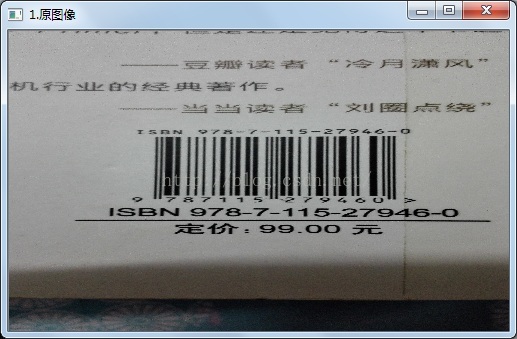
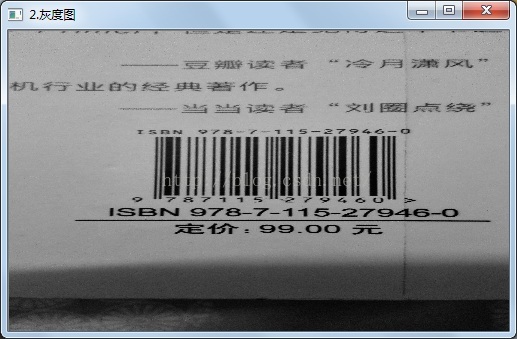
**Opencv：10个步骤检测出图片中条形码**

（https://blog.csdn.net/dcrmg/article/details/52095508）

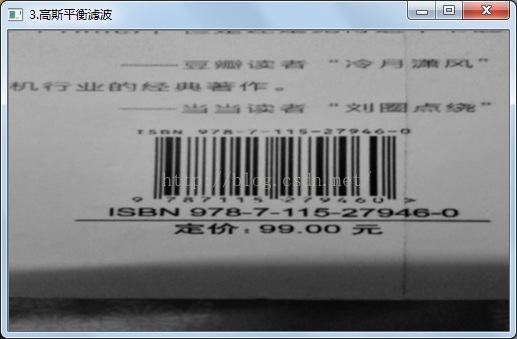
1. 原图像大小调整，提高运算效率



2. 转化为灰度图



3. 高斯平滑滤波



4.求得水平和垂直方向灰度图像的梯度差,使用Sobel算子



5.均值滤波，消除高频噪声



6.二值化



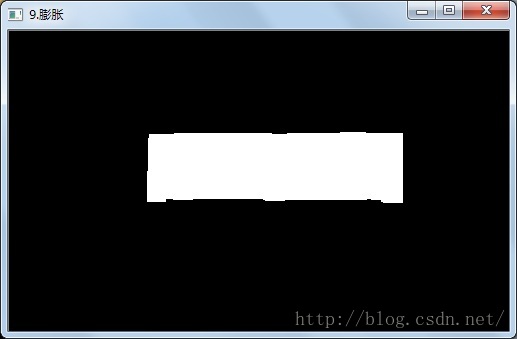
7.闭运算，填充条形码间隙



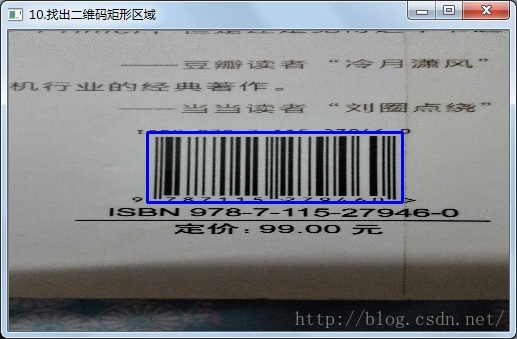
8. 腐蚀，去除孤立的点



9. 膨胀，填充条形码间空隙，根据核的大小，有可能需要2~3次膨胀操作



10.通过findContours找到条形码区域的矩形边界



实现：

**[cpp]** [view plain](https://blog.csdn.net/dcrmg/article/details/52095508) [copy](https://blog.csdn.net/dcrmg/article/details/52095508)

[print?](https://blog.csdn.net/dcrmg/article/details/52095508)

1. #include "core/core.hpp"
2. #include "highgui/highgui.hpp"
3. #include "imgproc/imgproc.hpp"
5. using namespace cv;
7. int main(int argc,char \*argv[])
8. {
9. Mat image,imageGray,imageGuussian;
10. Mat imageSobelX,imageSobelY,imageSobelOut;
11. image=imread(argv[1]);
13. //1. 原图像大小调整，提高运算效率
14. resize(image,image,Size(500,300));
15. imshow("1.原图像",image);
17. //2. 转化为灰度图
18. cvtColor(image,imageGray,CV\_RGB2GRAY);
19. imshow("2.灰度图",imageGray);
21. //3. 高斯平滑滤波
22. GaussianBlur(imageGray,imageGuussian,Size(3,3),0);
23. imshow("3.高斯平衡滤波",imageGuussian);
25. //4.求得水平和垂直方向灰度图像的梯度差,使用Sobel算子
26. Mat imageX16S,imageY16S;
27. Sobel(imageGuussian,imageX16S,CV\_16S,1,0,3,1,0,4);
28. Sobel(imageGuussian,imageY16S,CV\_16S,0,1,3,1,0,4);
29. convertScaleAbs(imageX16S,imageSobelX,1,0);
30. convertScaleAbs(imageY16S,imageSobelY,1,0);
31. imageSobelOut=imageSobelX-imageSobelY;
32. imshow("4.X方向梯度",imageSobelX);
33. imshow("4.Y方向梯度",imageSobelY);
34. imshow("4.XY方向梯度差",imageSobelOut);
36. //5.均值滤波，消除高频噪声
37. blur(imageSobelOut,imageSobelOut,Size(3,3));
38. imshow("5.均值滤波",imageSobelOut);
40. //6.二值化
41. Mat imageSobleOutThreshold;
42. threshold(imageSobelOut,imageSobleOutThreshold,180,255,CV\_THRESH\_BINARY);
43. imshow("6.二值化",imageSobleOutThreshold);
45. //7.闭运算，填充条形码间隙
46. Mat  element=getStructuringElement(0,Size(7,7));
47. morphologyEx(imageSobleOutThreshold,imageSobleOutThreshold,MORPH\_CLOSE,element);
48. imshow("7.闭运算",imageSobleOutThreshold);
50. //8. 腐蚀，去除孤立的点
51. erode(imageSobleOutThreshold,imageSobleOutThreshold,element);
52. imshow("8.腐蚀",imageSobleOutThreshold);
54. //9. 膨胀，填充条形码间空隙，根据核的大小，有可能需要2~3次膨胀操作
55. dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);
56. dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);
57. dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);
58. imshow("9.膨胀",imageSobleOutThreshold);
59. vector<vector<Point>> contours;
60. vector<Vec4i> hiera;
62. //10.通过findContours找到条形码区域的矩形边界
63. findContours(imageSobleOutThreshold,contours,hiera,CV\_RETR\_EXTERNAL,CV\_CHAIN\_APPROX\_NONE);
64. for(int i=0;i<contours.size();i++)
65. {
66. Rect rect=boundingRect((Mat)contours[i]);
67. rectangle(image,rect,Scalar(255),2);
68. }
69. imshow("10.找出二维码矩形区域",image);
71. waitKey();
72. }

#include "core/core.hpp"

#include "highgui/highgui.hpp"

#include "imgproc/imgproc.hpp"

using namespace cv;

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

{

Mat image,imageGray,imageGuussian;

Mat imageSobelX,imageSobelY,imageSobelOut;

image=imread(argv[1]);

//1. 原图像大小调整，提高运算效率

resize(image,image,Size(500,300));

imshow("1.原图像",image);

//2. 转化为灰度图

cvtColor(image,imageGray,CV\_RGB2GRAY);

imshow("2.灰度图",imageGray);

//3. 高斯平滑滤波

GaussianBlur(imageGray,imageGuussian,Size(3,3),0);

imshow("3.高斯平衡滤波",imageGuussian);

//4.求得水平和垂直方向灰度图像的梯度差,使用Sobel算子

Mat imageX16S,imageY16S;

Sobel(imageGuussian,imageX16S,CV\_16S,1,0,3,1,0,4);

Sobel(imageGuussian,imageY16S,CV\_16S,0,1,3,1,0,4);

convertScaleAbs(imageX16S,imageSobelX,1,0);

convertScaleAbs(imageY16S,imageSobelY,1,0);

imageSobelOut=imageSobelX-imageSobelY;

imshow("4.X方向梯度",imageSobelX);

imshow("4.Y方向梯度",imageSobelY);

imshow("4.XY方向梯度差",imageSobelOut);

//5.均值滤波，消除高频噪声

blur(imageSobelOut,imageSobelOut,Size(3,3));

imshow("5.均值滤波",imageSobelOut);

//6.二值化

Mat imageSobleOutThreshold;

threshold(imageSobelOut,imageSobleOutThreshold,180,255,CV\_THRESH\_BINARY);

imshow("6.二值化",imageSobleOutThreshold);

//7.闭运算，填充条形码间隙

Mat element=getStructuringElement(0,Size(7,7));

morphologyEx(imageSobleOutThreshold,imageSobleOutThreshold,MORPH\_CLOSE,element);

imshow("7.闭运算",imageSobleOutThreshold);

//8. 腐蚀，去除孤立的点

erode(imageSobleOutThreshold,imageSobleOutThreshold,element);

imshow("8.腐蚀",imageSobleOutThreshold);

//9. 膨胀，填充条形码间空隙，根据核的大小，有可能需要2~3次膨胀操作

dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);

dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);

dilate(imageSobleOutThreshold,imageSobleOutThreshold,element);

imshow("9.膨胀",imageSobleOutThreshold);

vector<vector<Point>> contours;

vector<Vec4i> hiera;

//10.通过findContours找到条形码区域的矩形边界

findContours(imageSobleOutThreshold,contours,hiera,CV\_RETR\_EXTERNAL,CV\_CHAIN\_APPROX\_NONE);

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

{

Rect rect=boundingRect((Mat)contours[i]);

rectangle(image,rect,Scalar(255),2);

}

imshow("10.找出二维码矩形区域",image);

waitKey();

}

使用另一幅图片的效果如下：



底部的二维码左侧边界定位错位，检测发现在二值化的时候左侧第二个条码部分被归零了，导致在之后的腐蚀操作中被腐蚀掉了。调整阈值分界值180到160，重新运行正确：



版权声明：本文为博主原创文章，转载请注明出处。 https://blog.csdn.net/dcrmg/article/details/52095508

文章标签