Project 3

CSE-573(Computer Vision And Image Processing)

Prof. Junsong Yuan

Student name: Disha Mehra

UBIT name: dishameh

Person Number: 50288911

Question 1:

Part a)

In Part a, I have implemented erosion and dilation separately and I have taken a rectangular mask of size (5,3).

To remove the noises in the image, I have used two morphology image processing algorithm separately resulting in two images.

```
def dilation(img,mask):
    new_img = np.zeros(img.shape)
    h= len(mask)//2
    w= len(mask[0])//2
    for i in range(h,len(img)-h):
        for j in range(w,len(img[0])-w):
            val_region = img[i - h: i + h + 1, j - w: j + w + 1]
            val = 0
            for k in range(len(mask)):
                for l in range(len(mask[0])):
                    if(val_region[k,l] == mask[k,l]):
                        val += 1
            if(val != 0):
                new_img[i][j] = 255
    return new_img
```

1st morphological image processing algorithm:

I have applied opening (erosion followed by dilation) and then closing (dilation followed by erosion).

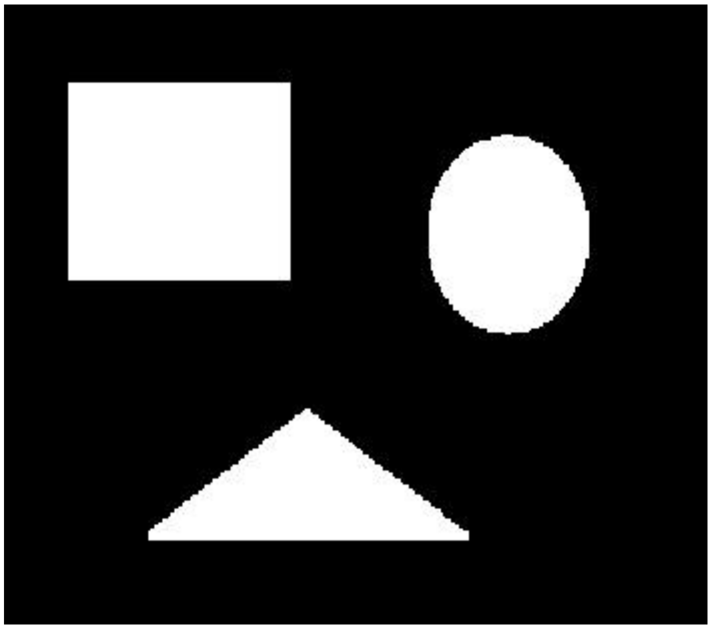
```
def erosion(img,mask):
    new_img = np.zeros(img.shape)


    h= len(mask)//2
    w= len(mask[0])//2


    for i in range(h,len(img)-h):
        for j in range(w,len(img[0])-w):
            val_region = img[i - h: i + h + 1, j - w: j + w + 1]
            val = 0
            for k in range(len(mask)):
                for l in range(len(mask[0])):
                    if(val_region[k,l] == mask[k,l]):
                        val += 1
            if(val == mask.size):
                new_img[i][j] = 255
    return new_img
```

2nd morphological image processing algorithm:

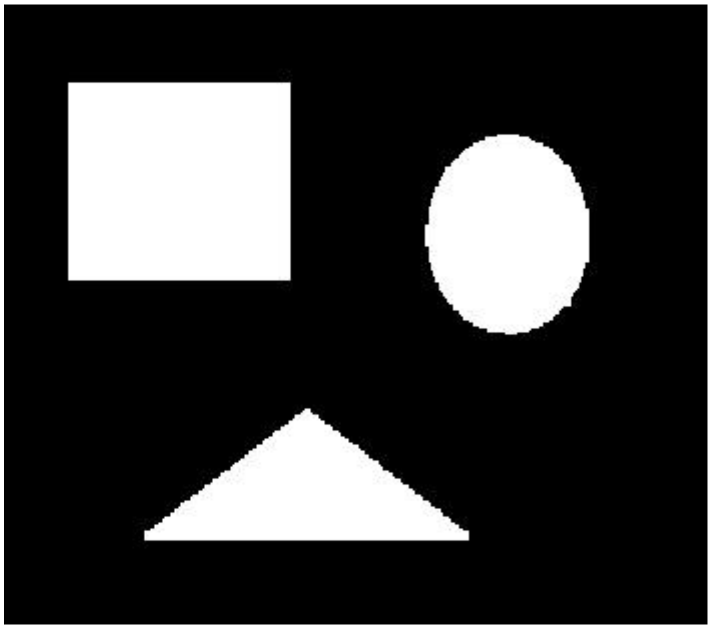
I have applied closing (dilation followed by erosion) and then opening (erosion followed by dilation).

```
img_1=erosion(dilation(dilation(erosion(img,mask),mask),mask),mask)
img_2=dilation(erosion(erosion(dilation(img,mask),mask),mask),mask)
cv2.imwrite('res_noise1.jpg',img_1) #opening followed by closing
cv2.imwrite('res_noise2.jpg',img_2) #closing followed by opening
```

Res_noise1



Res_noise2



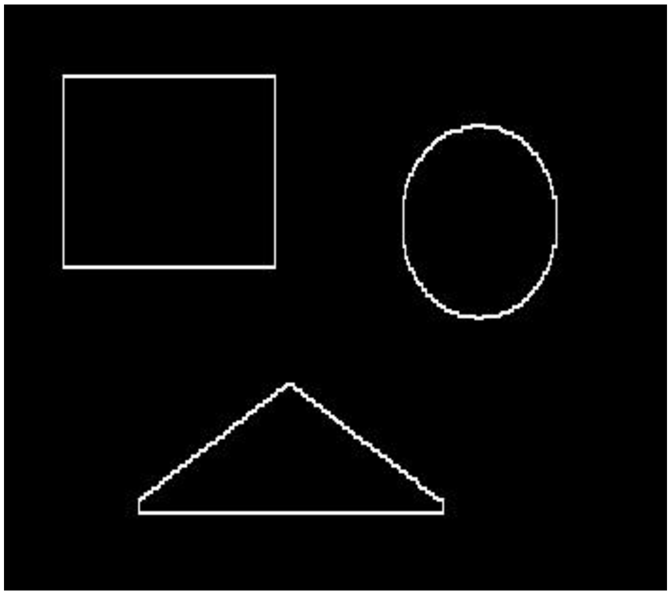
Part b)

On comparison of both the images though the results seem to be similar but they are not. The oval and triangle in both the images tend to differ because of how I have applied the morphological operator or kernel to the picture. The oval still has some protruding sides in res_noise2 whereas res_noise1 is much smoother. The same is with triangle but triangle in res_noise2 has better shaped corners as compared to the triangle in res_noise1.

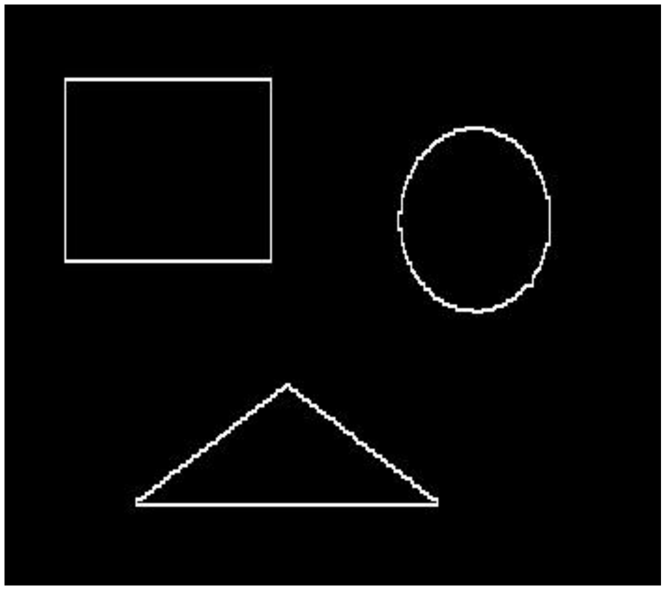
Part c)

I have in res_bound1 image dilated the res_noise1 image and subtracted from res_noise1 image and in res_bound2 image I have eroded the res_noise2 images and subtracted the from res_noise2 image itself to get the boundaries of both (res_noise1 and res_noise2) the images.

```
img_3=dilation(img_1,mask)

img_bound_1=img_3-img_1 #Image boundary using dilation

cv2.imwrite('res_bound1.jpg',img_bound_1)


img_4=erosion(img_2,mask)

img_bound_2=img_2-img_4 #Image boundary using erosion

cv2.imwrite('res_bound2.jpg',img_bound_2)
```



Res_bound1



Res_bound2

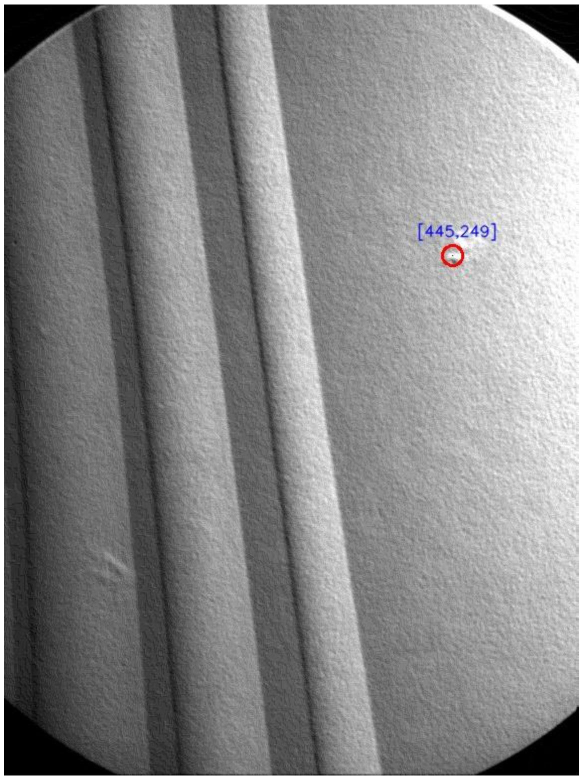
Question 2:

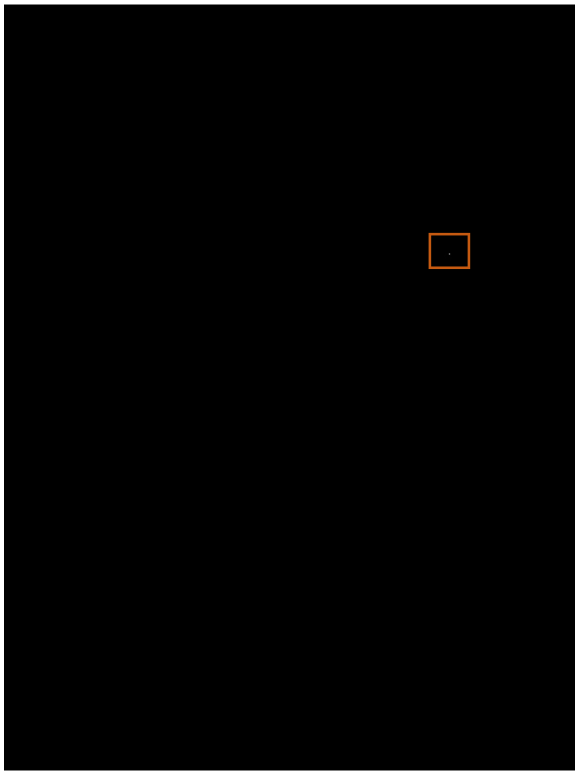
Part a)

```
import numpy as np
import cv2
img1 = cv2.imread('original_imgs/turbine-blade.jpg')
gray_scale = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)


kernel = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]], dtype = np.int8)
image_h, image_w = gray_scale.shape
kernel_h, kernel_w=kernel.shape
new_img = np.zeros(gray_scale.shape)
image_padded = np.zeros((gray_scale.shape[0] + 2, gray_scale.shape[1] + 2))
image_padded[1:-1, 1:-1] = gray_scale
img_1 = gray_scale.copy()
for i in range(image_h):
   for j in range(image_w):
      val=np.sum(kernel*image_padded[i:i+3,j:j+3])
      if val>600:
         new_img[i,j]=255
      else:
         new_img[i,j]=0
```

```
largest_num = new_img[0][0]
for row_idx, row in enumerate(new_img):
   for col_idx, num in enumerate(row):
      if num > largest_num:
         largest_num = num
         id_x=col_idx
         id_y=row_idx
large_val = largest_num
id_f_x=id_x
id_f_y=id_y
print(id_f_x,id_f_y)
```

```
cv2.circle(img1,(id_f_x,id_f_y), 10, (0,0,255), 2)

font=cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img1, "[445,249]", (410,230), font, 0.5, (255, 0, 0), 1, cv2.LINE_AA)

cv2.imwrite('porous_img.jpg',img1)
```

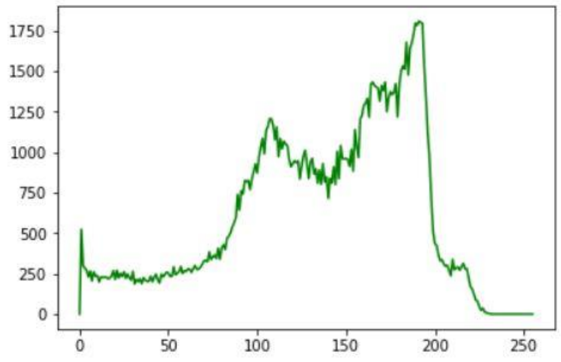




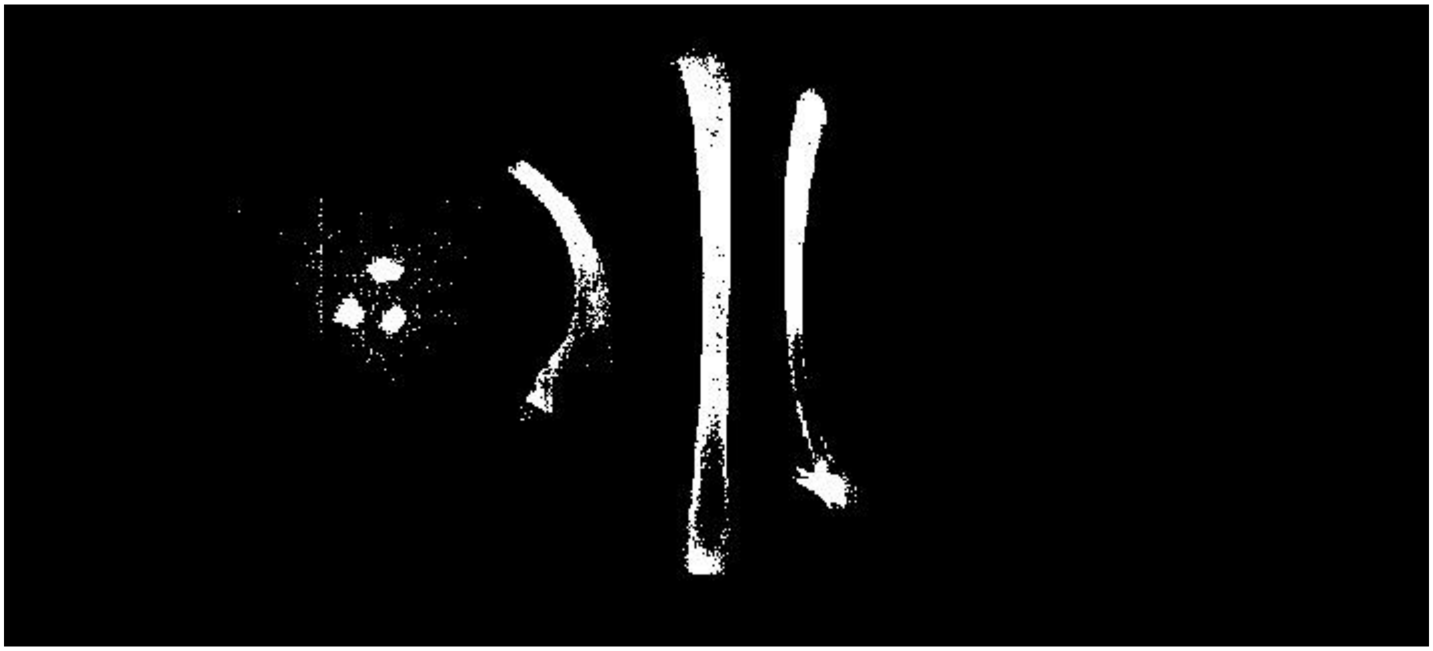
Above shown is the image with the porosity marked to it along with its coordinates.

Part b)

In this we had to choose the optimal threshold. For that I have drawn a histogram without using any hist libraries.

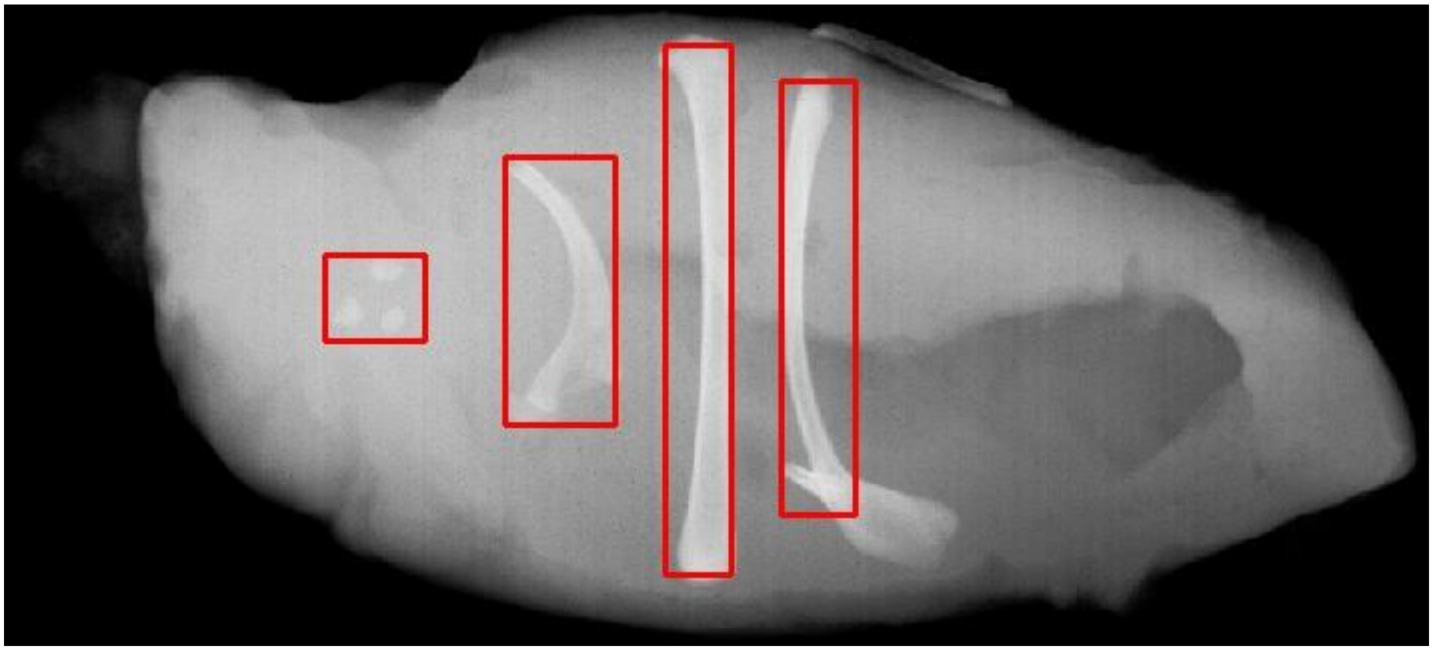


As can be seen from the plot that the pixels with intensities greater than 200 is the optimal threshold.



After choosing 200 as the threshold I got the image showing all the bones correctly in the image.

```
from matplotlib import pyplot as plt
img_segment = cv2.imread('original_imgs/segment.jpg')
segment = cv2.cvtColor(img_segment, cv2.COLOR_BGR2GRAY)
print(len(segment))
print(len(segment[0]))
arr = []
for i in range(len(segment)):
    for j in range (len(segment[0])):
        if segment[i][j]!=0:
            arr.append(segment[i][j])
count_arr=np.zeros([256,1])
for i in range(0,len(arr)):
    count_arr[arr[i]]=count_arr[arr[i]]+1
plt.plot(count_arr,color='green')
plt.show()
i, j = np.where(new_img == 255)
print(i,j)
```



The coordinates of 4 rectangles are:

1. (125,160),(125,210),(168,160),(168,210)
2. (76,250),(76,305),(210,250),(210,305)
3. (20,330),(20,363),(285,330),(285,363)
4. (38,388),(38,425),(255,388),(255,425)

Question 3:

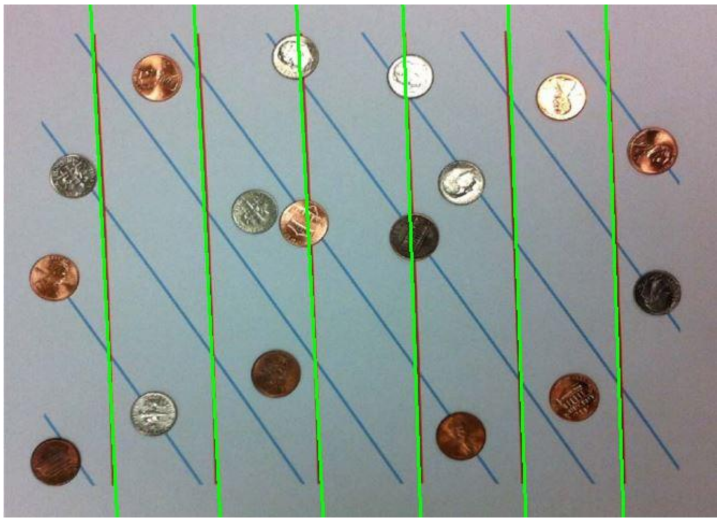
Part a)

```
def hough_lines_acc(img, theta_resolution=1):
    height, width = img.shape
    img_diagonal = np.ceil(np.sqrt(height*height + width*width))
    rhos = np.linspace(-img_diagonal, img_diagonal, img_diagonal*2)
    thetas = np.deg2rad(np.arange(-90, 90, theta_resolution))
    hough_acc = np.zeros((len(rhos), len(thetas)), dtype=np.uint8)
    y_id, x_id = np.nonzero(img) # find all edge (nonzero) pixel indexes
    for i in range(len(x_id)): # cycle through edge points
        x = x_id[i]
        y = y_id[i]
        for i in range(len(thetas)):
            rho = int((x * np.cos(thetas[i]) + y * np.sin(thetas[i])) + img_diagonal)
            hough_acc[rho, i] += 1
    return hough_acc, rhos, thetas


def hough_peaks(hough_acc, num_peaks, nhood_size):
    # loop through number of peaks to identify
    indicies = []
    hough_acc_1 = np.copy(hough_acc)
    for i in range(num_peaks):
        idmax = np.argmax(hough_acc_1)
        hough_acc_idmax = np.unravel_index(idmax, hough_acc_1.shape)
        indicies.append(hough_acc_idmax)
        idmax_y, idmax_x = hough_acc_idmax
        if (idmax_x - (nhood_size/2)) < 0:
            min_x = 0
        else:
            min_x = idmax_x - (nhood_size/2)
```

```
if ((idmax_x + (nhood_size/2) + 1) > hough_acc.shape[1]):
            max_x = hough_acc.shape[1]
        else:
            max_x = idmax_x + (nhood_size/2) + 1
        if (idmax_y - (nhood_size/2)) < 0:
            min_y = 0
        else:
            min_y = idmax_y - (nhood_size/2)
        if ((idmax_y + (nhood_size/2) + 1) > hough_acc.shape[0]):
            max_y = hough_acc.shape[0]
        else:
            max_y = idmax_y + (nhood_size/2) + 1
        for x in range(int(min_x), int(max_x)):
            for y in range(int(min_y), int(max_y)):
                hough_acc_1[y, x] = 0
                if (x == min_x or x == (max_x - 1)):
                    hough_acc[y, x] = 255
                if (y == min_y or y == (max_y - 1)):
                    hough_acc[y, x] = 255
    return indicies, hough_acc
```

```
def hough_lines_draw(img, indicies, rhos, thetas):
    for i in range(len(indicies)):
        rho = rhos[indicies[i][0]]
        theta = thetas[indicies[i][1]]
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

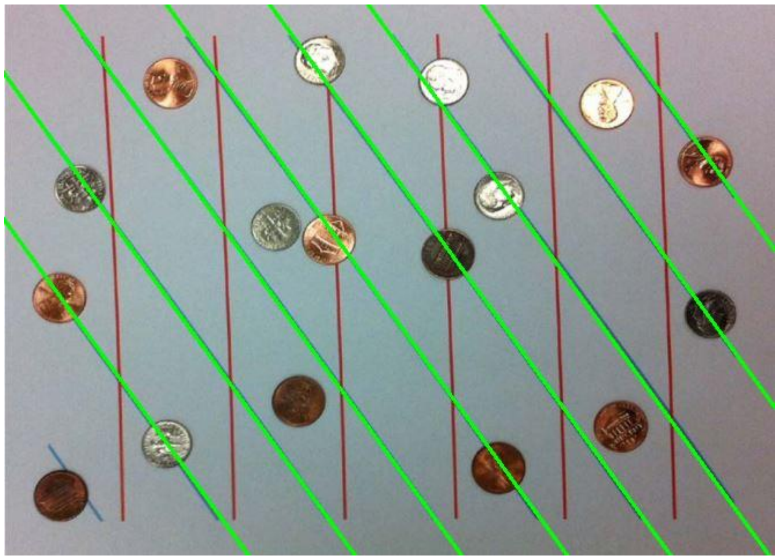
In part a, we had to detect all red lines. First I converted the image to where only the red channel is active and all other channels that is blue and green are zero. After doing that I blurred the image and found edges of the image. After that the Hough algorithm is used to fill in the accumulators and then with maximum votes pinning to each accumulator the peaks are selected and lines are drawn.



The algorithm was able to detect all 6 red lines.

Part b)

In part b, we had to detect all blue lines. First I converted the image to where only the blue channel is active and all other channels that is red and green are zero. After doing that I blurred the image and found edges of the image. After that the Hough algorithm is used to fill in the accumulators and then with maximum votes pinning to each accumulator the peaks are selected and lines are drawn.



The algorithm was able to detect 8 lines correctly.

References:

1. http://aishack.in/tutorials/hough-transform-basics/
2. https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html
3. https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html#morphology-1
4. https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#basic-threshold
5. https://github.com/alyssaq/hough_transform
6. https://classroom.udacity.com/courses/ud810/lessons/2870588566/concepts/34824487940923
7. https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html
8. https://docs.opencv.org/3.4/d1/db7/tutorial_py_histogram_begins.html