#include <opencv2/opencv.hpp>

#include "core/core.hpp"

#include "highgui/highgui.hpp"

#include "imgproc/imgproc.hpp"

#include "zbar.h"

using namespace cv;

using namespace std;

using namespace zbar; //添加zbar名称空间

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

{

Mat image, imageGray, imageGuussian;

Mat imageSobelX, imageSobelY, imageSobelOut;

image = imread("C:\\Users\\xiaodong\\Desktop\\123\\QRcode测试\\2.jpg");

//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();

}

#include <opencv2/opencv.hpp>

#include "core/core.hpp"

#include "highgui/highgui.hpp"

#include "imgproc/imgproc.hpp"

#include "zbar.h"

using namespace cv;

using namespace std;

using namespace zbar; //添加zbar名称空间

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

{

Mat image, imageGray, imageGuussian;

Mat imageSobelX, imageSobelY, imageSobelOut;

image = imread("C:\\Users\\xiaodong\\Desktop\\123\\QRcode测试\\barcode.png");

//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);

//11.裁剪

Rect myRect = boundingRect(contours[0]);

Mat img = Mat(image, myRect);

imshow("11.裁剪", img);

//12.识别

ImageScanner scanner;

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

Mat imageGray2;

cvtColor(img, imageGray2, CV\_RGB2GRAY);

int width = imageGray2.cols;

int height = imageGray2.rows;

uchar \*raw = (uchar \*)imageGray2.data;

Image imageZbar(width, height, "Y800", raw, width \* height);

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;

}

imshow("Source Image", image);

waitKey();

imageZbar.set\_data(NULL, 0);

system("pause");

waitKey();

}