**opencv二维码识别解码**

（https://blog.csdn.net/nick123chao/article/details/77573675）

阅读数：2241

**目的：**

使用opencv库识别QR二维码，框出图片中的二维码，并使用开源库Zxing解码，在这过程中学习理解opencv库相应的函数。

**环境：**

1. window7系统

      2. QT create

**1.准备**

首先安装QT和QT create开发环境，window下使用cmake编译opencv生产lib库。这部分内容网上很多资料，直接搜索安装就行。

**2.工程代码**

QT create配置使用opencv库。在工程的pro文件下面添加指定opencv库的头文件和lib的路径。如下：

**[cpp]** [view plain](https://blog.csdn.net/nick123chao/article/details/77573675) [copy](https://blog.csdn.net/nick123chao/article/details/77573675)

[print?](https://blog.csdn.net/nick123chao/article/details/77573675)

1. <span style="font-size:14px;">INCLUDEPATH+=C:\Qt\opencv\include\opencv\
2. C:\Qt\opencv\include\opencv2\
3. C:\Qt\opencv\include
5. LIBS+=C:\Qt\opencv\lib\libopencv\_calib3d320.dll.a\
6. C:\Qt\opencv\lib\libopencv\_core320.dll.a\
7. C:\Qt\opencv\lib\libopencv\_features2d320.dll.a\
8. C:\Qt\opencv\lib\libopencv\_flann320.dll.a\
9. C:\Qt\opencv\lib\libopencv\_highgui320.dll.a\
10. C:\Qt\opencv\lib\libopencv\_imgcodecs320.dll.a\
11. C:\Qt\opencv\lib\libopencv\_imgproc320.dll.a\
12. C:\Qt\opencv\lib\libopencv\_ml320.dll.a\
13. C:\Qt\opencv\lib\libopencv\_objdetect320.dll.a\
14. C:\Qt\opencv\lib\libopencv\_photo320.dll.a\
15. C:\Qt\opencv\lib\libopencv\_shape320.dll.a\
16. C:\Qt\opencv\lib\libopencv\_stitching320.dll.a\
17. C:\Qt\opencv\lib\libopencv\_superres320.dll.a\
18. C:\Qt\opencv\lib\libopencv\_video320.dll.a\
19. C:\Qt\opencv\lib\libopencv\_videoio320.dll.a\
20. C:\Qt\opencv\lib\libopencv\_videostab320.dll.a</span>

INCLUDEPATH+=C:\Qt\opencv\include\opencv\

C:\Qt\opencv\include\opencv2\

C:\Qt\opencv\include

LIBS+=C:\Qt\opencv\lib\libopencv\_calib3d320.dll.a\

C:\Qt\opencv\lib\libopencv\_core320.dll.a\

C:\Qt\opencv\lib\libopencv\_features2d320.dll.a\

C:\Qt\opencv\lib\libopencv\_flann320.dll.a\

C:\Qt\opencv\lib\libopencv\_highgui320.dll.a\

C:\Qt\opencv\lib\libopencv\_imgcodecs320.dll.a\

C:\Qt\opencv\lib\libopencv\_imgproc320.dll.a\

C:\Qt\opencv\lib\libopencv\_ml320.dll.a\

C:\Qt\opencv\lib\libopencv\_objdetect320.dll.a\

C:\Qt\opencv\lib\libopencv\_photo320.dll.a\

C:\Qt\opencv\lib\libopencv\_shape320.dll.a\

C:\Qt\opencv\lib\libopencv\_stitching320.dll.a\

C:\Qt\opencv\lib\libopencv\_superres320.dll.a\

C:\Qt\opencv\lib\libopencv\_video320.dll.a\

C:\Qt\opencv\lib\libopencv\_videoio320.dll.a\

C:\Qt\opencv\lib\libopencv\_videostab320.dll.a

opencv识别定位二维码的代码如下：

**[cpp]** [view plain](https://blog.csdn.net/nick123chao/article/details/77573675) [copy](https://blog.csdn.net/nick123chao/article/details/77573675)

[print?](https://blog.csdn.net/nick123chao/article/details/77573675)

1. <span style="font-size:14px;">#include "opencv2/highgui/highgui.hpp"
2. #include "opencv2/imgproc/imgproc.hpp"
3. #include <iostream>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <math.h>
8. #include <QDebug>
10. using namespace cv;
11. using namespace std;

14. Mat src; Mat src\_gray;

17. RNG rng(12345);
18. //Scalar colorful = CV\_RGB(rng.uniform(0,255),rng.uniform(0,255),rng.uniform(0,255));
20. //获取轮廓的中心点
21. Point Center\_cal(vector<vector<Point> > contours,int i)
22. {
23. int centerx=0,centery=0,n=contours[i].size();
24. //在提取的小正方形的边界上每隔周长个像素提取一个点的坐标，
25. //求所提取四个点的平均坐标（即为小正方形的大致中心）
26. centerx = (contours[i][n/4].x + contours[i][n\*2/4].x + contours[i][3\*n/4].x + contours[i][n-1].x)/4;
27. centery = (contours[i][n/4].y + contours[i][n\*2/4].y + contours[i][3\*n/4].y + contours[i][n-1].y)/4;
28. Point point1=Point(centerx,centery);
29. return point1;
30. }

33. int main( int argc, char\*\* argv[] )
34. {
35. src = imread( "core.jpg", 1 );
36. Mat src\_all=src.clone();

39. //彩色图转灰度图
40. cvtColor( src, src\_gray, CV\_BGR2GRAY );
41. //对图像进行平滑处理
42. blur( src\_gray, src\_gray, Size(3,3) );
43. //使灰度图象直方图均衡化
44. equalizeHist( src\_gray, src\_gray );
45. namedWindow("src\_gray");
46. imshow("src\_gray",src\_gray);

49. Scalar color = Scalar(1,1,255 );
50. Mat threshold\_output;
51. vector<vector<Point> > contours,contours2;
52. vector<Vec4i> hierarchy;
53. Mat drawing = Mat::zeros( src.size(), CV\_8UC3 );
54. Mat drawing2 = Mat::zeros( src.size(), CV\_8UC3 );
55. Mat drawingAllContours = Mat::zeros( src.size(), CV\_8UC3 );
57. //指定112阀值进行二值化
58. threshold( src\_gray, threshold\_output, 112, 255, THRESH\_BINARY );
60. namedWindow("Threshold\_output");
61. imshow("Threshold\_output",threshold\_output);

64. /\*查找轮廓
65. \*  参数说明
66. 输入图像image必须为一个2值单通道图像
67. contours参数为检测的轮廓数组，每一个轮廓用一个point类型的vector表示
68. hiararchy参数和轮廓个数相同，每个轮廓contours[ i ]对应4个hierarchy元素hierarchy[ i ][ 0 ] ~hierarchy[ i ][ 3 ]，
69. 分别表示后一个轮廓、前一个轮廓、父轮廓、内嵌轮廓的索引编号，如果没有对应项，该值设置为负数。
70. mode表示轮廓的检索模式
71. CV\_RETR\_EXTERNAL 表示只检测外轮廓
72. CV\_RETR\_LIST 检测的轮廓不建立等级关系
73. CV\_RETR\_CCOMP 建立两个等级的轮廓，上面的一层为外边界，里面的一层为内孔的边界信息。如果内孔内还有一个连通物体，这个物体的边界也在顶层。
74. CV\_RETR\_TREE 建立一个等级树结构的轮廓。具体参考contours.c这个demo
75. method为轮廓的近似办法
76. CV\_CHAIN\_APPROX\_NONE 存储所有的轮廓点，相邻的两个点的像素位置差不超过1，即max（abs（x1-x2），abs（y2-y1））==1
77. CV\_CHAIN\_APPROX\_SIMPLE 压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标，例如一个矩形轮廓只需4个点来保存轮廓信息
78. CV\_CHAIN\_APPROX\_TC89\_L1，CV\_CHAIN\_APPROX\_TC89\_KCOS 使用teh-Chinl chain 近似算法
79. offset表示代表轮廓点的偏移量，可以设置为任意值。对ROI图像中找出的轮廓，并要在整个图像中进行分析时，这个参数还是很有用的。
80. \*/
81. findContours( threshold\_output, contours, hierarchy,  CV\_RETR\_TREE, CHAIN\_APPROX\_NONE, Point(0, 0) );
83. int c=0,ic=0,k=0,area=0;
85. //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
86. int parentIdx=-1;
87. for( int i = 0; i< contours.size(); i++ )
88. {
89. //画出所以轮廓图
90. drawContours( drawingAllContours, contours, parentIdx,  CV\_RGB(255,255,255) , 1, 8);
91. if (hierarchy[i][2] != -1 && ic==0)
92. {
93. parentIdx = i;
94. ic++;
95. }
96. else if (hierarchy[i][2] != -1)
97. {
98. ic++;
99. }
100. else if(hierarchy[i][2] == -1)
101. {
102. ic = 0;
103. parentIdx = -1;
104. }
106. //有两个子轮廓
107. if ( ic >= 2)
108. {
109. //保存找到的三个黑色定位角
110. contours2.push\_back(contours[parentIdx]);
111. //画出三个黑色定位角的轮廓
112. drawContours( drawing, contours, parentIdx,  CV\_RGB(rng.uniform(0,255),rng.uniform(0,255),rng.uniform(0,255)) , 1, 8);
113. ic = 0;
114. parentIdx = -1;
115. }
116. }
118. //填充的方式画出三个黑色定位角的轮廓
119. for(int i=0; i<contours2.size(); i++)
120. 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() );
122. //获取三个定位角的中心坐标
123. Point point[3];
124. for(int i=0; i<contours2.size(); i++)
125. {
126. point[i] = Center\_cal( contours2, i );
127. }
129. //计算轮廓的面积，计算定位角的面积，从而计算出边长
130. area = contourArea(contours2[1]);
131. int area\_side = cvRound (sqrt (double(area)));
132. for(int i=0; i<contours2.size(); i++)
133. {
134. //画出三个定位角的中心连线
135. line(drawing2,point[i%contours2.size()],point[(i+1)%contours2.size()],color,area\_side/2,8);
136. }
138. namedWindow("DrawingAllContours");
139. imshow( "DrawingAllContours", drawingAllContours );
141. namedWindow("Drawing2");
142. imshow( "Drawing2", drawing2 );
144. namedWindow("Drawing");
145. imshow( "Drawing", drawing );

148. //接下来要框出这整个二维码
149. Mat gray\_all,threshold\_output\_all;
150. vector<vector<Point> > contours\_all;
151. vector<Vec4i> hierarchy\_all;
152. cvtColor( drawing2, gray\_all, CV\_BGR2GRAY );

155. threshold( gray\_all, threshold\_output\_all, 45, 255, THRESH\_BINARY );
156. 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

159. Point2f fourPoint2f[4];
160. //求最小包围矩形
161. RotatedRect rectPoint = minAreaRect(contours\_all[0]);
163. //将rectPoint变量中存储的坐标值放到 fourPoint的数组中
164. rectPoint.points(fourPoint2f);

167. for (int i = 0; i < 4; i++)
168. {
169. line(src\_all, fourPoint2f[i%4], fourPoint2f[(i + 1)%4]
170. , Scalar(20,21,237), 3);
171. }
173. namedWindow("Src\_all");
174. imshow( "Src\_all", src\_all );
176. //框出二维码后，就可以提取出二维码，然后使用解码库zxing，解出码的信息。
177. //或者研究二维码的排布规则，自己写解码部分
179. waitKey(0);
180. return(0);
181. }
182. </span>

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <QDebug>

using namespace cv;

using namespace std;

Mat src; Mat src\_gray;

RNG rng(12345);

//Scalar colorful = CV\_RGB(rng.uniform(0,255),rng.uniform(0,255),rng.uniform(0,255));

//获取轮廓的中心点

Point Center\_cal(vector<vector<Point> > contours,int i)

{

int centerx=0,centery=0,n=contours[i].size();

//在提取的小正方形的边界上每隔周长个像素提取一个点的坐标，

//求所提取四个点的平均坐标（即为小正方形的大致中心）

centerx = (contours[i][n/4].x + contours[i][n\*2/4].x + contours[i][3\*n/4].x + contours[i][n-1].x)/4;

centery = (contours[i][n/4].y + contours[i][n\*2/4].y + contours[i][3\*n/4].y + contours[i][n-1].y)/4;

Point point1=Point(centerx,centery);

return point1;

}

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

{

src = imread( "core.jpg", 1 );

Mat src\_all=src.clone();

//彩色图转灰度图

cvtColor( src, src\_gray, CV\_BGR2GRAY );

//对图像进行平滑处理

blur( src\_gray, src\_gray, Size(3,3) );

//使灰度图象直方图均衡化

equalizeHist( src\_gray, src\_gray );

namedWindow("src\_gray");

imshow("src\_gray",src\_gray);

Scalar color = Scalar(1,1,255 );

Mat threshold\_output;

vector<vector<Point> > contours,contours2;

vector<Vec4i> hierarchy;

Mat drawing = Mat::zeros( src.size(), CV\_8UC3 );

Mat drawing2 = Mat::zeros( src.size(), CV\_8UC3 );

Mat drawingAllContours = Mat::zeros( src.size(), CV\_8UC3 );

//指定112阀值进行二值化

threshold( src\_gray, threshold\_output, 112, 255, THRESH\_BINARY );

namedWindow("Threshold\_output");

imshow("Threshold\_output",threshold\_output);

/\*查找轮廓

\* 参数说明

输入图像image必须为一个2值单通道图像

contours参数为检测的轮廓数组，每一个轮廓用一个point类型的vector表示

hiararchy参数和轮廓个数相同，每个轮廓contours[ i ]对应4个hierarchy元素hierarchy[ i ][ 0 ] ~hierarchy[ i ][ 3 ]，

分别表示后一个轮廓、前一个轮廓、父轮廓、内嵌轮廓的索引编号，如果没有对应项，该值设置为负数。

mode表示轮廓的检索模式

CV\_RETR\_EXTERNAL 表示只检测外轮廓

CV\_RETR\_LIST 检测的轮廓不建立等级关系

CV\_RETR\_CCOMP 建立两个等级的轮廓，上面的一层为外边界，里面的一层为内孔的边界信息。如果内孔内还有一个连通物体，这个物体的边界也在顶层。

CV\_RETR\_TREE 建立一个等级树结构的轮廓。具体参考contours.c这个demo

method为轮廓的近似办法

CV\_CHAIN\_APPROX\_NONE 存储所有的轮廓点，相邻的两个点的像素位置差不超过1，即max（abs（x1-x2），abs（y2-y1））==1

CV\_CHAIN\_APPROX\_SIMPLE 压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标，例如一个矩形轮廓只需4个点来保存轮廓信息

CV\_CHAIN\_APPROX\_TC89\_L1，CV\_CHAIN\_APPROX\_TC89\_KCOS 使用teh-Chinl chain 近似算法

offset表示代表轮廓点的偏移量，可以设置为任意值。对ROI图像中找出的轮廓，并要在整个图像中进行分析时，这个参数还是很有用的。

\*/

findContours( threshold\_output, contours, hierarchy, CV\_RETR\_TREE, CHAIN\_APPROX\_NONE, Point(0, 0) );

int c=0,ic=0,k=0,area=0;

//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;

}

//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;

}

}

//填充的方式画出三个黑色定位角的轮廓

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() );

//获取三个定位角的中心坐标

Point point[3];

for(int i=0; i<contours2.size(); i++)

{

point[i] = Center\_cal( contours2, i );

}

//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[1]);

int area\_side = cvRound (sqrt (double(area)));

for(int i=0; i<contours2.size(); i++)

{

//画出三个定位角的中心连线

line(drawing2,point[i%contours2.size()],point[(i+1)%contours2.size()],color,area\_side/2,8);

}

namedWindow("DrawingAllContours");

imshow( "DrawingAllContours", drawingAllContours );

namedWindow("Drawing2");

imshow( "Drawing2", drawing2 );

namedWindow("Drawing");

imshow( "Drawing", drawing );

//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[0]);

//将rectPoint变量中存储的坐标值放到 fourPoint的数组中

rectPoint.points(fourPoint2f);

for (int i = 0; i < 4; i++)

{

line(src\_all, fourPoint2f[i%4], fourPoint2f[(i + 1)%4]

, Scalar(20,21,237), 3);

}

namedWindow("Src\_all");

imshow( "Src\_all", src\_all );

//框出二维码后，就可以提取出二维码，然后使用解码库zxing，解出码的信息。

//或者研究二维码的排布规则，自己写解码部分

waitKey(0);

return(0);

}

下面是代码运行的图片，代码处理过程的图片都有，包括二值化图片，轮廓图，找到的定位角图和最终的框出二维码图都显示出来了。如下：



这部分的代码主要是修改自[guanyonglai](http://my.csdn.net/guanyonglai" \t "_blank)博主的代码，在此基础上学习了相应的opencv库函数，并根据自己的理解添加了一些注释。实现的主要功能是根据QR二维码三个定位角的特点找出定位角的坐标，并框出整个QR二维码。

框出二维码后能做的事情就多了，可以使用相应的开源解码库解出二维码的信息，比如Zxing库，libdmtx库。Zxing库能解码的格式和支持的语音格式挺多的，网址：[https://github.com/zxing/zxing](https://github.com/zxing/zxing" \t "_blank)。libdmtx库主要解码data matrix二维码，网址：[http://libdmtx.sourceforge.net/](http://libdmtx.sourceforge.net/" \t "_blank)。这个库我使用过，很简单，直接调用几个函数就可以解出码的信息。

解码部分使用libdmtx库的方式已经实现的了，有时间再补充。去除opencv库，直接用算法处理图片并定位二维码并解码的也写了demo，这部分内容较多，有时间再补充，需要大概思路的可以留言。