[**OPENCV条形码检测与识别**](http://www.cnblogs.com/dengxiaojun/p/5278679.html)

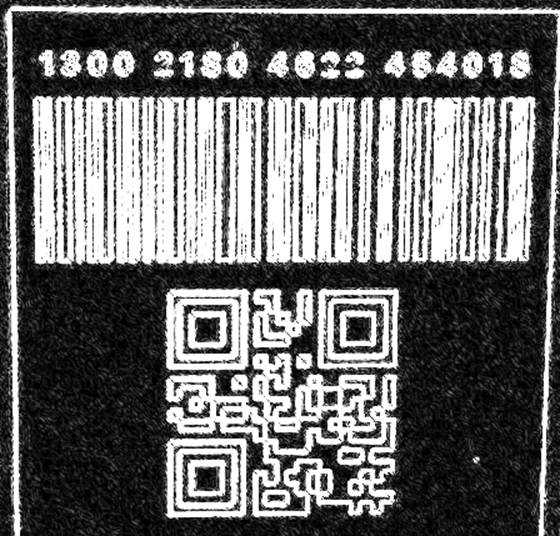
（https://www.cnblogs.com/dengxiaojun/p/5278679.html）

　　条形码是当前超市和部分工厂使用比较普遍的物品,产品标识技术,使用摄像头检测一张图片的条形码包含有两个步骤,第一是定位条形码的位置,定位之后剪切出条形码,并且识别出条形码对应的字符串,然后就可以调用网络,数据库等手段快速进行后续处理.

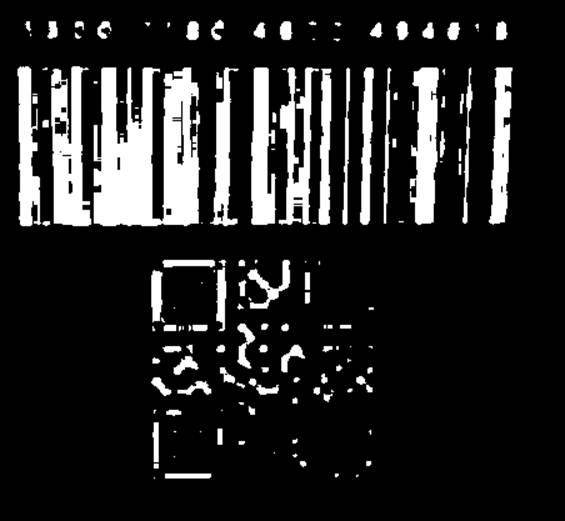
      条形码识别要考虑到条形码的特点,本文针对的是条形码在图片中的位置相对垂直,没有各种倾斜的那种条形码,如下图所示



要定位首先要检视这种条形码的特点,这种图像在X方向上的梯度肯定很明显,同时,Y方向的梯度就没这么明显,所以第一步,我们应该将图像的灰度图像分别计算梯度,用X方向梯度减去Y方向梯度,这样可以保留X方向特征并且去除Y方向的干扰,处理之后图像如下所示



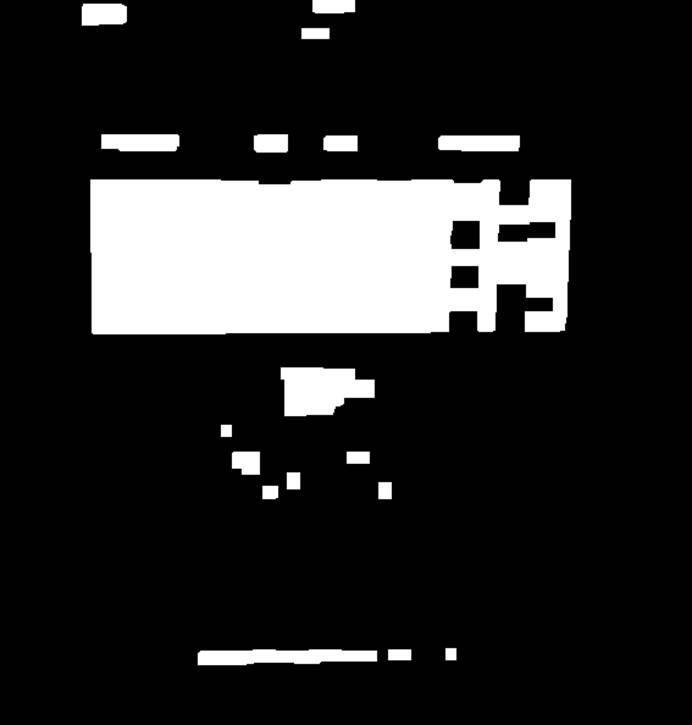
可以看到,二维码对一维码的定位形成了干扰,但是二维码的空间漏洞相对一维码多很懂,于是我们考虑进行一次模糊并且二值化,看能不能有所效果,如下(记得调整相应的模糊化参数和阈值参数,得到相对最好的结果)



有一定的效果,但是此时又出现问题条形码出现了黑色的缝隙,不利于定位完整区域,这个时候要进行一些形态学操作,去除黑色缝隙,我们选择闭运算,算子根据缝隙的情况,宽度大于高度,矩形缝隙.处理以后的结果.

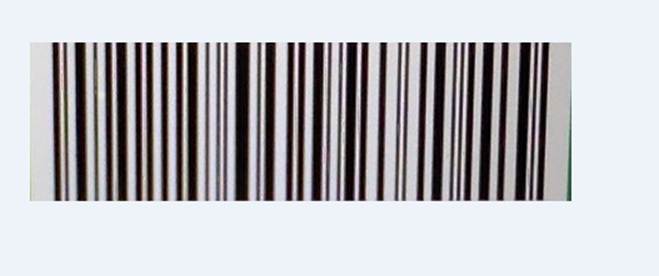


效果可以,又出现问题,二维码的区域连着,还是面积很大,对后面我们算区域面积依然有影响,但是我们观测二维码的连接区域明显要比一维码的连接区域要细很多,也就是说,我们可以很快的腐蚀断二维码的连接,同时还保持一维码的连接,然后在膨胀回来,二维码的连接断开就应该不会有这个大块的区域连着了,注意,膨胀和腐蚀的次数应当是一致的,保证得到结果区域的准确.我选择膨胀腐蚀四次,先膨胀断开二维码连接,最后的结果显示如下



此时,二维码的影响就基本没有了,现在我们只需要先查找轮廓,然后计算图像中每个轮廓的面积,选出面积最大的那个轮廓,计算这个轮廓的最小外包矩形,就能找到相应的图像区域了.这样操作的结果和切分出来的条形码如下所示



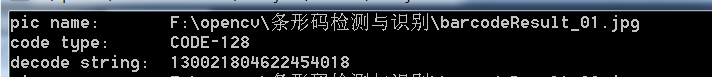


到目前为止,我们已经完成了条形码的位置定位,并且剪切出了条形码的团,接下里对这个图案进行识别,识别之前,总结一下

1. **形态学梯度运算,忽略Y方向梯度,着眼于X方向梯度**
2. **图像模糊化,为了便于后期的图像连接**
3. **图像求阈值,加速算法处理,并合理使用模糊化的效果**
4. **形态学去除黑洞,闭运算**
5. **膨胀腐蚀,断开二维码连接**
6. **查找轮廓,计算轮廓最大面积,拟合轮廓矩形,得到最终结果**

接下来条形码识别,可以使用zbar识别库,库的简介就不说了,可以自己去官网下载,安装时候记得选上第三个选项,否则没有头文件.

安装完成后,到安装目录,将bin目录加入环境变量,在VS中VC++目录的include中加入头文件地址,lib地址,并加入lib名称(连接器-输入-附加依赖项),然后就可以使用了,具体使用查看下面的代码,结果如下



代码如下

[复制代码](javascript:void(0);)

#include <opencv2/opencv.hpp>

#include <iostream>

#include <zbar.h>

using namespace cv;

using namespace std;

using namespace zbar;

int main(int argc,char\* argv[])

{

char fileNameString[100];

char windowNameString[50];

char resultFileNameSring[100];

Mat srcImage,grayImage,blurImage,thresholdImage,gradientXImage,gradientYImage,gradientImage,morphImage;

for (int fileCount = 1;fileCount < 8;fileCount++)

{

sprintf(fileNameString,"F:\\opencv\\条形码检测与识别\\barcode\_0%d.jpg",fileCount);

sprintf(windowNameString,"result 0%d",fileCount);

sprintf(resultFileNameSring,"F:\\opencv\\条形码检测与识别\\barcodeResult\_0%d.jpg",fileCount);

//读取图像

srcImage = imread(fileNameString);

if(srcImage.empty())

{

cout<<"image file read error"<<endl;

return -1;

}

//图像转换为灰度图像

if(srcImage.channels() == 3)

{

cvtColor(srcImage,grayImage,CV\_RGB2GRAY);

}

else

{

grayImage = srcImage.clone();

}

//建立图像的梯度幅值

Scharr(grayImage,gradientXImage,CV\_32F,1,0);

Scharr(grayImage,gradientYImage,CV\_32F,0,1);

//因为我们需要的条形码在需要X方向水平,所以更多的关注X方向的梯度幅值,而省略掉Y方向的梯度幅值

subtract(gradientXImage,gradientYImage,gradientImage);

//归一化为八位图像

convertScaleAbs(gradientImage,gradientImage);

//看看得到的梯度图像是什么样子

//imshow(windowNameString,gradientImage);

//对图片进行相应的模糊化,使一些噪点消除

blur(gradientImage,blurImage,Size(9,9));

//模糊化以后进行阈值化,得到到对应的黑白二值化图像,二值化的阈值可以根据实际情况调整

threshold(blurImage,thresholdImage,210,255,THRESH\_BINARY);

//看看二值化图像

//imshow(windowNameString,thresholdImage);

//二值化以后的图像,条形码之间的黑白没有连接起来,就要进行形态学运算,消除缝隙,相当于小型的黑洞,选择闭运算

//因为是长条之间的缝隙,所以需要选择宽度大于长度

Mat kernel = getStructuringElement(MORPH\_RECT,Size(21,7));

morphologyEx(thresholdImage,morphImage,MORPH\_CLOSE,kernel);

//看看形态学操作以后的图像

//imshow(windowNameString,morphImage);

//现在要让条形码区域连接在一起,所以选择膨胀腐蚀,而且为了保持图形大小基本不变,应该使用相同次数的膨胀腐蚀

//先腐蚀,让其他区域的亮的地方变少最好是消除,然后膨胀回来,消除干扰,迭代次数根据实际情况选择

erode(morphImage, morphImage, getStructuringElement(MORPH\_RECT, Size(3,3)),Point(-1,-1),4);

dilate(morphImage, morphImage, getStructuringElement(MORPH\_RECT, Size(3,3)),Point(-1,-1),4);

//看看形态学操作以后的图像

//imshow(windowNameString,morphImage);

vector<vector<Point2i>>contours;

vector<float>contourArea;

//接下来对目标轮廓进行查找,目标是为了计算图像面积

findContours(morphImage,contours,RETR\_EXTERNAL,CHAIN\_APPROX\_SIMPLE);

//计算轮廓的面积并且存放

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);

if(!imwrite(resultFileNameSring,resultImage))

{

cout<<"file save error!"<<endl;

return -2;

}

}

}

//检测到了之后进行条形码识别

FileStorage file("F:\\opencv\\条形码检测与识别\\result.xml",FileStorage::WRITE);

for (int fileCount = 1;fileCount < 8;fileCount++)

{

sprintf(resultFileNameSring,"F:\\opencv\\条形码检测与识别\\barcodeResult\_0%d.jpg",fileCount);

sprintf(windowNameString,"result 0%d",fileCount);

Mat result = imread(resultFileNameSring);

if(!result.empty())

{

//现在开始识别

cvtColor(result,grayImage,CV\_RGB2GRAY);

int width = grayImage.cols; // extract dimensions

int height = grayImage.rows;

Image image(width,height,"Y800",grayImage.data,width\*height);

ImageScanner scanner;

scanner.set\_config(ZBAR\_NONE,ZBAR\_CFG\_ENABLE,1);

int n = scanner.scan(image);

for (Image::SymbolIterator symbol = image.symbol\_begin(); symbol != image.symbol\_end();++symbol)

{

cout <<"pic name:\t"<<resultFileNameSring<<endl<<"code type:\t"<<symbol->get\_type\_name()<<endl<<\

"decode string:\t"<<symbol->get\_data()<<endl;

image.set\_data(NULL,0);

//xml文件写入

}

}

}

waitKey(0);

return 1;

}

[复制代码](javascript:void(0);)

资源如下

http://download.csdn.net/detail/dengrengong/9461797