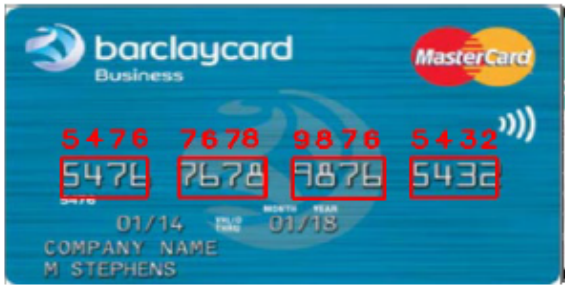
Credit Card Recognition Project

1.Brief Introduction

This project aims to recognize the 16-digit card number of 8 different credit card (card 1 to 8 as file name indicates) with opencv-python(cv2 verision:4.2.0.34) . You can find the image of each card in the folder named 'img'.

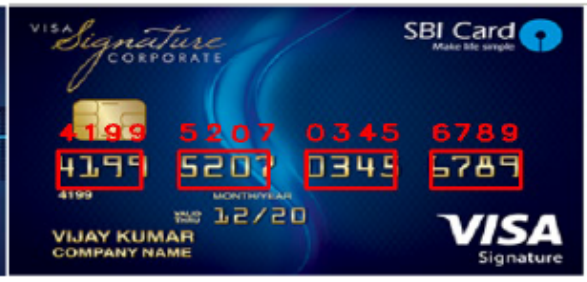
For card 1-7, this project successfully detect all 16 digits on each card. Below shows all the results:



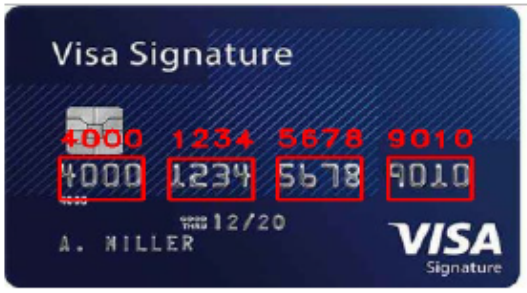
Card 1

Card 2

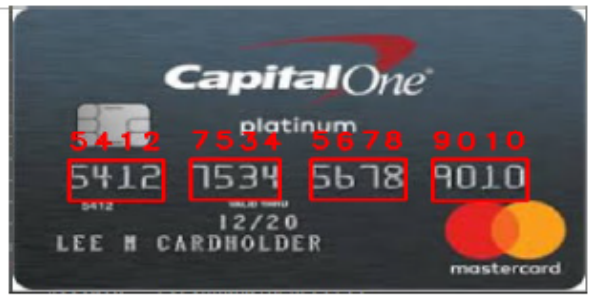
Card 3



Card 4



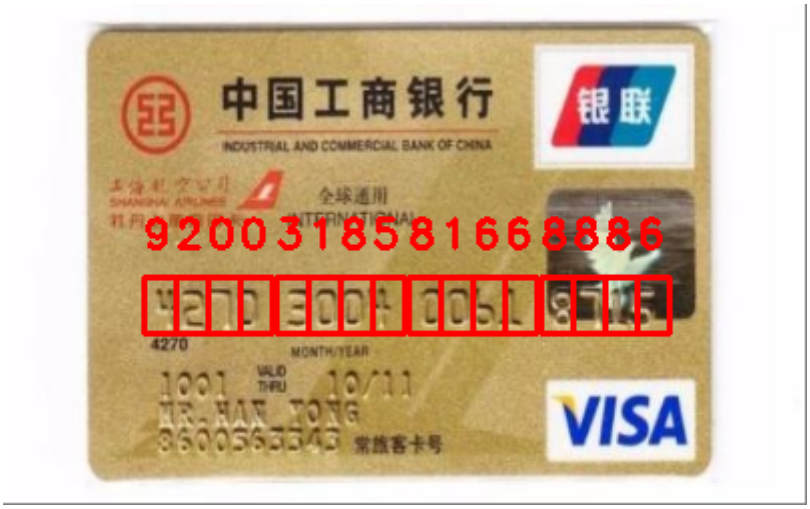
Card 5



Card 6

Card 7

For card 8, this project fails to recognize all the digits. The best one correctly detects 5 digits out of 16. The corresponding code is in file **'unsloved_card8.py'**. Below shows the 'best' result:



This project is a good start for someone(like me) who just begin their learning path in opencv and computer vision, and only requires some basic understanding of image operations, including:

- Image morphology: dilation, erosion, opening, closing, tophat, blackhat
- Filter and convolution: threshold operation(otus's), kerme;s, convolution
- Edge detector and template matching: Sobel, Canny, template matching
- Contours: find/draw, features
- Drawing and annotating

Here lists some useful reference both on the project and opencv:

- OCR project:
  - github
  - csdn
- opencv
  - MyNote
  - "Learning OpenCV3" by A.k. & G.B.: Chapter6, 10, 12, 14.

2.Project Design

The whole project can be narrowed down into 5 parts:

1. Create templates for all 10 digits(0-9):
   - note that, there are 2 different using in these credit card. Specifically, the 1st, 3rd and 4th card use template 1('ref1.png'), and the rest 5 cards use template 2('ref2.png').
2. Process the image in order to locate each digit on the card:
   - to process the card with 'dark' background by tophat, e.g. card 1,3-6
   - to process the card with 'bright' background by blackhat, e.g. card 2
   - for the the card with combined background, e.g. card 7, dividing the card into dark ROI and bright ROI, then eliminate background noise by tophat and blackhat, respectively
3. Template matching:
   - use the locations of each digits found in part2, and the template created in part 1, to match the template.
4. Visualization
   - to draw the result by cv2.rectangle() and cv2.putText()
5. Passing parameters' function and packaging:
   - to use argparse module to build the parameter-passing function
   - to aggregate the results for all 8 cards by packaging

3.Cases

Case 1: Card 1, 3-6

- They all have 'dark' background, and it turns out that the tophat technique fulfill image processing need.
- Regarding iteration: card 1 do not require any iteration, while the rest four cards require 2 iterations
- Card 3-6 can share exactly the same parameters/codes, no need to tune.

Case 2: Card 2

- The blackhat with 2 iterations would successfully detect the digit on it.

Case 3: Card 3

- It has both bright and dark part, so I divide the card into dark ROI(roi_2) and bright ROI(roi_1&3) as below:

```
    roi_1 = gray[135:165,:125]
    roi_2 = gray[135:165,125:180]
    roi_3 = gray[135:165,180:]
```

- For roi_1&3, I dealt with them in the same way as in case 1.
- For roi_2. I tried blackhat it, but it failed to show a clear contours. Thus, I used Canny edge detector, which results in a good match

Case 4: The unsolved case - card 8

- This project succeed in locating all the 16 digits on this card. By Canny-->dilation, the digits are evenly divided into for groups, and the first 2 groups have pretty clear shapes, so that I can locate all the individual digits by assuming(which is also the truth) groups, and the digits in each group, are evenly separated. The result is shown above.
- This project fails in template matching in this case. I've tried all the above methods in case 1-3 and found the best contours/image come from blackhat. Then I tried to tune several key parameters. including: the kernel size and type, the roi size, the iteration times, the threshold values. In the end, I could only correctly recognize 5 digits out of 16 as the best result.
- If anyone can provide a better solution for this case, pls contact/at me on GiuHub. Thx in advance.