Project -3 Edge and region-based recognition:

By
Rahul Pasunuri (rahupasu@indiana.edu)
Jothin Reddy Vedre (jvedre@indiana.edu)

Pre-Processing:
As the images are of very high quality, the size of the components increases becomes very high, and it becomes computationally not feasible to work with it. So, we have re-sized the image to a proportionate size of 20% of the original width and height. We have ensured that the aspect ratio remains same even in the re-sized image so that the circles in the original image remain as circles in the re-sized image as well.

2.1)
Once we resize the image, we apply a Gaussian blur with "sigma = 1.5" and then do Canny Edge Detection which involves the following:

(i) Convolution with the Sobel operators:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

where A is the input image, Gy is the vertical derivative and Gx is the horizontal derivatives. Once we get the derivatives, we calculate the gradient magnitude and gradient direction using the following formulas:

Gradient Magnitude:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Gradient Direction:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

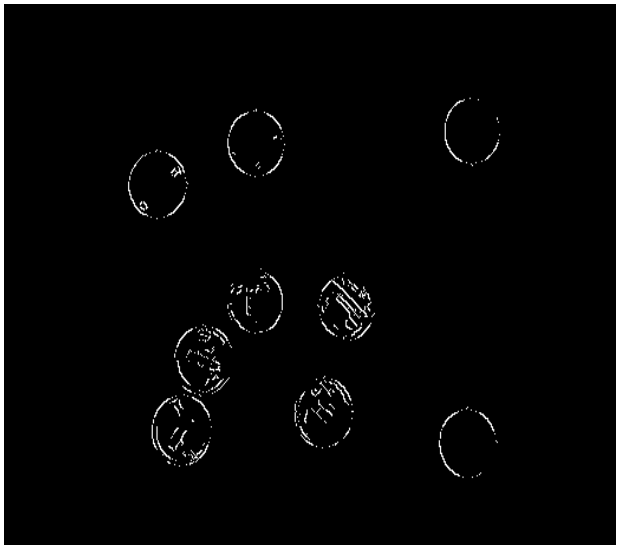
(ii) Non – Maximum Suppression:

Once we get the gradient magnitude and the direction, we do non – maximum suppression to thin the edges. For each pixel, we check the neighbor pixels perpendicular to it's gradient direction and assign the value of the current pixel to 0 if that neighbor's magnitude is greater than current pixel's magnitude. We do this for all the pixels, thus reducing the likelihood of identifying thick edges.

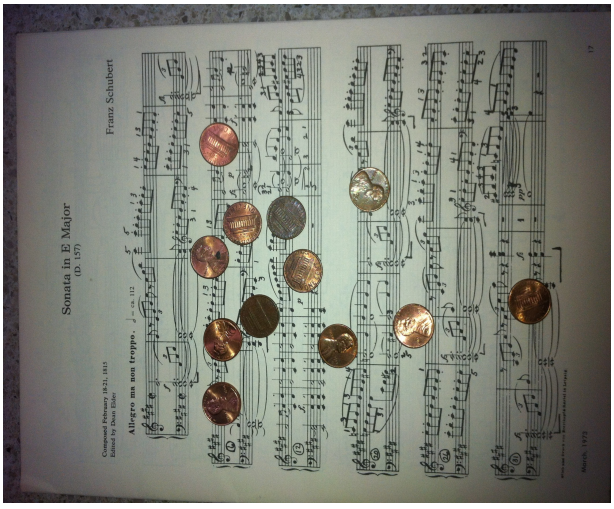
(iii) Hysteresis Threshold:

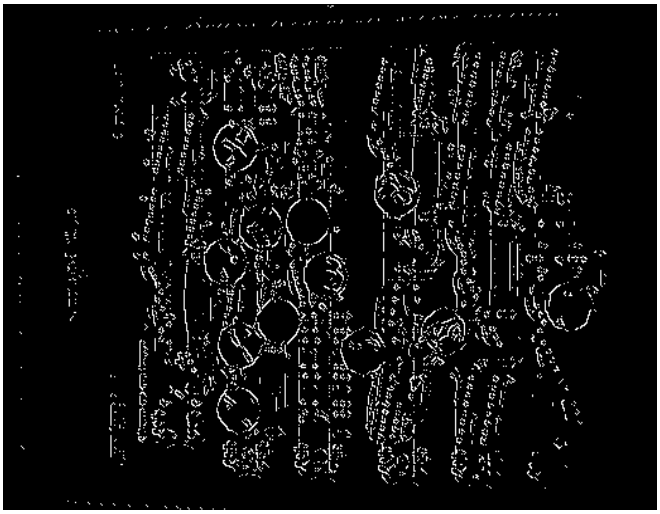
Once we thin the edges, we remove noisy edges by using hysteresis thresholds. We use two threshold values, "edgeLower = 50" and "edgeUpper = 100". Everytime we encounter a pixel with value greater than the "edgeUpper", we assign it a value of "255", look for neighbor pixels with value greater than "edgeLower" and consider them as edge pixels by assigning value "255".

Canny Edge Detection – Example -1 :



Canny Edge Detection – Example – 2 :





2.2)
After Canny Edge Detection, we use Hough Transform to look for circles among the edges. We use the following steps:

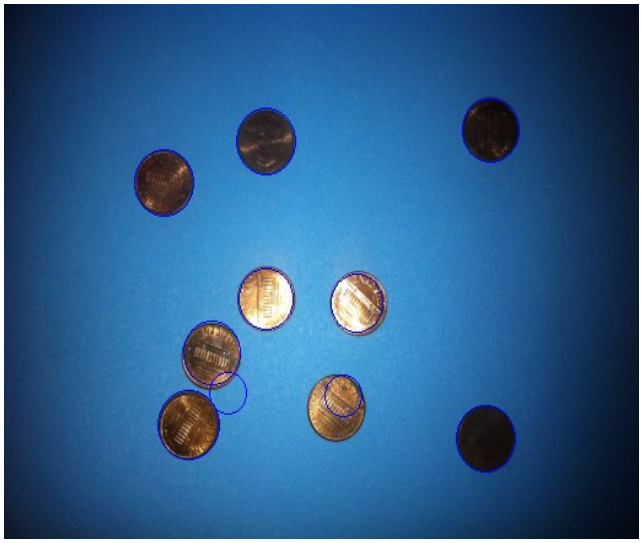
(i) We provide a radius range of “radiusLow = 15” and “radiusHigh = 35” to account for circles in the image after resize, since the training images provided to us and the test images were taken at the same time using the same camera, with the same height, zoom and resolution. For each pixel, we then loop through the radius range and theta 0 to 360 degrees, to apply votes in our accumulator cell, by increasing the accumulator value (by 1) at the possible center of the circle on which that pixel might be present.

(ii) Once we have the accumulator cell, we normalize the range of the accumulator values to 0 – 255.

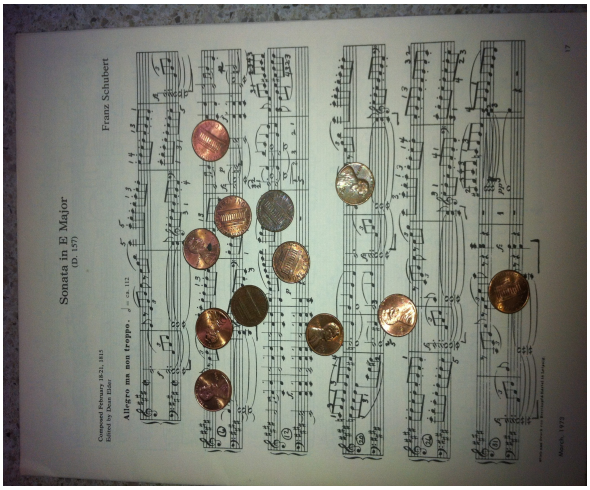
After normalizing the values, for each pixel, we obtain the radius with maximum votes to draw the circle outline with that radius. A pixel is identified as a center if it's pixel value is greater than the "circleThreshold = 110". We then calculate the maximum value among the possible centers of the circle to make sure that a single circle is not detected multiple times. We do this for a neighborhood size of "19".

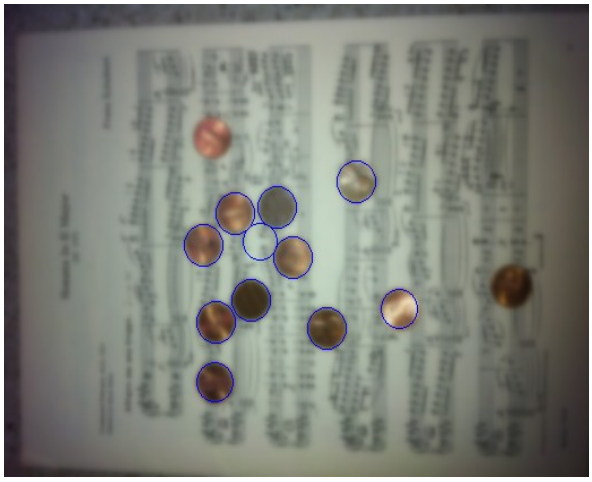
(iii) Once we get the center and radius with the maximum votes, we draw a blue color outline to the circle, using those values.

Hough Tranform – Example 1:



Hough Transform – Example 2:



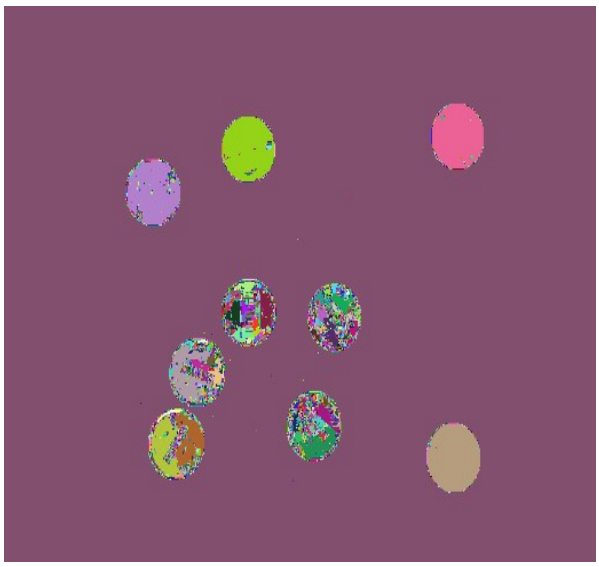


3Q)
After segmentation, we are assigning random RGB colors to each component and saving it in the "regions.png" file. Also, we use some heuristics to determine whether a given component is a circle or not using the aspect ratio and fill ratio measures. Using these, we identify circles, and assign random RBG colors to each circle, and then give a unique color to all non-circle components and save the final image with the name "seg_detected.png".

3.1)
Below results are obtained using a max-edge-weight threshold as '20'.

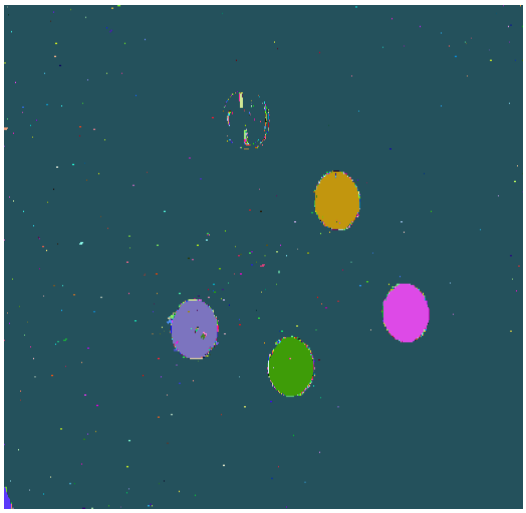
Example-1:
Input Image:

Regions Image:



Example-2:
Input Image:

Regions Image:

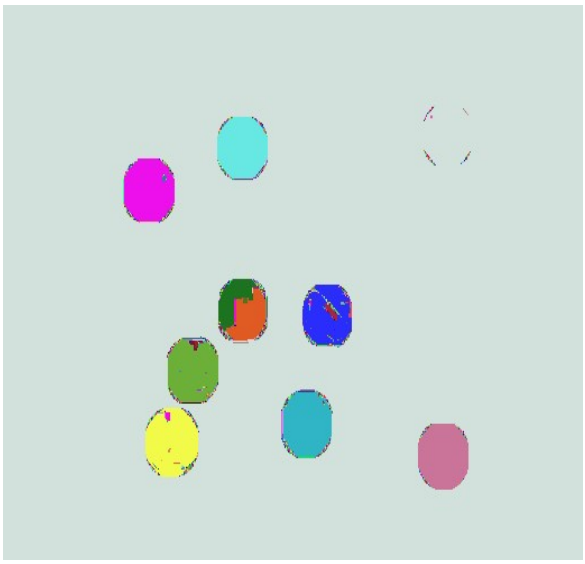


3.2)
Here we blurred the input image with a gaussian kernel (sigma value 1), and then used the 'k' value as "20".

Example-1:

Input Image:

Regions Image:

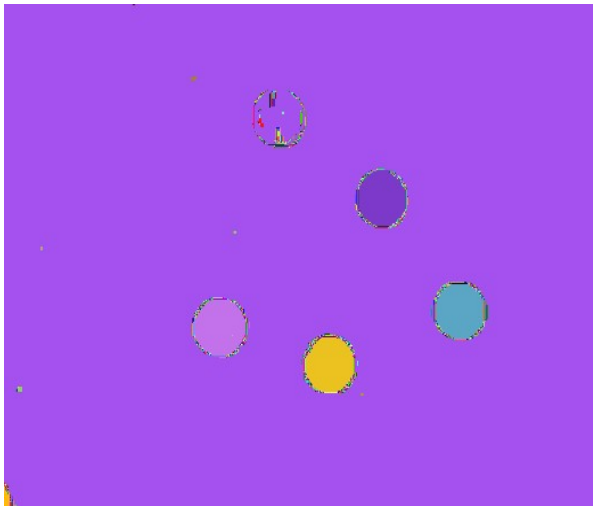


Example-2:

k value used here is “10”.

Input Image:

Regions Image:



3.3) Here we are reporting results using the modified segmentation algorithm created in (3.2). We used three simple measures for identifying the "circularity" of a component.

1) Minimum Circle Size:

This threshold filters those segments which have a vertex size below some threshold. (a value of 100 was used here.) This threshold is used to filter out the components which are too small.

2) Aspect Ratio:
The aspect ration of a circle is 1. So our components must be having an aspect ratio close to 1. So, we have used two thresholds, a min-aspect-ratio (a value of 0.85), and a max-aspect-ratio (a value of 1.15).

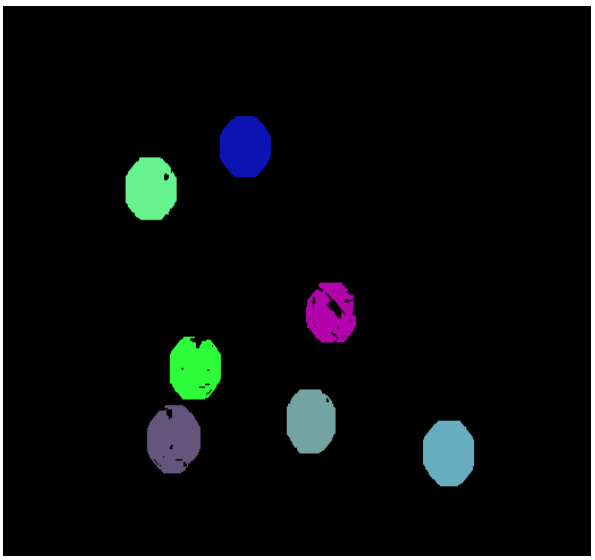
3) Fill Ratio:
By fill ratio, we mean the ratio of area occupied by the circle to the area of the bounding square of the circle. The ideal value for the fill ratio is "pi/4". So, we use two thresholds two ensure that the component's fill ratio is close to that value. (a min value of 0.7 and a max value of 0.9 were used here).

Here, we are giving random colors to circular components, and "black" color to all non-circular components.

Example-1:

Input Image:

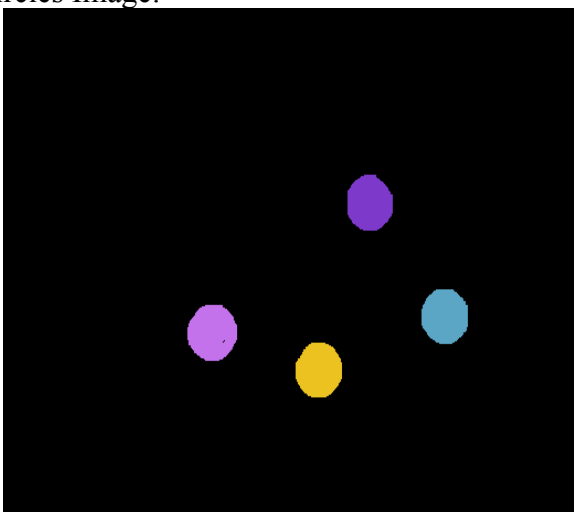
Circles Image:



Example-2:

Input Image:

Circles Image:



4Q) We are using the Hough Transform approach (implemented as a part of Q2) to detect coins.

To identify coins from the circles identified, we are using the below measures.

1) Color Variance:
We observe that, the color values inside a coin, have a minimum and a maximum color-variance. We have used these two thresholds to filter out some of the circles. (a minimum value of 10 and a maximum value of 80 are used)

2) Max Radius and Min Radius:
After we identify all the circles, we take a mean of the radius of all the circles, and we filter out the circles whose radius is far from the mean. This filters out the outliers, who have extreme radius values.

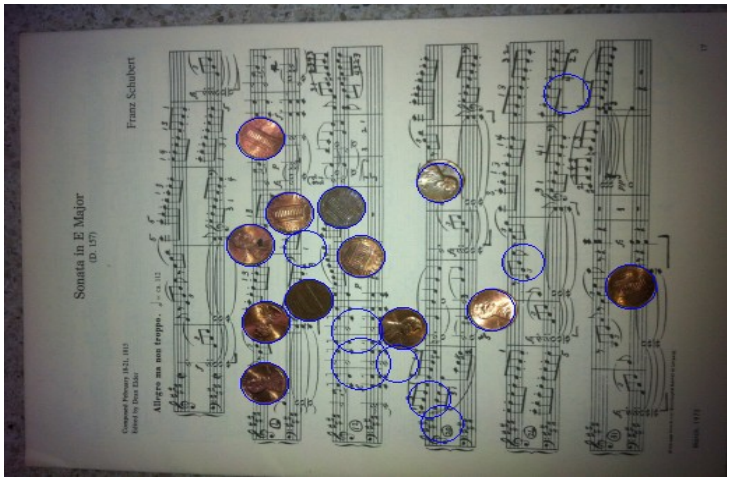
3) Color Matching:
We have observed that the color values in the RGB spectra follow some properties for coins. We have used theses conditions to filter circles which are not coins.
a) Atleast 10% of the pixels inside the circle must have RED value > 50.
b) Either RED value > GREEN value or GREEN – RED < 15.
c) Similarly GREEN > BLUE or BLUE – GREEN <15.

Using the above filters, we got the below results.

Example-1:

Example-2:

Example-3:



Citations:

http://en.wikipedia.org/wiki/Sobel_operator

http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

https://www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf