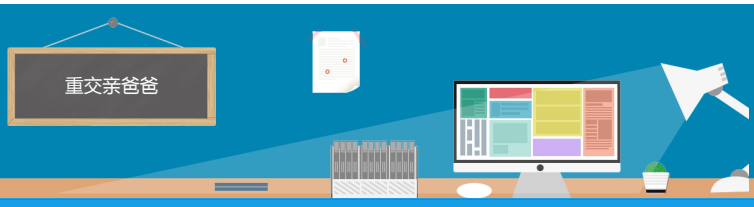


基于opencv3.0和zbar下条形码和二维码的识别与解码

其中对条码与二维码的识别分为以下4个步骤

1. 利用opencv和Zbar（或者Zxing）对标准的条形码图片（即没有多余背景干扰，且图片没有倾斜）进行解码，将解码信息显示出来，并与原始信息对比。

2. 利用opencv和Zbar（或者Zxing）对标准的QR二维码图片（即没有多余背景干扰，且图片没有倾斜）进行解码，将解码信息显示出来，并与原始信息对比。

3. 对非标准条形码，进行定位，然后用Zbar（或者Zxing）解码显示。

4. 对非标准的QR二维码图片，进行定位，然后用Zbar（或者Zxing）解码显示。

1. 利用opencv和Zbar（或者Zxing）对标准的条形码图片（即没有多余背景干扰，且图片没有倾斜）进行解码，将解码信息显示出来，并与原始信息对比。

2. 利用opencv和Zbar（或者Zxing）对标准的QR二维码图片（即没有多余背景干扰，且图片没有倾斜）进行解码，将解码信息显示出来，并与原始信息对比。

这两部对于zbar可以一并操作。

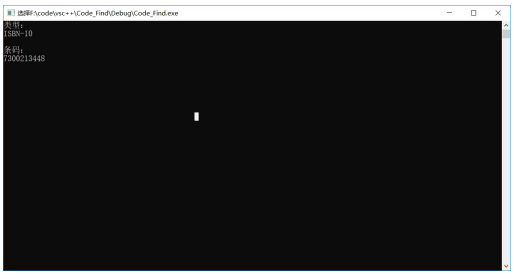
操作步骤主要分为两部分：A.原图进行灰度转化，B.送入Zbar扫描仪进行扫描(调用ImageScanner)

源码如下：

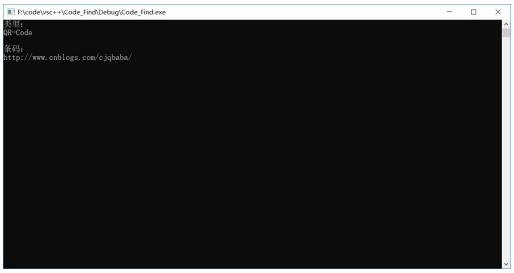
```
/*****************************************************
函数名称：Dis_Barcode
函数功能：识别条形码和二维码
传入参数：
返 回 值：
建立时间：2018-05-19
修改时间：
建 立 人：
修 改 人：
其它说明：这里是借鉴其他人的代码：
原文链接：https://www.cnblogs.com/dengxiaojun/p/5278679.html
以下代码是经过改动的
*****************************************************/
void MyClass::Dis_code(Mat image){
    Mat imageGray;  // 所转化成的灰度图像
    //定义一个扫描仪
    ImageScanner scanner;
    scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);

    cvtColor(image, imageGray, CV_RGB2GRAY);
    imshow("灰度图", imageGray);
    // 获取所提取图像的长和宽
    int width = imageGray.cols;
    int height = imageGray.rows;
    // 在zbar中进行扫描时候，需要将OpenCV中的Mat类型转换为(uchar *)类型，raw中存放的是图像的地址;对应的图像需要转成zbar中对应的图像zbar::Image
    uchar *raw = (uchar *)imageGray.data;
    Image imageZbar(width, height, "Y800", raw, width * height);
    // 扫描相应的图像imageZbar(imageZbar是zbar::Image类型，存储着该入的图像)
    scanner.scan(imageZbar); //扫描条码
    Image::SymbolIterator symbol = imageZbar.symbol_begin();
    if (imageZbar.symbol_begin() == imageZbar.symbol_end())
    {
        cout << "查询条码失败，请检查图片！" << endl;
    }
    for (; symbol != imageZbar.symbol_end(); ++symbol)
    {
        cout << "类型:" << endl << symbol->get_type_name() << endl << endl;
        cout << "条码:" << endl << symbol->get_data() << endl << endl;
    }

    waitKey(); // 等待按下esc键，若需要延时1s则改用waitKey(1000);

    // 将图像中的数据置为0
    imageZbar.set_data(NULL, 0);
    system("pause");
}
```

结果如下：

条形码识别：



二维码识别：



3. 对非标准条形码，进行定位，然后用Zbar（或者Zxing）解码显示

在条形码的识别上，根据条形码的特性，我们只关心x轴上的形态。通过x轴的宽度进行确定条码的大小，y轴根据实际提取进行区分

处理的目标：

A.消去非码的其他物体图形

B.划定条码的范围

C.提取图片的ROI区域（即条码区域）

总体分为：

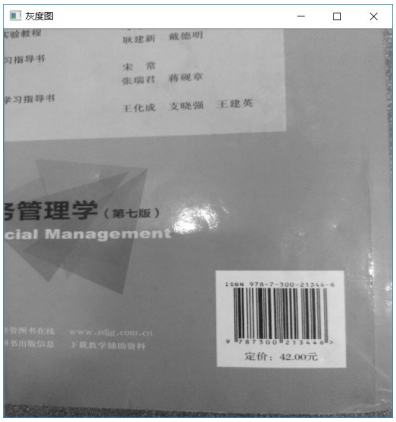
灰度处理-》高斯平滑-》Sobel x—y梯度差-》均值滤波-》二值化-》闭运算-》腐蚀膨胀-》获取ROI

```
/*************************************************
函数名称: Run
函数功能: 开始
传入参数:
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
void MyClass::Run(){
    Mat image;
    image = getGray(srcimage);//获取灰度图
    image = getGass(image);//高斯平滑滤波
    image = getSobel(image);//Sobel x-y梯度差
    image = getBlur(image);//均值滤波除高频噪声
    image = getThold(image);//二值化
    image = getBya(image);//闭运算
    image = getErode(image);//腐蚀
    image = getDilate(image);//膨胀
    image = getRect(image, srcimage);//获取ROI
    imshow("最后的图", image);
    Dis_code(image);
    waitKey();
}
```

灰度处理（消除颜色干扰）

```
/*************************************************
函数名称: getGray
函数功能: 灰度处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
Mat MyClass::getGray(Mat image, bool show){//show默认false 待定参数法
    Mat cimage;
    cvtColor(image, cimage, CV_RGBA2GRAY);
    if (show)
    imshow("灰度图", cimage);
    return cimage;
}
```

处理结果：



高斯滤波处理（消除高斯噪声）

```
/*************************************************
函数名称: getGass
函数功能: 高斯滤波处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
```

if(-1==1){
    show(2);
这个不是很理解

--那兵

2. Re:算法训练 2的次幂表示(蓝桥杯C++写法)

大佬牛批

--那兵

3. Re:基于opencv3.0下的运动车辆检测

良心文章，感谢，学到了。

--penglaixian

4. Re:基于opencv3.0和zbar下条形码和二维码的识别与解码

请教下博主， angle的值应该是在负90到0之间，所以检测条形码这段判断代码有什么用处？新手求教//为了防止找错,要检查这个矩形的偏斜角度不能超标 //如果超标,那就是没找到 if (minRect...

--码农翻身做主人

5. Re:基于opencv3.0下的人脸识别和识别部分的高斯模糊处理

这篇也是，人脸检测被当成人脸识别了

--ChrisZZ

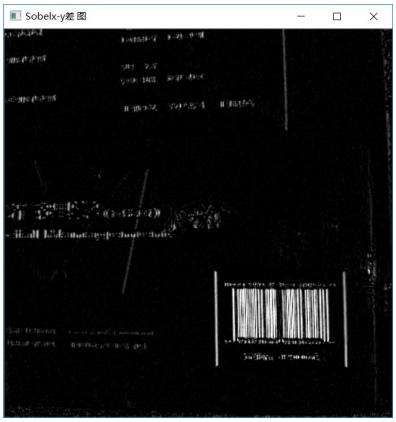
```
Mat MyClass::getGass(Mat image, bool show){
    Mat cimage;
    GaussianBlur(image, cimage, Size(3, 3), 0);
    if (show)
    imshow("高斯滤波图", cimage);
    return cimage;
}
```

处理结果：

Sobel x-y差处理（只考虑x轴，消除y轴不必要信息）

```
/*****************************************************
函数名称: getSobel
函数功能: Sobel处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*****************************************************/
Mat MyClass::getSobel(Mat image, bool show){
    Mat cimageX16s, cimageY16s, imageSobelX, imageSobelY, out;
    Sobel(image, cimageX16s, CV_16S, 1, 0, 3, 1, 0, 4);
    Sobel(image, cimageY16s, CV_16S, 0, 1, 3, 1, 0, 4);
    convertScaleAbs(cimageX16s, imageSobelX, 1, 0);
    convertScaleAbs(cimageY16s, imageSobelY, 1, 0);
    out = imageSobelX - imageSobelY;
    if (show)
    imshow("Sobelx-y差 图", out);
    return out;
}
```

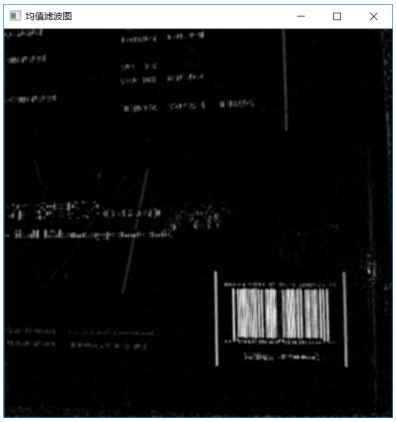
处理结果：



均值滤波处理（消除高频噪声）

```
/*****************************************************
函数名称: getBlur
函数功能: 均值滤波处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*****************************************************/
Mat MyClass::getBlur(Mat image, bool show){
    Mat cimage;
    blur(image, cimage, Size(3, 3));
     if (show)
    imshow("均值滤波图", cimage);
    return cimage;
}
```

处理结果：



二值化处理（使图像中数据量大为减少，从而能凸显出目标的轮廓）

```
/*************************************************
函数名称: getThold
函数功能: 二值化处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
Mat MyClass::getThold(Mat image, bool show){
    Mat cimage;
    threshold(image, cimage, 112, 255, CV_THRESH_BINARY);
    if (show)
    imshow("二值化图", cimage);
    return cimage;
}
```

处理结果：

闭运算处理（扩大轴之间的间隙）

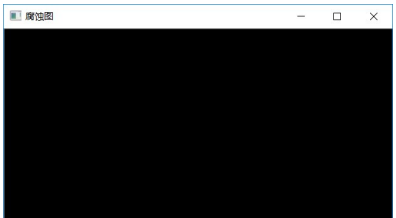
```
/*************************************************
函数名称: getBya
函数功能: 闭运算处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
Mat MyClass::getBya(Mat image, bool show){
    morphologyEx(image, image, MORPH_CLOSE, element);
    if (show)
    imshow("闭运算图", image);
    return image;
}
```

处理结果：

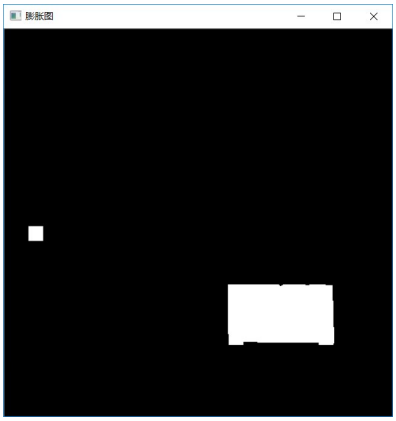
腐蚀膨胀（消去干扰点和合并条码区域）

```
/*************************************************
函数名称: getErode
函数功能: 腐蚀处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
Mat MyClass::getErode(Mat image, bool show){
    //Mat cimage;
    erode(image, image, element);
    if (show)
    imshow("腐蚀图", image);
    return image;
}
/*************************************************
函数名称: getDilate
函数功能: 膨胀处理
传入参数: Mat image
返 回 值:
建立时间: 2018-05-19
修改时间:
建 立 人:
修 改 人:
其它说明:
*************************************************/
Mat MyClass::getDilate(Mat image, bool show){
    for (int i = 0; i < 3; i++)
        dilate(image, image, element);
    if (show)
    imshow("膨胀图", image);
    return image;
}
```

处理结果：



膨胀图

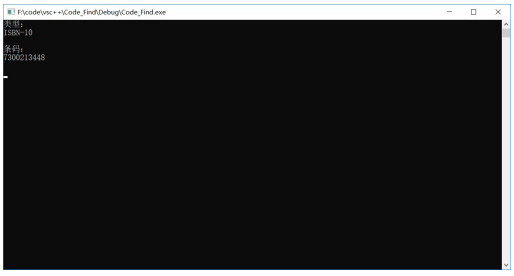


获取ROI（为Zbar处理作预处理）

```
/*************************************************
函数名称：getRect
函数功能：获取码的区域
传入参数：Mat image, Mat simage原图
返 回 值：
建立时间：2018-05-19
修改时间：
建 立 人：
修 改 人：
其它说明：借鉴其他人进行改进
*************************************************/
Mat MyClass::getRect(Mat image, Mat simage, bool show){
    vector<vector<Point>> contours;
    vector<Vec4i> hiera;
    Mat cimage;
    findContours(image, contours, hiera, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
    vector<float>contourArea;
    for (int i = 0; i < contours.size(); i++)
    {
        contourArea.push_back(cv::contourArea(contours[i]));
    }
    //找出面积最大的轮廓
    double maxValue; Point maxLoc;
    minMaxLoc(contourArea, NULL, &maxValue, NULL, &maxLoc);
    //计算面积最大的轮廓的最小的外包矩形
    RotatedRect minRect = minAreaRect(contours[maxLoc.x]);
    //为了防止找错，要检查这个矩形的偏斜角度不能超标
    //如果超标，那就是没找到
    if (minRect.angle<2.0)
    {
        //找到了矩形的角度，但是这是一个旋转矩形，所以还要重新获得一个外包最小矩形
        Rect myRect = boundingRect(contours[maxLoc.x]);
        //把这个矩形在源图像中画出来
        //rectangle(srcImage,myRect,Scalar(0,255,255),3,LINE_AA);
        //看看显示效果，找的对不对
        //imshow(windowNameString,srcImage);
        //将扫描的图像裁剪下来，并保存为相应的结果，保留一些x方向的边界，所以对rect进行一定的扩张
        myRect.x = myRect.x - (myRect.width / 20);
        myRect.width = myRect.width*1.1;
        Mat resultImage = Mat(srcimage, myRect);
        return resultImage;
    }

    for (int i = 0; i<contours.size(); i++)
    {
        Rect rect = boundingRect((Mat)contours[i]);
        //cimage = simage(rect);
        rectangle(simage, rect, Scalar(0), 2);
        if (show)
        imshow("转变图", simage);
    }
    return simage;
}
```

处理结果：

最后识别处理结果：

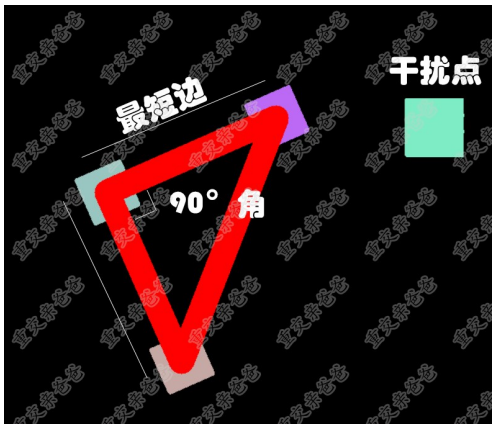


4. 对非标准的QR二维码图片，进行定位，然后用Zbar（或者Zxing）解码显示。

这里主要参考https://blog.csdn.net/nick123chao/article/details/77573675的博客。不过该博客的处理没有考虑多个识别点时的情况：

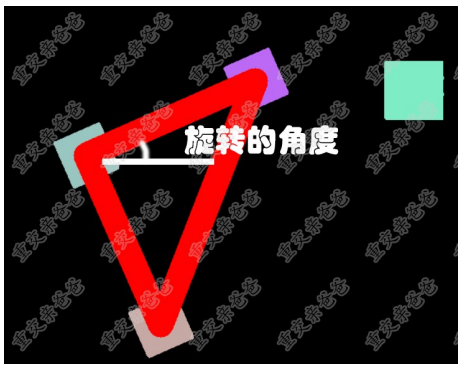
例图：

本文主要处理去除干扰的识别点的方向进行研究解决。根据二维码特性：



我们只要找到90°±Δx的角，且夹角两边为最小的边即可。

找到三个点后，我们需要对齐做旋转处理，旋转的角度如下：



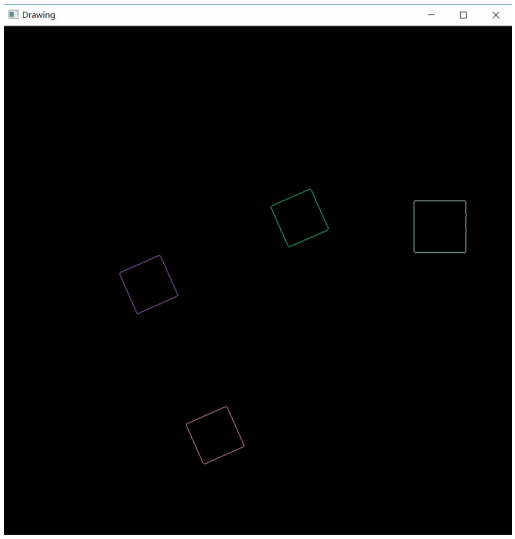
其中处理的步骤分为：

灰度处理-》边缘检测-》特征轮廓检测-》提取特征点-》排除干扰点-》绘制直角三角形-》纠正旋转-》提取ROI-》识别

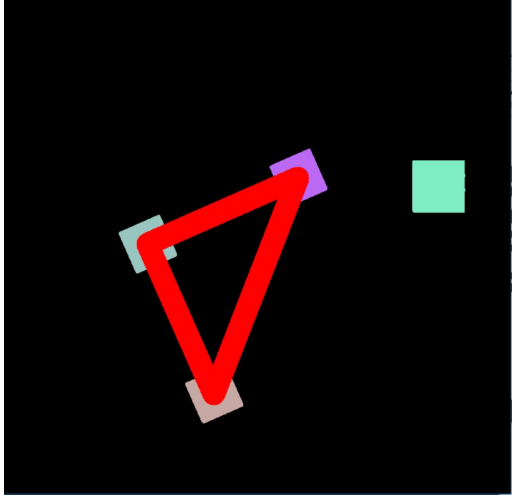
这里先给效果，后展示代码

边缘检测：

特征轮廓检测

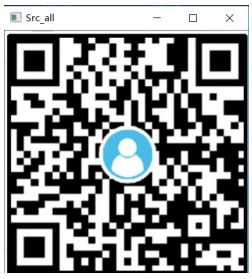


提取特征点-》排除干扰点-》绘制直角三角形



纠正旋转

提取ROI



识别

```
F:\code\vsc++\Code_Find\Debug\Code_Find.exe
类型:
QR-Code

条码:
http://www.cnblogs.com/cjqbaba/
```

源码如下：

```
/*****************************************************
函数名称：QrRun
函数功能：开始
传入参数：
返 回 值：
建立时间：2018-05-19
修改时间：
建 立 人：
修 改 人：
其它说明：
*****************************************************/
void MyClass::QrRun(){
    RNG rng(12345);
    //imshow("原图", srcimage);
    Mat src_all = srcimage.clone();
    Mat src_gray;
    //灰度处理
    src_gray = getBlur(getGray(srcimage));

    Scalar color = Scalar(1, 1, 255);
    Mat threshold_output;
    vector<vector<Point> > contours, contours2;
    vector<Vec4i> hierarchy;
    Mat drawing = Mat::zeros(srcimage.size(), CV_8UC3);
    Mat drawing2 = Mat::zeros(srcimage.size(), CV_8UC3);
    Mat drawingAllContours = Mat::zeros(srcimage.size(), CV_8UC3);

    threshold_output = getThold(src_gray);

    findContours(threshold_output, contours, hierarchy, CV_RETR_TREE, CHAIN_APPROX_NONE, Point(0, 0));

    int c = 0, ic = 0, k = 0, area = 0;
    // 边缘检测
    //通过黑色定位角作为父轮廓，有两个子轮廓的特点，筛选出三个定位角
    int parentIdx = -1;
    for (int i = 0; i< contours.size(); i++)
    {
        //画出所以轮廓图
        drawContours(drawingAllContours, contours, parentIdx, CV_RGB(255, 255, 255), 1, 8);
        if (hierarchy[i][2] != -1 && ic == 0)
        {
            parentIdx = i;
            ic++;
        }
        else if (hierarchy[i][2] != -1)
        {
            ic++;
        }
        else if (hierarchy[i][2] == -1)
        {
```

```
            ic = 0;
            parentIdx = -1;
        }
        //特征轮廓检测 - 》
        //有两个子轮廓
        if (ic >= 2)
        {
            //保存找到的三个黑色定位角
            contours2.push_back(contours[parentIdx]);
            //画出三个黑色定位角的轮廓
            drawContours(drawing, contours, parentIdx, CV_RGB(rng.uniform(0, 255), rng.uniform(0, 255),
rng.uniform(0, 255)), 1, 8);
            ic = 0;
            parentIdx = -1;
        }
    }
    //提取特征点
    //填充的方式画出黑色定位角的轮廓
    for (int i = 0; i<contours2.size(); i++)
        drawContours(drawing2, contours2, i, CV_RGB(rng.uniform(100, 255), rng.uniform(100, 255),
rng.uniform(100, 255)), -1, 4, hierarchy[k][2], 0, Point());

    //获取定位角的中心坐标
    vector<Point> pointfind;
    for (int i = 0; i<contours2.size(); i++)
    {
        pointfind.push_back(Center_cal(contours2, i));
    }
    //排除干扰点
    Mat dst;
    Point point[3]; double angle; Mat rot_mat;
    ///选择合适的点-核心筛选
    if (pointfind.size()>3){
        double lengthA = 10000000000000000, lengthB = 10000000000000000000;
        for (int i = 0; i < pointfind.size(); i++){
            for (int j = 0; j < pointfind.size(); j++){
                for (int k = 0; k < pointfind.size(); k++){
                    if (i != j&&j != k&&i != k){
                        double dxa, dxb,dya,dyb;
                        double k1, k2, wa, wb;
                        dxa = pointfind[i].x - pointfind[j].x;
                        dxb = pointfind[i].x - pointfind[k].x;
                        dya = pointfind[i].y - pointfind[j].y;
                        dyb = pointfind[i].y - pointfind[k].y;
                        if (dxa == 0 || dxb == 0)continue;
                        k1 = dya/dxa;
                        k2 = dyb/dxb ;
                        wa = sqrt(pow(dya, 2) + pow(dya, 2));
                        wb = sqrt(pow(dyb, 2) + pow(dxb, 2));
                        double anglea = abs(atan(k1) * 180 / CV_PI) + abs(atan(k2) * 180 / CV_PI);
                        if (int(anglea)>=85&&int(anglea)<=95&&wa<=lengthA&&wb<=lengthB){
                            lengthA = wa;
                            lengthB = wb;
                            point[0] = pointfind[i];
                            point[1] = pointfind[j];
                            point[2] = pointfind[k];
                        }
                    }
                }
            }
        }
    }
    else{
        for (int i = 0; i < 3; i++){
            point[i] = pointfind[i];
        }
    }
    //绘制直角三角形
    //计算轮廓的面积，计算定位角的面积，从而计算出边长
    area = contourArea(contours2[0]);
    int area_side = cvRound(sqrt(double(area)));
    for (int i = 0; i < 3; i++){
        line(drawing2, point[i], point[(i + 1)%3], color, area_side / 2, 8);
    }

    //纠正旋转
    //判断是否正对
    if (!IsCorrect(point)){
    //进入修正环节
        double angle; Mat rot_mat;
        int start = 0;
        for (int i = 0; i < 3; i++){
            double k1, k2,kk;
            k1 = (point[i].y - point[(i + 1) % 3].y) / (point[i].x - point[(i + 1) % 3].x);
            k2 = (point[i].y - point[(i + 2) % 3].y) / (point[i].x - point[(i + 2) % 3].x);
            kk = k1*k2;
            if (k1*k2 <0)
                start = i;
        }
        double ax, ay, bx, by;
        ax = point[(start + 1) % 3].x;
        ay = point[(start + 1) % 3].y;
        bx = point[(start + 2) % 3].x;
        by = point[(start + 2) % 3].y;
        Point2f center(abs(ax - bx) / 2, abs(ay -by)/ 2);
        double dy = ay - by;
        double dx = ax - bx;
        double k3 = dy / dx;
        angle =atan(k3) * 180 / CV_PI;//转化角度
        rot_mat = getRotationMatrix2D(center, angle, 1.0);

        warpAffine(src_all, dst, rot_mat, src_all.size(), 1, 0, 0);//旋转原图查看
        warpAffine(drawing2, drawing2, rot_mat, src_all.size(), 1, 0, 0);//旋转连线图
        warpAffine(src_all, src_all, rot_mat, src_all.size(), 1, 0, 0);//旋转原图

        namedWindow("Dst");
        imshow("Dst", dst);
    }

    namedWindow("DrawingAllContours");
    imshow("DrawingAllContours", drawingAllContours);

    namedWindow("Drawing2");
    imshow("Drawing2", drawing2);

    namedWindow("Drawing");
    imshow("Drawing", drawing);

    //提取ROI
    //接下来要框出这整个二维码
    Mat gray_all, threshold_output_all;
    vector<vector<Point> > contours_all;
    vector<Vec4i> hierarchy_all;
    cvtColor(drawing2, gray_all, CV_BGR2GRAY);


    threshold(gray_all, threshold_output_all, 45, 255, THRESH_BINARY);
    findContours(threshold_output_all, contours_all, hierarchy_all, RETR_EXTERNAL, CHAIN_APPROX_NONE, Point(0,
0));//RETR_EXTERNAL表示只寻找最外层轮廓


    Point2f fourPoint2f[4];
    //求最小包围矩形
    RotatedRect rectPoint = minAreaRect(contours_all[1]);//pointfind.size()-3

    //将rectPoint变量中存储的坐标值放到 fourPoint的数组中
    rectPoint.points(fourPoint2f);

    int maxx = 0, maxy = 0, minx = 100000, miny = 100000;
    for (int i = 0; i < 4; i++)
    {
        if (maxx < fourPoint2f[i].x)maxx = fourPoint2f[i].x;
        if (maxy < fourPoint2f[i].y)maxy = fourPoint2f[i].y;
        if (minx > fourPoint2f[i].x)minx = fourPoint2f[i].x;
        if (miny > fourPoint2f[i].y)miny = fourPoint2f[i].y;
        line(src_all, fourPoint2f[i % 4], fourPoint2f[(i + 1) % 4]
            , Scalar(0), 3);
    }
    namedWindow("Src_all");
    ///边际处理
    int set_inter = 5;
    while (true)
    {
        minx -= set_inter;
        miny -= set_inter;
        maxx += set_inter;
        maxy += set_inter;
        if (maxx > srcimage.size().width || maxy > srcimage.size().height || minx < 0 || miny < 0){
            minx += set_inter;
            miny += set_inter;
            maxx -= set_inter;
            maxy -= set_inter;
            set_inter--;
        }
        else
        {
            break;
        }
    }
    imshow("Src_all", src_all(Rect(minx, miny, maxx - minx, maxy - miny)));//ROI
    Mat fout = src_all(Rect(minx, miny, maxx - minx, maxy - miny));//ROI

```

```
    //识别
    Dis_code(fout);

    waitKey(0);
    destroyAllWindows();
}
```

由于在解码上是采用其他人的方法，存在解码问题。（到时候有机会自己再写下）

ps：目前的zbar不支持中文识别，但是zxing可以。所以借鉴本文的需要改进下识别的模块即可。

本文的不足之处：

这里还做了测试，对于旋转180°以上的二维码图片存在可能无法识别的问题。以及码眼为非正方形的也无法识别。

如需要源码请转移至码云：https://gitee.com/cjqbaba/MediaTest/tree/Code_Find进行源码克隆下载

如有问题请留言评论。转载请注明出处，谢谢。

标签: C++, opencv3.0, 二维码识别, 条形码识别, zbar, ROI, Sobel

好文要顶 关注我 收藏该文

重交亲爸爸
关注 - 0
粉丝 - 6
+加关注

5 推荐 0 反对

« 上一篇： 基于opencv3.0下的运动车辆检测
» 下一篇： kotlin面向对象-笔记

posted @ 2018-05-23 12:00 重交亲爸爸 阅读(19991) 评论(2) 编辑 收藏 举报