Project Report for CIE6004

217019004 Yiping Jiang

Introduction

Our goal is the capture of the bar code on express lists or other surfaces. Usually, we have two bar codes , a big one and a small one, containing the same information.

And here we have 3 images for testing, among which 2.jpg is the best one of them, while 1.jpg is has a rotation and 3.jpg has 2 QR codes very close to the smaller bar code.

Figure 1: A fairly ideal image

This one is a fairly ideal image for testing.

Figure 2: Another image with rotation

A rotation of 45° is applied to another express list. Why is 45? Because gradient add/minus operation plays a key role in the capturing process, and a 45° rotation will do most "harm" or "damage" to this operation, if the harm or damage does exist.

Figure 3: QR codes very close case

Choice for Methods

In our three assignments, we do many operations on images with different characteristics to

make them more comfortable when being watched. However, in most cases, pictures are not that luckily designed, so we should do multiple steps to obtain a satisfying result.

Considering the characteristics for bar code, it's natural to think about gradient related method; and for the choice of kernel for gradient, it's easy to apply different kernels with different sizes in opencv, and after lots of tryings, I decided to use Schar:

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

And we compute gradX for a pixel (x_1, y_1) like this:

$$\sum_{x=1}^{3}\sum_{y=1}^{3} G_x(x,y)S(x_1+x-2, y_1+y-2)$$

Then we add gradX and gradY for every pixel to obtain a gradient graph with the same size as original picture.

(Notice that I have tried both adding and subtracting the gradients on X and Y by using cv2.add and cv2.substract functions, and then abandoned the latter one because it has difficulty bearing rotation.)

But QR codes and logos will still be severe disturbances. So how do we solve this problem? Or, what else should we do to enhance the result of gradient add/minus below?

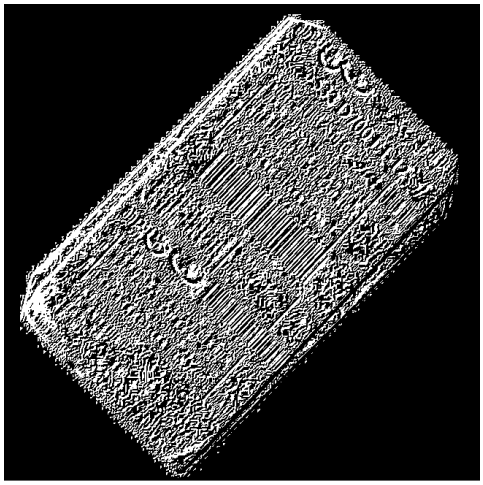

Figure 4: Gradient

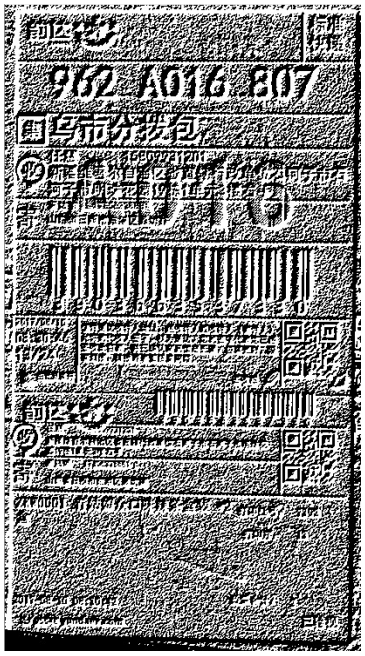

Figure 5: Gradient for Figure 3

Image smoothing is taken into consideration, since QR codes and logos has more holes and their structures are not as strict as bar code. So a proper size of blurring should help with distinguishing them from bar code.

And here is the result after blurring and threshold binary basing on the gradient result:

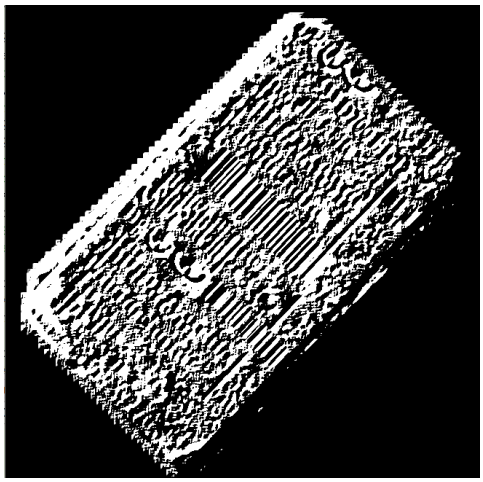

Figure 6: Blurring

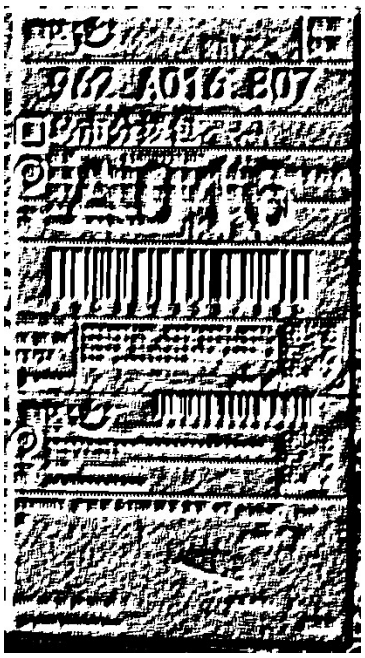

Figure 7: Blurring for Figure 3

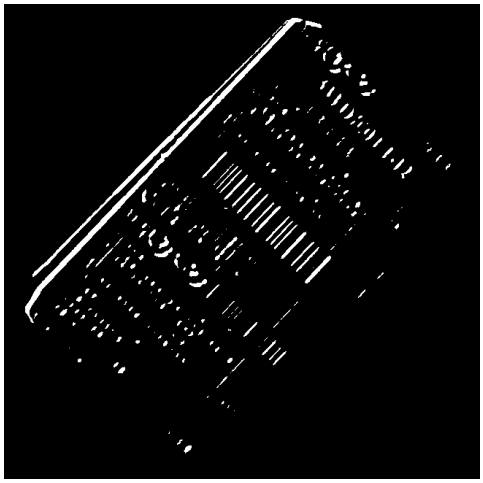

Figure 8: Threshold binary

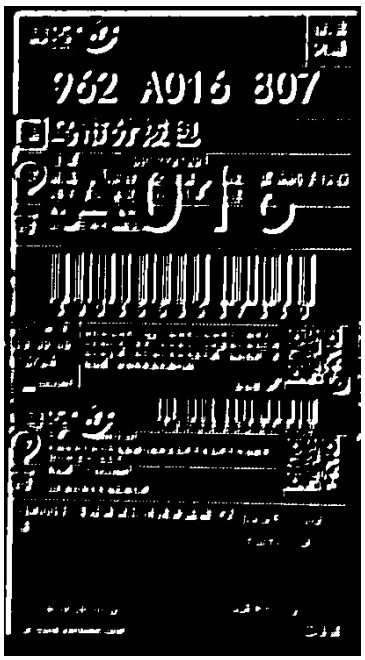

Figure 9: Threshold binary for Figure 3

We notice that, blurring and threshold binary made the bar code the biggest part with obvious typical structure, and the QR codes and logos are blurred.

And basing on the result above, we can apply several times of erosion first(connect white regions and kill small useless parts) and dilation afterwards(ensure the white region back to original size) to ensure that the bar code becomes the largest white region:

Figure 10: Erosion for several times

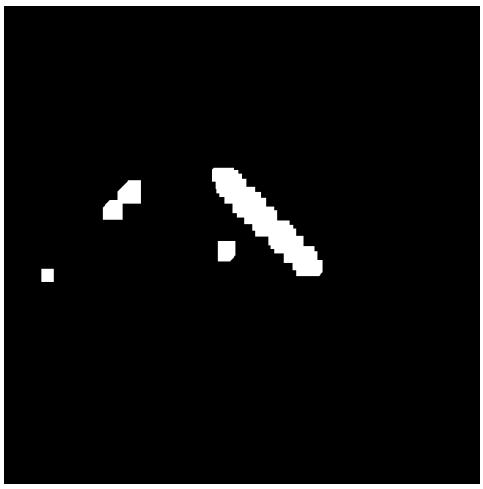

Figure 11: Dilation afterwards

Note that we erose or dilate for more than once for disconnect some possible connected disturbance which may be connected and become the largest white part together instead of the oriented big bar code.

And to ensure the size for the bar code, we erose and dilate for same times.

Then we can find contour for the bar code and paint a box onto the image:

Figure 12: Contour

Finally we use four_point_transform to cut the bar code out:

Figure 13: Cutout bar code

So the methods we use can be listed as below:

Gradient computation

Blurring, threshold binary

Closing operation

Erosion and dilation

Finding contour and cutting-out basing on four points

Difficulties:

Sizes for kernels(gradient, blurring, closing, erosion, dilation);

How many iterations for each kind of operation is enough to exclude those useless components without damaging the oriented bar code;

If the angle and position of express list are not satisfying, will the code still work? This is specifically referring to the 2nd and 3rd pictures.

Codes and Environment

Finally I chose OpenCV(basing on python) to do the project. The running environment is OpenCV 2.4 + Python 2.7 on Ubuntu 16.04 Operating System.

And for detailed code, please see capture.py.

Summary

The idea for this project is obviously not something new or original, since there are already many APPs and programs doing this job. I hope to experience a transformation from user/customer to a developer through this project.

To capture a bar or QR code inside a not well structured picture seems easy to do as a user, but how to realize the function or even enhance the function all by myself is a challenge.

And the process and related methods have nothing special. We learned all of them in class. But how to select the proper methods and how to combine and use them in order, and especially, how to set a proper size for the parameters in the built-in functions...these all takes time to adjust and practice repeatedly.

Others

Here we attach result for the other two images:

Figure 14: Contour 2

Figure 15: Barcode 2

Figure 16: Contour 3

Figure 17: Barcode 3