades\\haarcascade_frontalface_alt.xml");
        faceDetector.load("E:\\OpenCV-2.3.0\\data\\lbpcascades\\lbpcascade_frontalface.xml");
        eyeDetector1.load("E:\\OpenCV-2.3.0\\data\\haarcascades\\haarcascade_eye.xml");
        eyeDetector2.load("E:\\OpenCV-2.3.0\\data\\haarcascades\\haarcascade_eye_tree_eyeglasses.xml");


    }catch (cv::Exception e){}
    if(faceDetector.empty())
    {
        cerr<<"error:couldn't load face detector (";
        cerr<<"lbpcascade_frontalface.xml)!"<<endl;
        exit(1);
    }

    Mat img=imread(argv[1],1);
    Rect largestObject;
    const int scaledWidth=320;
    detectLargestObject(img,faceDetector,largestObject,scaledWidth);
    Mat img_rect(img,largestObject);
    Point leftEye,rightEye;
    Rect searchedLeftEye,searchedRightEye;
    detectBothEyes(img_rect,eyeDetector1,eyeDetector2,leftEye,rightEye,&searchedLeftEye,&searchedRightEye);
    //仿射变换
    Point2f eyesCenter;
    eyesCenter.x=(leftEye.x+rightEye.x)*0.5f;
    eyesCenter.y=(leftEye.y+rightEye.y)*0.5f;
    cout<<"左眼中心坐标 "<<leftEye.x<<" and "<<leftEye.y<<endl;
    cout<<"右眼中心坐标 "<<rightEye.x<<" and "<<rightEye.y<<endl;
    //获取两个人眼的角度
    double dy=(rightEye.y-leftEye.y);
    double dx=(rightEye.x-leftEye.x);
    double len=sqrt(dx*dx+dy*dy);
    cout<<"dx is "<<dx<<endl;
    cout<<"dy is "<<dy<<endl;
    cout<<"len is "<<len<<endl;
    double angle=atan2(dy,dx)*180.0/CV_PI;
    const double DESIRED_RIGHT_EYE_X=1.0f-0.16;
    //得到我们想要的尺度化大小
    const int DESIRED_FACE_WIDTH=70;
    const int DESIRED_FACE_HEIGHT=70;
    double desiredLen=(DESIRED_RIGHT_EYE_X-0.16);
    cout<<"desiredlen is "<<desiredLen<<endl;
    double scale=desiredLen*DESIRED_FACE_WIDTH/len;
    cout<<"the scale is "<<scale<<endl;
    Mat rot_mat = getRotationMatrix2D(eyesCenter, angle, scale);
    double ex=DESIRED_FACE_WIDTH * 0.5f - eyesCenter.x;
    double ey = DESIRED_FACE_HEIGHT * DESIRED_LEFT_EYE_Y-eyesCenter.y;
    rot_mat.at<double>(0, 2) += ex;
    rot_mat.at<double>(1, 2) += ey;
    Mat warped = Mat(DESIRED_FACE_HEIGHT, DESIRED_FACE_WIDTH,CV_8U, Scalar(128));
    warpAffine(img_rect, warped, rot_mat, warped.size());

    imshow("warped",warped);

    rectangle(img,Point(largestObject.x,largestObject.y),Point(largestObject.x+largestObject.width,largestObject.y+largestObject.height),Scalar(0,0,255),2,8);
    rectangle(img_rect,Point(searchedLeftEye.x,searchedLeftEye.y),Point(searchedLeftEye.x+searchedLeftEye.width,searchedLeftEye.y+searchedLeftEye.height),Scalar(0,255,0),2,8);
    rectangle(img_rect,Point(searchedRightEye.x,searchedRightEye.y),Point(searchedRightEye.x+searchedRightEye.width,searchedRightEye.y+searchedRightEye.height),Scalar(0,255,0),2,8);


    //getPreprocessedFace
    imshow("img_rect",img_rect);
    imwrite("img_rect.jpg",img_rect);
    imshow("img",img);
    waitKey();

}



/*
1、采用给出的参数在图像中寻找目标，例如人脸
2、可以使用Haar级联器或者LBP级联器做人脸检测，或者甚至眼睛，鼻子，汽车检测
3、为了使检测更快，输入图像暂时被缩小到'scaledWidth'，因为寻找人脸200的尺度已经足够了。
```

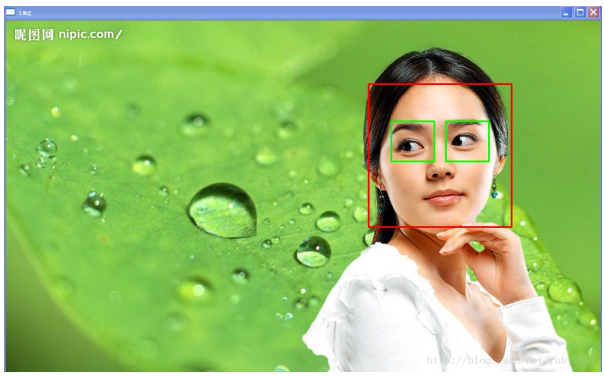
分类专栏
C/C++ 52篇
算法设计 48篇
网络 15篇
Linux 60篇
Lua 2篇
OpenCV 35篇
汇编 1篇
嵌入式 1篇
数学 1篇
数据库 4篇
安全 5篇
PHP 6篇
Java 7篇
Android 38篇
C# 1篇
Web 6篇
笔试题 10篇
工具 11篇
python 1篇
windows 8篇
计算机基础 1篇
Mac 6篇
Nginx 1篇
Objective-C 5篇

```
*/
void detectObjectsCustom(const Mat &img, CascadeClassifier &cascade, vector<Rect> &objects, int scaledWidth, int flags, Size minFeatureSize, float searchScaleFactor, int minNeighbors)
{

    //如果输入的图像不是灰度图像,那么将BRG或者BGRA彩色图像转换为灰度图像
    Mat gray;
    if (img.channels() == 3) {
        cvtColor(img, gray, CV_BGR2GRAY);
    }
    else if (img.channels() == 4) {
        cvtColor(img, gray, CV_BGRA2GRAY);
    }
    else {
        // 直接使用输入图像.既然它已经是灰度图像
        gray = img;
    }

    // 可能的缩小图像.是检索更快
    Mat inputImg;

    float scale = img.cols / (float)scaledWidth;
    if (img.cols > scaledWidth) {
        // 缩小图像并保持同样的宽高比
        int scaledHeight = cvRound(img.rows / scale);
        resize(gray, inputImg, Size(scaledWidth, scaledHeight));
    }
    else {
        // 直接使用输入图像.既然它已经小了
        inputImg = gray;
    }

    //标准化亮度和对比度来改善暗的图像
    Mat equalizedImg;
    equalizeHist(inputImg, equalizedImg);

    // 在小的灰色图像中检索目标
    cascade.detectMultiScale(equalizedImg, objects, searchScaleFactor, minNeighbors, flags, minFeatureSize);

    // 如果图像在检测之前暂时的被缩小了,则放大结果图像
    if (img.cols > scaledWidth) {
        for (int i = 0; i < (int)objects.size(); i++ ) {
            objects[i].x = cvRound(objects[i].x * scale);
            objects[i].y = cvRound(objects[i].y * scale);
            objects[i].width = cvRound(objects[i].width * scale);
            objects[i].height = cvRound(objects[i].height * scale);
        }
    }

    //确保目标全部在图像内部.以防它在边界上
    for (int i = 0; i < (int)objects.size(); i++ ) {
        if (objects[i].x < 0)
            objects[i].x = 0;
        if (objects[i].y < 0)
            objects[i].y = 0;
        if (objects[i].x + objects[i].width > img.cols)
            objects[i].x = img.cols - objects[i].width;
        if (objects[i].y + objects[i].height > img.rows)
            objects[i].y = img.rows - objects[i].height;
    }

    // 返回检测到的人脸矩形,存储在objects中
}

/*
1、仅寻找图像中的单个目标.例如最大的人脸,存储结果到largestObject
2、可以使用Haar级联器或者LBP级联器做人脸检测,或者甚至眼睛,鼻子,汽车检测
3、为了使检测更快,输入图像暂时被缩小到'scaledWidth'.因为寻找人脸200的尺度已经足够了。
4、注释:detectLargestObject()要比 detectManyObjects()快。
*/
void detectLargestObject(const Mat &img, CascadeClassifier &cascade, Rect &largestObject, int scaledWidth)
{
    //仅寻找一个目标 (图像中最大的).
    int flags = CV_HAAR_FIND_BIGGEST_OBJECT;// | CASCADE_DO_ROUGH_SEARCH;
    // 最小的目标大小.
    Size minFeatureSize = Size(20, 20);
    // 寻找细节,尺度因子,必须比1大
    float searchScaleFactor = 1.1f;
    // 多少检测结果应当被滤掉.这依赖于你的检测系统是多坏,如果minNeighbors=2 .大量的good or bad 被检测到。如果
    // minNeighbors=6.意味着只good检测结果.但是一些将漏掉。即可靠性 VS  检测人脸数量
    int minNeighbors = 4;

    // 执行目标或者人脸检测.仅寻找一个目标(图像中最大的)
    vector<Rect> objects;
    detectObjectsCustom(img, cascade, objects, scaledWidth, flags, minFeatureSize, searchScaleFactor, minNeighbors);
    if (objects.size() > 0) {
        // 返回仅检测到的目标
        largestObject = (Rect)objects.at(0);
    }
    else {
        // 返回一个无效的矩阵
        largestObject = Rect(-1,-1,-1,-1);
    }
}

void detectManyObjects(const Mat &img, CascadeClassifier &cascade, vector<Rect> &objects, int scaledWidth)
{
    // 寻找图像中的许多目标
    int flags = CV_HAAR_SCALE_IMAGE;

    // 最小的目标大小.
    Size minFeatureSize = Size(20, 20);
    //  寻找细节,尺度因子,必须比1大
    float searchScaleFactor = 1.1f;
    // 多少检测结果应当被滤掉.这依赖于你的检测系统是多坏,如果minNeighbors=2 .大量的good or bad 被检测到。如果
    // minNeighbors=6.意味着只good检测结果.但是一些将漏掉。即可靠性 VS  检测人脸数量
    int minNeighbors = 4;

    // 执行目标或者人脸检测.寻找图像中的许多目标
    detectObjectsCustom(img, cascade, objects, scaledWidth, flags, minFeatureSize, searchScaleFactor, minNeighbors);
}
/*
1、在给出的人脸图像中寻找双眼.返回左眼和右眼的中心.如果当找不到人眼时,或者设置为Point(-1,-1)
2、注意如果你想用两个不同的级联器寻找人眼.你可以传递第二个人眼检测器.例如如果你使用的一个常规人眼检测器和带眼镜的人眼检测器一样好.或者左眼检测器和右眼检测器一样好.
或者如果你不想第二个检测器.仅传一个来初始化级联检测器。
3、如果需要的话.也可以存储检测到的左眼和右眼的区域
*/
void detectBothEyes(const Mat &face, CascadeClassifier &eyeCascade1, CascadeClassifier &eyeCascade2, Point &leftEye, Point &rightEye, Rect *searchedLeftEye, Rect *searchedRightEye)
{
    //跳过人脸边界.因为它们经常是头发和耳朵.这不是我们关心的
    /*
    // For "2splits.xml": Finds both eyes in roughly 60% of detected faces, also detects closed eyes.
    const float EYE_SX = 0.12f;
    const float EYE_SY = 0.17f;
    const float EYE_SW = 0.37f;
    const float EYE_SH = 0.36f;
    */
    /*
    // For mcs.xml: Finds both eyes in roughly 80% of detected faces, also detects closed eyes.
    const float EYE_SX = 0.10f;
    const float EYE_SY = 0.19f;
    const float EYE_SW = 0.40f;
    const float EYE_SH = 0.36f;
    */

    // For default eye.xml or eyeglasses.xml: Finds both eyes in roughly 40% of detected faces, but does not detect closed eyes.
    //haarcascade_eye.xml检测器在由下面确定的人脸区域内搜索最优。
    const float EYE_SX = 0.16f;//x
    const float EYE_SY = 0.26f;//y
    const float EYE_SW = 0.30f;//width
```

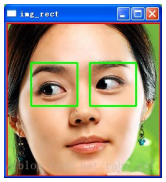
```
    const float EYE_SH = 0.28f;//height

    int leftX = cvRound(face.cols * EYE_SX);
    int topY = cvRound(face.rows * EYE_SY);
    int widthX = cvRound(face.cols * EYE_SW);
    int heightY = cvRound(face.rows * EYE_SH);
    int rightX = cvRound(face.cols * (1.0-EYE_SX-EYE_SW) );  // 右眼的开始区域

    Mat topLeftOfFace = face(Rect(leftX, topY, widthX, heightY));
    Mat topRightOfFace = face(Rect(rightX, topY, widthX, heightY));
    Rect leftEyeRect, rightEyeRect;

    // 如果需要的话，然后搜索到的窗口给调用者
    if (searchedLeftEye)
        *searchedLeftEye = Rect(leftX, topY, widthX, heightY);
    if (searchedRightEye)
        *searchedRightEye = Rect(rightX, topY, widthX, heightY);

    // 寻找左区域，然后右区域使用第一个人眼检测器
    detectLargestObject(topLeftOfFace, eyeCascade1, leftEyeRect, topLeftOfFace.cols);
    detectLargestObject(topRightOfFace, eyeCascade1, rightEyeRect, topRightOfFace.cols);

    // 如果人眼没有检测到，尝试另外一个不同的级联检测器
    if (leftEyeRect.width <= 0 && !eyeCascade2.empty()) {
        detectLargestObject(topLeftOfFace, eyeCascade2, leftEyeRect, topLeftOfFace.cols);
        //if (leftEyeRect.width > 0)
        //    cout << "2nd eye detector LEFT SUCCESS" << endl;
        //else
        //    cout << "2nd eye detector LEFT failed" << endl;
    }
    //else
    //    cout << "1st eye detector LEFT SUCCESS" << endl;

    // 如果人眼没有检测到，尝试另外一个不同的级联检测器
    if (rightEyeRect.width <= 0 && !eyeCascade2.empty()) {
        detectLargestObject(topRightOfFace, eyeCascade2, rightEyeRect, topRightOfFace.cols);
        //if (rightEyeRect.width > 0)
        //    cout << "2nd eye detector RIGHT SUCCESS" << endl;
        //else
        //    cout << "2nd eye detector RIGHT failed" << endl;
    }
    //else
    //    cout << "1st eye detector RIGHT SUCCESS" << endl;

    if (leftEyeRect.width > 0) {   // 检查眼是否被检测到
        leftEyeRect.x += leftX;    //矫正左眼矩形，因为人脸边界被去除掉了
        leftEyeRect.y += topY;
        leftEye = Point(leftEyeRect.x + leftEyeRect.width/2, leftEyeRect.y + leftEyeRect.height/2)
;
    }
    else {
        leftEye = Point(-1, -1);    // 返回一个无效的点
    }

    if (rightEyeRect.width > 0) { //检查眼是否被检测到
        rightEyeRect.x += rightX; // 矫正左眼矩形，因为它从图像的右边界开始
        rightEyeRect.y += topY;  // 矫正右眼矩形，因为人脸边界被去除掉了
        rightEye = Point(rightEyeRect.x + rightEyeRect.width/2, rightEyeRect.y + rightEyeRect.heig
ht/2);
    }
    else {
        rightEye = Point(-1, -1);    // 返回一个无效的点
    }
}
```

运行效果图：

1、



2、检测到的人脸矩形区域：



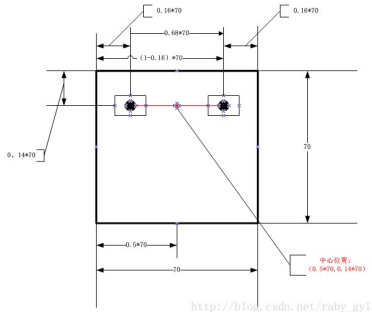
3、人脸矫正和尺寸归一化到70*70后的结果图：

我要说的：

1、代码是截取的原文中的一小部分，搭配好环境可以直接运行，人家的程序可能适用于网络摄像头拍的正对着人脸的，一个人脸图像。而不是针对一般的有一群人，人脸小一些的，或者人脸不是正面的图像，你可以跟lena图像试一下，它只能检测到一只左眼（真实的右眼），而另外一只检测不到，那么就会返回一个无效的点Point(-1,-1)作为眼睛的中心，那么更别提后面的旋转了，即后面的旋转肯定也是不对的。在你用本程序测试的时候，一定要选择一个合理的图像。

2、我讲一下关于旋转平移的代码的理解：

首先我们看下图：



这是我们的目的图像，满足的要求为:

(1)大小为70*70;

(2)两个眼睛直接的距离为（1-0.16）*70;（图中的（0.16，0.14）是左眼中心在图像中比例位置，由于人眼时对称的，则右眼所在比例位置为（0.68，0.14），要想获得真实的位置乘以70即可，对于本例是这样的）

(3)两个人眼连线的中心位置；

有了上述三个条件，我们就可以固定一双人眼在一个大小（这里是70*70）固定的图像中的具体位置。下面的代码就是要实现这个功能：

```
Point2f eyesCenter;//原图像两个眼睛连续的中心点（参考下图参）
    eyesCenter.x=(leftEye.x+rightEye.x)*0.5f;
    eyesCenter.y=(leftEye.y+rightEye.y)*0.5f;
double dy=(rightEye.y-leftEye.y);
    double dx=(rightEye.x-leftEye.x);
    double len=sqrt(dx*dx+dy*dy);//原图像两个眼睛之间的距离
double angle=atan2(dy,dx)*180.0/CV_PI;//计算出来的旋转角度
```

```
//目标图像的位置
const double DESIRED_RIGHT_EYE_X=1.0f-0.16;
const int DESIRED_FACE_WIDTH=70;
    const int DESIRED_FACE_HEIGHT=70;
    double desiredLen=(DESIRED_RIGHT_EYE_X-0.16);//目标图像两个眼睛直接的比例距离，乘以WIDTH=70即得到距离
double scale=desiredLen*DESIRED_FACE_WIDTH/len;//通过目的图像两个眼睛距离除以原图像两个眼睛的距离，得到旋转矩阵的尺度因子。
```

```
Mat rot_mat = getRotationMatrix2D(eyesCenter, angle, scale);//绕原图像两眼连线中心点旋转，旋转角度为angle，缩放尺度为scale
```

```
//难点部分理解，实现中心点的平移，来控制两个眼睛在图像中的位置：
        double ex=DESIRED_FACE_WIDTH * 0.5f - eyesCenter.x;//获取x方向的平移因子，即目标两眼连线中心点的x坐标–原图像两眼连线中心点x坐标
double ey = DESIRED_FACE_HEIGHT * DESIRED_LEFT_EYE_Y-eyesCenter.y;//获取x方向的平移因子，即目标两眼连线中心点的x坐标–原图像两眼连线中心点x坐标
    rot_mat.at<double>(0, 2) += ex;//将上述结果加到旋转矩阵中控制x平移的位置
    rot_mat.at<double>(1, 2) += ey;//将上述结果加到旋转矩阵中控制y平移的位置
    Mat warped = Mat(DESIRED_FACE_HEIGHT, DESIRED_FACE_WIDTH,CV_8U, Scalar(128));
    warpAffine(img_rect, warped, rot_mat, warped.size());
```

我们可以假想一下，上边的代码如果没有最后一个中心点的平移，之前的旋转矩阵只能控制图像的缩放和两个眼睛直接的相对位置，但是控制不了两个眼睛在图像中的位置，即固定两个眼在图像中的位置。

补充知识(仿射变换)：

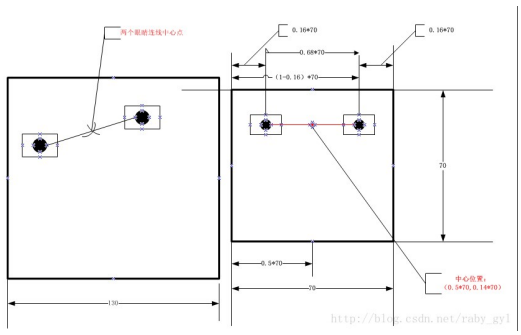
$$\begin{cases} x' = ax + by + m \\ y' = cx + dy + n \end{cases}$$

也可以用矩阵的形式表示如下：

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

上图中a0,b0为控制平移的因子，如果我们领a2=1,a1=0,b2=0,b1=1，即变为u=x+a0,v=y+b0;

参考图：



人脸规范化 face normalization 07-05
将扫描的人脸或其他的 规范化 如 电子护照的水平 人脸规范化 face normalization人脸规范化 face normalization人脸规范化 face normalization

人脸矫正 人脸转正 jacke121的专栏 4722
人脸对齐：是找到鼻子的特征点在哪，眼睛的特征点在哪，能矫正 也可以用来换脸。 https://github.com/YadiraF/PRNet 是在3ddfa基础上改进的： https:/…

评论 11 您还未登录，请先 登录 后发表或查看评论

人脸识别之人眼定位、人脸矫正、人脸尺寸标准化—<Mas... 2-11
我感觉下面的程序——对人眼定位,人脸矫正,人脸尺寸化,对于初学人脸识别,做人脸的预处理非常有帮助~ 程序的思路是:首先通过人脸检测级联器检测到人…

人脸矫正之人眼检测实例(Android) - CSDN博客 9-8
人脸识别之人眼定位、人脸矫正、人脸尺寸标准化---<Mastering OpenCV with Practical Computer Vision Projects > 8332 代码来源于我感觉下面的程序…

人脸识别中的图像形变矫正 11-09
图像形变矫正.vcproj 这是使用应用程序向导生成的 VC++ 项目的主项目文件。 它包含生成该文件的 Visual C++ 的版本信息，以及有关使用应用程序向导…

openCV学习笔记（十） —— 人脸检测之面部特征（眼睛、鼻子、嘴巴） 热门推荐 天空的博客 3万+
环境：VS2017 版本：3.4.1 人脸面部特征检测的准确性依赖于学习到的分类器。 分类器可以从以下地址下载：https://github.com/opencv/opencv_contri…

人脸检测(detection)与人脸校准(alignment) - dwj63367... 8-28
人脸识别之人眼定位、人脸矫正、人脸尺寸标准化---<Mastering OpenCV with Practical Computer Vision Projects > 阅读数 1万+ 代码来源于我感觉下面…

计算机视觉实验四 人脸面部识别_小西几的博客-CSDN博... 2-6
在检测到人脸并定位面部关键特征点之后,主要的人脸区域就可以被裁剪出来,经过预处理之后,输入后端的识别算法。识别算法要完成人脸特征的提取,并与…

人脸检测中几种框框大小的选择~ love模式识别 1万+
人脸检测应用极为广泛，内部细节也偏多，尤其是涉及到几种类型的框，这几种框的大小之前有着千丝万缕的联系，对检测性能的好坏影响程度大小不一…

Android camera2 实现估算眼睛到屏幕的距离 人脸识别 护眼宝验室 soso密斯密斯的博客 2351
记一次不成熟功能的开发记录 主要使用的API camera2 FaceDetector 一、实现思路： 1.首先用camera2打开前置摄像头获取前置摄像头的照片 2.通过Fac…

人脸检测——矫正人脸生成标签_MachineLP的专栏 2-6
人脸检测——矫正人脸生成标签 直接代码: importglob importrandom size_list = (12,24,48) forsizeinsize_list: lines = [] forlinrange(45): dir_path ="data_pr…

人脸识别眼睛定位算法_永无止境的小白_人脸定位算法 2-9
人脸识别眼睛定位算法 眼睛定位 根据眼睛定位的流程,一般分为以下几步:眼睛区域初定位,噪声去除,眼睛的精确定位。 1.眼睛区域初定位,对正面人脸还是…

基于深度模型的人脸对齐和姿态标准化 gzq0723的博客 1587
美好世界Wonderful world相隔41天，Edison又回来了，由于自己和团队的事情比较多，没有在我们的"计算机视觉战队"平台花费过多的精力，今天我来和…

OpenCV objdetect人脸识别例子校正 cxzq2010的专栏 716
http://www.opencv.org.cn/opencvdoc/2.3.2/html/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html#cascade-classifier 一文中 的代码在…

人脸识别中的人脸监测、定位校准和对比等技术_喜欢打酱... 2-24
随着大数据时代的到来,"人脸"也将成为数据的一部分,人脸识别如何实现?本文将为大家从人脸检测、人脸定位、人脸校准以及人脸对比等方面详细阐述人…

人脸检测——Faster R-CNN - zyazky - CSDN博客 10-27
人脸识别之人眼定位、人脸矫正、人脸尺寸标准化---<Mastering OpenCV with Practical Computer Vision Projects > 05-05 8666 代码来源于我感觉下面的…

特征工程 Major_S的博客 148
特征工程特征工程特征工程

python opencv 人脸识别 纠偏_手把手教你如何用 OpenCV + Python 实现人脸识别-阿里云开发者社区... weixin_32026658的博客 44
下午的时候，配好了OpenCV的Python环境，OpenCV的Python环境搭建。于是迫不及待的想体验一下opencv的人脸识别，如下文。必备知识Haar-likeHa…

机器学习(8)-人脸识别和 人脸定位_Tas的博客_人脸定位 2-24
- 正样本采样:即人脸的部分，需要把图片中人脸的部分裁剪出来,要注意的是,裁剪出来后的图片要人工过一遍,数据的好坏对训练的结果影响很大。 - 负样本:…

人脸识别之人脸对齐(一)–定义及作用 - CSDN博客 12-7
人脸对齐任务即根据输入的人脸图像,自动定位出面部关键特征点,如眼睛、鼻尖、嘴角点、眉毛以及人脸各部件轮廓点等,如下图所示。 这项技术的应用很…

OpenCV人脸识别实验（一）——特征脸（Eigenfaces）及其重构的源代码详解 loveliuzz的博客 2万+
1、介绍Introduction 从OpenCV2.4开始，加入了新的类FaceRecognizer，我们可以使用它便捷地进行人脸识别实验。 本实验采用的编程环境为：opencv…

eastlhu
码龄11年 暂无认证
96 原创 | 10万+ 周排名 | 144万+ 总排名 | 64万+ 访问 | 等级
6811 积分 | 130 粉丝 | 89 获赞 | 73 评论 | 203 收藏
私信 关注

解决Mac下Unity启动黑屏

juicewall: 仍然没有解决 mac 10.15 版本装unity5.6.7 启动还是黑

Mac OS 任意显示器 开启HiDPI方法

iDevelop2020: 能否把最后的《DisplayProductID-YYYY》文件贴出来，这个文件需...

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

Objective-C中的instancetype和id关键字

Objective-C中的多态性分析

解决Mac下Unity启动黑屏

2018年 11篇　2017年 12篇
2016年 38篇　2015年 36篇
2014年 158篇　2013年 32篇
2012年 35篇

人脸识别像素最低_人脸识别的技术要求　最新发布　weixin_39623273的博客　4788
技术标准要求GA/T893-2010 安防生物特征识别应用术语规定了常见的安防生物特征识别应用常见的术语，其中包括人脸识别技术常用术语。GB/T 35678...

ResNet：用dlib实现人脸识别—C++，包括人脸检测和人脸矫正(附代码)　意疏的学习笔记　7137
前言 前段时间用dlib包做了一下人脸识别设计，这里做一个简单的笔记。设计实现了，人脸的检测，人脸的截取，人脸的矫正和人脸的识别。通过命令行...

使用opencv和dlib进行人脸姿态估计(python)　yuanlulu的博客　1万+
概述 在计算机视觉中，物体的姿态是指相对于相机的相对取向和位置。 本文主要参考了《Head Pose Estimation using OpenCV and Dlib》这篇文章。进...

07-人脸识别-人脸矫正　1103
人脸矫正有几个问题。 1.歪头： 2.侧脸： 3.半边脸：缺失另外半边脸，要寻找其他的解决方案。 大多数情况下，截取到的人脸是包含歪头和侧脸的现象...

简易人脸分割　Junzzy的博客　7914
很多时候，我们需要对脸部位置进行提取以达到某种定位的需求。 这时候就需要对脸部进行分割处理。 脸部的分割算法有好多种，识别的算法更是一片浩...

【OpenCV+Dlib】C++实现人脸几何矫正（保持头部水平且大小固定）原理+源码　Feeryman_Lee的博客　986
在图像实际应用过程中，在很多情况下我们需要截取下人脸的某些位置，以眼睛为例，在视频流的情况下，随着人头部的晃动...与相机的距离的变化，眼...

人脸识别之仿射变换　746
对于检测到人脸经行矫正，用于下一阶段网络的输入 import numpy as np import cv2 class FaceWarpException(Exception): ...'In F...

眼睛定位　2016
转自：http://www.cnblogs.com/ImageVision/archive/2012/03/14/2396534.html 根据眼睛定位的流程，一般分为以下几步：眼...