Project: Coin Detection in an Image

The topic of the project is to locate coins of different sizes in a given test image.

1 Read Image

Use *imread* function to read the image pixels into an array. Convert it into gray scale using the following combination of R, G, B components of each pixel: 0.2989*R + 0.5870*G + 0.1140*B. To display images you can use *image*, *imshow* or *imagesc*.

2 Edge Detection

Read about edge detection methods in general. We will implement a well known method, Sobel edge detection, and follow the three basic steps:

- Obtain the gradient of the image.
- Find the magnitude
- Threshold the magnitude image.

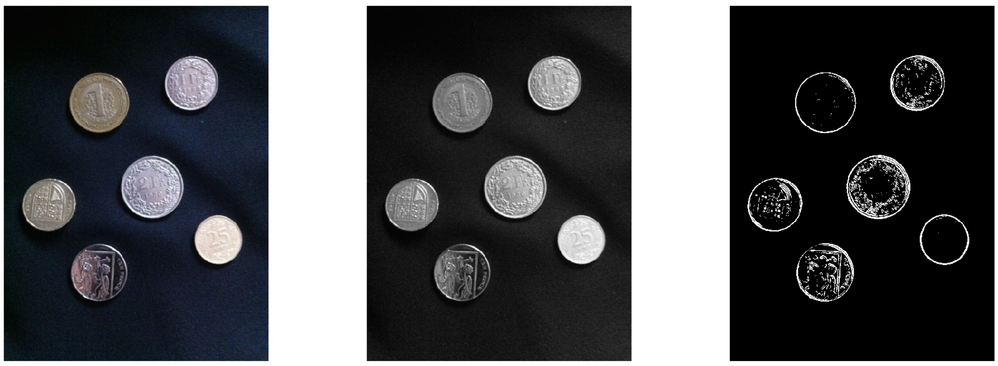
We will use the following filter masks:

$$F_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1] \end{bmatrix}.$$

Use the mask F_1 for x direction and F_2 for y direction, convolve them with the image I and obtain the gradient of the image, G_x and G_y. This means moving the filters one row or one column at a time over the image starting from the top left corner until the bottom right corner. At each position we calculate the sum of the products of each filter element with the image pixel value at the corresponding position, find the magnitude of the vector: $sqrt(G_x^2 + G_y^2)$, and assign it to the center location (of the 3x3 subimage whose elements were used in the calculation) in the new image. Since we need 3x3 image pixels, the border pixels are not considered, so the assigments start from the pixel (2, 2). Finally, find a suitable threshold to generate a binary image: pixels with magnitue less than the thresold are set to 0 and the rest are set to 1. Compare your results with the built-in matlab function, *edge*, using the sobel operator. Example results from the steps until now can be seen in Figure1. You can implement another method as well and compare the results, if you have time.

You can experiment with morphological operations such as erosion, dilation, opening, closing, removing small regions after finding regions using connected component analysis, if you have time, to improve the results.



(a) The original image (b) The grayscale image (c) The edge image

Figure 1: Steps to prepare the image for Hough Transform: Conversion to grayscale and edge detection using Sobel operator.

3 Hough Transform

We will find the circle locations in the image using Hough Transform. Read about Hough transform for the detection of lines and circles. The details of the algorithm are as follows: A circle with the center (a, b) and radius c is given by $(x - a)^2 + (y - b)^2 = c^2$. So, we have a 3D parameter space. We will maintain an accumulator A for these parameters. Initially $A(a, b, c) = 0$. A point in this 3D space will correspond to a circle in the image. So for each point that lie on a circle in the image, we will increment the corresponding entry in the accumulator. We will increment an accumulator entry by 1, if the parameter values for that entry correspond to a circle that the pixel $(I(x, y))$ belongs to. We can summarize the algorithm as follows:

```
for every (x,y) where I(x,y)==1
for all a and b
$c = sqrt((x - a)^2 + (y - b)^2)$
$A(a, b, c) = A(a, b, c) + 1$
Find the maxima in the parameter space. A maxima corresponds to a circle.
```

The result is a histogram or a vote matrix showing the frequency of circle points corresponding to certain (a,b,c) values (i.e. points lying on a common circle). A is thresholded such that only the large valued elements are taken. These elements correspond to strong circles in the original image. So, the maximum values in the accumulator correspond to the most prominent circles which are defined by the 3 coordinates of the entries with the maximum values. A maximum value at position (a,b,c), $A(a, b, c)$ corresponds to a circle $(x - a)^2 + (y - b)^2 = r^2$ in the original image. You can use *imregionalmax* function to find maximas, in addition to that, you can also use a threshold for the accumulator values to eliminate entries with low values.

Some of the test images (see the links below) have circles of equal sizes and some are with circles of different sizes. For simplicity, you can start implementing the algorithm assuming that the radius is given, and choose a radius value that leads to good results. Then try to extend your algorithm to search for circles with unknown radius. Select a reasonable range for radius. Try to implement this algorithm using the Matlab tricks (to speedup) that you discussed so that the program is not too slow, e.g., preallocate the accumulator, try to avoid for loops etc.

4 Detect Circles

Find the centers of the circles that would fit the coins. Plot the circles on top of the RGB image as in Figure2.

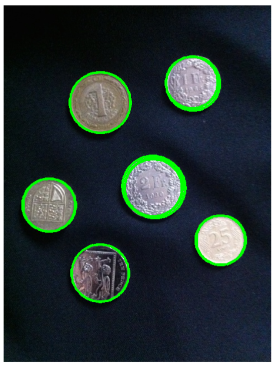


Figure 2: Circles detected with Hough transform.

5 Testing Your Program

Use the test images (see the links below), and make sure that your program is able to detect each coin in at least three images.

6 Examination

Prepare a brief report by explaining each algorithm and how you implemented them. Please submit all the .m files needed along with a short README file where you explain how to run your code.

7 Links

Download the test images from `www.csc.kth.se/~yaseminb/test.tar`.