Coin Detection:
Useful tool for counting coins

Shao-Hung Chiu, Tzu-Chun Peng
Department of Electrical Engineering, National Tsing Hua University
All of our code is on GitHub: https://github.com/TomChiu5566/Coin_Detection

Abstract

We introduce “Coin Detector”, a simple application which can detect and count the numbers of each kind of NTD coins. The system input is just an image only with coins and the standard size of a ten-dollar coin, and the output is the information of coins in the image. With moderate light, good contrast of background and without filming angles, the coin detector can be very accurate.

The coin detector is accomplished by a series of image processing, which includes filtering noises, finding edges, drawing contours, and finally cropping coins individually. As long as we crop coin images, we can focus on features of each. Among many features of coins, our coin detector only judges the kind by the size of each coin, which is the most distinctive feature. In the end, it can sum up the number of each kind and total dollars, and show it to users.

Figure 1. Coin Detection can count coins of each kind

1. Introduction

In modern society, people always buy stuffs with money and sell products for money. In everyday life, we only trade with a small amount of money, and hence, we have coins in hand. Usually, it is hard to tell how many dollars there are or how many ten-dollar coins there are in your wallet.

We need technology for this kind of trivia to make life more convenient. Therefore, we design the coin detector, which can help people to deal with coins. We design a series of processes to achieve detection, including filtering, edging, and finding contours. Then, comparing the size of each coin to that of the ten-dollar standard, it can tell the kind accurately.

The system is constructed under *Python3.5* environment, and includes many functions in *cv2*.

2. Method and Details

a. Gray and Blurred

At the very beginning, the image is turned into gray scale since we do not need RGB color scale for the following processes. We focus on intensity.

Then, we filter the image with a Gaussian filter of 15*15 in our design.[1] The goal of filtering is to eliminate noises in the image. Coins have patterns on them, and there are noises on the background. The image without noises and redundant components can make the following processes more smoothly.

b. Canny Edge

This is an algorithm developed by John F. Canny and it is implemented in *cv2*.[2] It calculates the horizontal and vertical gradient, and also the direction of the gradient.

Then, it applies “Non-maximum Suppression”, which can help check if a pixel is a local maximum in its direction. Here we get “thin edges”. Then we set upper threshold and lower threshold to select valid thin edges. An edge above the upper threshold is “sure-edge”. An edge below the lower edge is discarded, and an edge between the two values is judged by whether the edge is connected to a sure edge or not.

In practice, we set the upper threshold as 100 and the lower threshold as 30. After processing, we can get an image with clean edge, and each edge can be taken as contours.

c. Find Contours and Set Thresholds

Intuitively, we can now circle coins by edges. However, light and contrast are not necessarily good, so we need to pick out some unreasonable contours. For example, contours with too few or too much areas are discarded.

Then we draw the contours down on the image. We bound an individual contour with a rectangle whose height and width are almost the radius of the coin. Here, we also try to filter out contours which do not make any sense. Hence, we discard contours with height or width less than 10, and those with height or width more than 130. Also, we skip a contour if its moment, like the center of gravity, is too close to any processed contour before.

d. Crop Coins and Tell Size

In (c), we have rectangles with height and width almost the same as the radius of a coin. Now, we can judge a coin by the size of its rectangular box. In this stage, we need a standard in the image since we have to compare to something to get the actual size of a coin.

We recognize the standard by calculating the intensity in the image. Since our standard is all white, it should have the strongest intensity.

After calculating average intensity of rectangular boxes, we can find out which is the standard.

Considering NTD coins, the radii are[3]

1-dollar: 20mm

5-dollar: 22mm

10-dollar: 26mm

50-dollar: 28mm

And we use 10-dollar radius for our standard, so we calculate which range the average of the box's height and width is. And we determine which type of coin it is

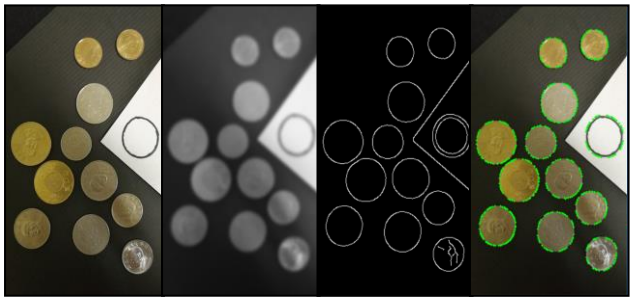

Figure2. From the left to the right are images of original, gray and filtered, canny edging, and effective contours

3. Results

We can successfully detect and tell the kind of coins by size. The followings are some successful detection.

Figure 3-1. One of successful detection

Figure3-2. One of successful detection

Figure3-3. One of successful detection

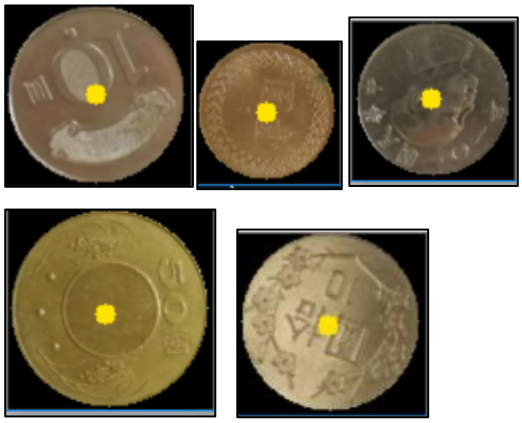

Figure3-4. Cropped Coins

Also, there are also some examples of unsuccessful detection, which we will discuss later.

4. Discussion

a. Incompleteness

In practice, we have to control the filming conditions. When conditions, e.g. light, coin colors, or filming angles, are not controlled well, the accuracy decreases.

For example, when light is too intense, the coins, especially 5-dollar and 10-dollar, reflect. We cannot accurately find which cropped coin represents the standard, so the whole system collides.

If there are several light sources when taking photos, they cause shadows of coins. For fear of inaccuracy, we use black background for coins. Since we use the black background for our input images, dark colors lead to failures in edge detecting. However, in order to boost the accuracy overall, we still use the black background and give up the corner case for now. The similar problem occurs when light is too weak.

Filming angles are obviously one of the crucial factors since we use size difference to distinguish types of coins. When taking photos of coins, we have to hold the camera as parallel to the coins as possible.

Figure 4-1. Weak Contrast

Figure 4-2. Take 5-dollar as standard by mistake

b. Trials and Failures

In the beginning, we tried several methods to detect edges. For instance, we tried watershed algorithm [4], closing and dilating images after adapting threshold [5], but the defects of them are that they all need very good contrast to raise the success rate. In order to come up with a more general and practical design, we give up those algorithms.

At the stage of telling types of coins, we were trying to build a system with scores. The criteria include patterns, colors, and sizes. However, pattern recognition needs more advanced processing skills, or perhaps deep learning, so we don't have a good result for this.

Also, the colors differ a lot even for the same kind of coins. For example, a 1-dollar coin is extremely dirty so that it looks like the color of 5-dollar coin. Or a 10-dollar coin is so oxidized that its color is similar to a 1-dollar coin. Besides, light affects the result meanwhile. We tried converted images to HSV color space but in vain. Judging the size is the most effective method, so in the end, we decide to tell the type of coins by size, but not colors or patterns.

c. Future work

The systems can be improved with several respects. First of all, we should simply the user interface. To illustrate, we can let the user to pick out which coin in the image is 10-dollar, and we count all coins of the rest. Then, we don't need the paper with a standard on it anymore.

Besides sizes, we can surely distinguish a coin by its pattern, and we believe that deep learning can help a lot and do better with conventional method. If we are equipped with the skills in the future, the accuracy can boost up.

The detecting stage sometimes does not function well due to contrast. If the problem can ultimately improve, the system will be perfect and we can do more effect on telling which kind of coin it is.

If the system can be made an application on smart phones, that will be more convenient for people to use it. It might be an effective way to make the detector more useful.

5. Conclusion

We designed a system of detecting coins in an image. It can count numbers of all kinds of coins. Perhaps, in the future, each store has this system for checking the changes. Or people use this application on their smart phone and change their lives to some degrees.

After finishing the design, we get more familiar with image processing. It is a very profound topic, and we learn a lot. To say it straightforward, it is so much fun when seeing the results. The result is encouraging, and it means a lot for us. In the future, we believe we have the ability to accomplish more difficult tasks.

6. Reference

[1] Gaussian Filter in cv2:

https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[2] Introduction of cv2.Canny():

https://docs.opencv.org/3.3.1/da/d22/tutorial_py_canny.html

[3] Coins' sizes:
https://museum.cbc.gov.tw/web/P2_3.aspx?menu=2&id=11&sub=54

[4] Watershed in cv2:
https://docs.opencv.org/3.3.1/d3/db4/tutorial_py_watershed.html

[5] Adapt threshold using cv2.threshold():
https://docs.opencv.org/3.3.1/d7/d4d/tutorial_py_thresholding.html