CSE -573 Intro to Computer Vision and Image Processing
Project 3

Manish Reddy Challamala
Department of Computer Science
University at buffalo
UBIT Name: manishre
Person Number : 50289714
manishre@buffalo.edu

December 3, 2018

1 Task1 - Morphology image processing

Abstract

The goal of this task is stated point wise below:

1. Remove the noise from the given image 'noise.jpg' by using the two Morphology image processing algorithm.
2. compare the output of above two resultant images and specify weather the two resultant images are same or not.
3. Extract the boundary of the two images and save the result.

1.1 Experimental set-up:

1. For this task, we are using the following libraries:
   1. Cv2, numpy, matplotlib
2. The morphology operations are stated below:
   1.1. Erosion - Erodes the image.
   1.2. Dilation - Dilates the image.
   1.3. Opening - It is nothing but Erosion followed by dilation operation.
   1.4. Closing - It is nothing but dilation followed by erosion operation.
3. The dilation operation, takes care of pepper noise.
4. The erosion operation, takes care of salt noise.
5. Algorithm for Erosion:
   5.1. consider a image matrix
   5.2. consider a structuring element. $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

5.3. Initialize a matrix with zero of size of image let it be Newimage
5.4. pad the original image.
5.5. Convolve the structuring element on the padded image and check weather the ones in structuring element overlaps only with the ones in padded image.
5.6. If the condition is True, update the newimage matrix with one and slide to the next position else zero.
5.7 Repeat the step 4.4 and 4.5 for all the pixel values in padded image.

6. Algorithm for dilation:
   6.1. Steps are same still step 4.4.
   6.2. Convolve the structuring element on padded image and perform a logical AND operation.
   6.3. If all the values of resultant Logical AND operation is zero then update the newimage matrix position with zero. else with one.
   6.4. Repeat the step 5.2 and 5.3 for all the pixel values in padded image.

1.2 Code:

```
#%% Libraries
import numpy as np
np.set_printoptions(threshold=np.inf)
import cv2
import copy
#%% Functions

def padding(image):
    width = image.shape[1]
    height = image.shape[0]
    pad_image = np.asarray([[255 for x in range (width+2)] for y in range (height+2)])
    pad_image[1:-1,1:-1] = image
#    cv2.imshow('padded',np.asarray(pad_image,dtype = 'uint8'))
#    cv2.waitKey(0)
#    cv2.destroyAllWindows()
    return np.asarray(pad_image,dtype = 'uint8')

def dilation(image,SE):
    D = image.copy()
    width = image.shape[1]
    height = image.shape[0]
    for i in range(0,height):
        for j in range(0,width):
            ao =[]
            for kr in range(0,3):
                for kc in range(0,3):
                     #ao.append(SE[kr,kc]*image[i-kr,j-kc])
                    a = image[i-kr,j-kc]
                    S_E = SE[kr,kc]
                    ao.append(a & S_E)
            D[i,j]=max(ao)
```

```
#    cv2.imshow('Dilation image',np.asarray(D,dtype ='uint8'))
#    cv2.waitKey(0)
#    cv2.destroyAllWindows()
    return np.asarray(D,dtype='uint8')
def erosion(image,SE):
    E = image.copy()
    width = image.shape[1]
    height = image.shape[0]
    for i in range(0,height):
        for j in range(0,width):
            ao =[]
            for kr in range(0,3):
                for kc in range(0,3):
                    a = image[i-kr,j-kc]
                    S_E = SE[kr,kc]
                    if (S_E==255 and a==S_E):
                        ao.append(255)
                    else:
                        ao.append(0)
            E[i,j]=min(ao)
#    cv2.imshow('erosion image',np.asarray(E,dtype ='uint8'))
#    cv2.waitKey(0)
#    cv2.destroyAllWindows()
    return np.asarray(E,dtype ='uint8')

def similarity(im_o, im_c):
mse = np.sum((im_o.astype("float") - im_c.astype("float")) ** 2)
mse /= float(im_o.shape[0] * im_o.shape[1])
return mse

def boundary(img_e,kernel):
    img_ero = erosion(img_e,kernel)
    img_dil = dilation(img_e,kernel)
    img_bound = np.subtract(img_dil,img_ero)
    return img_bound
#def opening():

#%% MAIN LOOP

#%% Loading the Images
image = cv2.imread('noise.jpg',0)
img = image.copy()
padded_image = padding(img)
cv2.imshow('orginal image',image)
cv2.waitKey(0)
cv2.imshow('Padded image',padded_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
print('dimensions',padded_image.shape)
# structring Element
```

```
kernel = np.matrix([[255,255,255],[255,255,255],[255,255,255]])
#%% ====================PART 1=============================================

# Closing
img0 = dilation(img,kernel)
img2= erosion(img0,kernel)
img3 = erosion(img2,kernel)
cv2.imwrite('res_noise1.jpg',np.asarray(img3))
# Opening
img4 = erosion(img,kernel)
img5 = dilation(img4,kernel)
img6 = dilation(img5,kernel)
cv2.imwrite('res_noise2.jpg',img6)
# displaying the images
cv2.imshow('closing',img3)
cv2.waitKey(0)
cv2.imshow('opening',img6)
cv2.waitKey(0)
cv2.destroyAllWindows()

#%%======================== PART 2 ============================================

# Finding comparisons between two images  by using mean square error
# Condtion1: if the both images are same then the error will be 0
# Conditon2: if the both images are not same then the value of error will be increasing
#-----------------------------
# For the variable o below we are caluclating
the comparing the logic by passing same image
o = similarity(img,img)
# Caluclaing the similarity between two resultant images
s = similarity(img3,img6)
print('s',s)
if o == 0:
    print('The two images are same')
else :
    print('The two images are not same')
if s == 0:
    print('The two images are same')
else :
    print('The two images are not same')

#%%================================ PART 3 ====================================
## The difference between dilation and erosion of a
image results in the boundary of a image
cv2.imwrite('res_bound1.jpg',img_boundo)
cv2.imwrite('res_bound2.jpg',img_boundc)
```

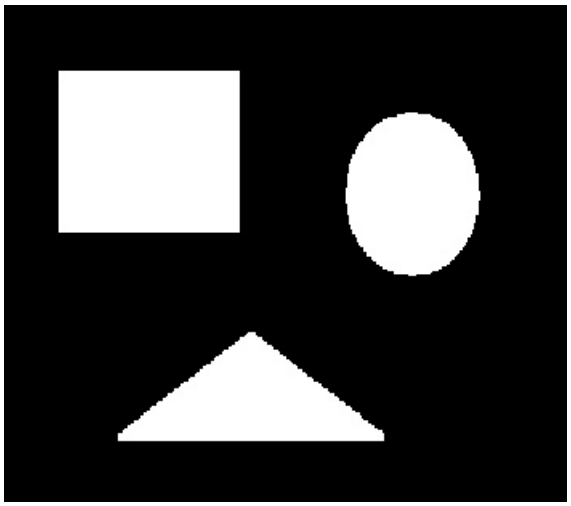
1.3 Output:

Output of 1b:

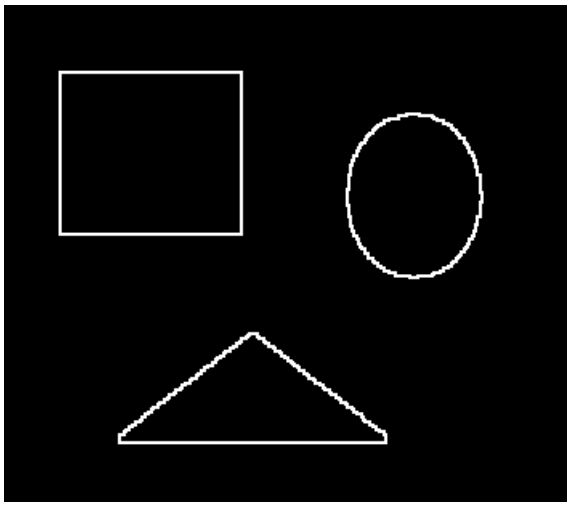
1. Comparing the two images by using Mean square error
2. Condtion1: if the both images are same then the error will be 0
3. Conditon2: if the both images are not same then the value of error will be increasing.
4. The images which are resulted from task [1a] are different.

Output of 1a and 1c:
Images after removal of noise.

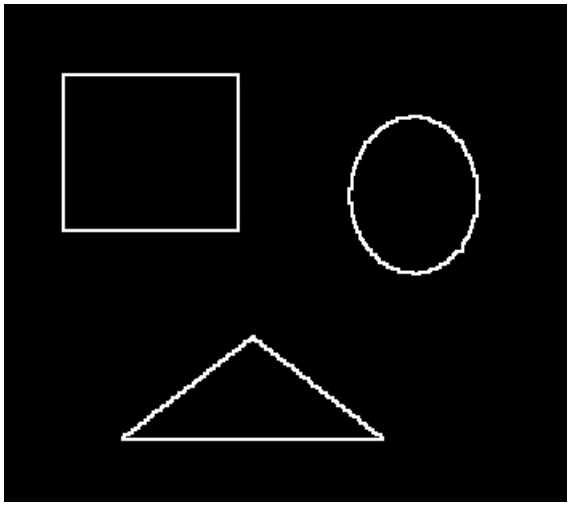
fig[1] : *Noise Free image by using closing operation*



$fig[2]:$ *Noise Free image by using opening operation*



$fig[3]:$ *Boundary extraction of* $resnoise1$ *image*



$fig[4]:$ *Boundary extraction of* $resnoise2$ *image*

2 Task 2 - Image Segmentation and Point detection.

Abstract

The goal of this task is stated point wise below:

1. Detect the porosity by using the point detection algorithm and point the coordinates of the detected point on the image.
2. Segment the object from background by choosing a threshold value and draw a rectangular bounding box around the object.

2.1 Experimental set-up:

1. For this task, we are using the following libraries:
   1. Cv2, numpy, matplotlib
2. Point Detection algorithm is a common way to calculate the gray-level discontinuities of a image by running a mask through the image. The formula to measure the weighted difference is given below:

$$R = \sum_{i=1}^{9} W_i Z_i$$

3. The kernel for the point detection is $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$.
4. Algorithm for Point Detection:
   4.1. Consider the image matrix.
   4.2. Consider the kernel shown above in step 3.
   4.3. Initialize a matrix with zeros of size of the image. Let it be the new image.
   4.4. Convolve the kernel on the image and calculate the sum of the neighbouring elements by using the above formula and update the sum value in the respective pixel in the new image.
   4.5. Repeat the step 4.4 for all the pixel values in original image.
5. Algorithm for segmenting the object and background:
   5.1. Plot the histogram by finding out the intensities of all the pixel value in the image.
   5.2. choose a threshold value manually and threshold the image by using that threshold value.
   5.3. Find out the pixel coordinates of the segmented object in the image and draw the bounding box around the object.

2.2 Code:

Task 2a:

```
"""
Created on Sun Dec  2 20:14:24 2018
@author: manishre
"""
import cv2
import numpy as np
import scipy.misc
#%% FUNCTIONS
def con(img,kernel):
    img1 =img.copy()
    height,width = img.shape
    for x in range (0,height):
        for y in range (0,width):
            su_x=0
            for k in range (0,3):
                for l in range (0,3):
                    a= img[x-k,y-l]
                    w1 = kernel[k][l]
                    su_x = su_x + (w1*a);
            img1[x,y] = su_x

    print(img1.shape)
    return img1

def Thresholding(img,tv,threshold):
    img_T = img.copy()
    height,width = img.shape
    if(threshold=='b'):
        for i in range(0,height):
            for j in range(0,width):
                if (img_T[i,j]>tv):
                    img_T[i,j] =255
                else:
                    img_T[i,j] =0
    else:
        for i in range(0,height):
            for j in range(0,width):
                if (img_T[i,j]>tv):
                    img_T[i,j] =0
                else:
                   img_T[i,j] =255
    return img_T
#%% IMREAD
image = scipy.misc.imread('point1.jpg',1)
#image = cv2.imread('point1.jpg',0)
img1 = image.copy()
kernel = [[-1,-1,-1],
```

```
          [-1,8,-1],
          [-1,-1,-1]]
#%%
image_con = con(img1,kernel)
#%%
image_con=  np.abs(image_con)/np.max(np.abs(image_con))
#%%image_con
image_thre = Thresholding(image_con,0.5,'b')
#%% Point Detection
point_coordinates =np.unravel_index(image_thre.argmax(),image_thre.shape)
print('The point is detected at:',point_coordinates)
cv2.circle(image_thre,(point_coordinates[1],point_coordinates[0]),
radius = 12, color=(255,255,255), thickness = 1)
cv2.putText(image_thre, str(point_coordinates),(point_coordinates[1]+6,
point_coordinates[0]+10), cv2.FONT_HERSHEY_SIMPLEX, 0.4,(255, 255, 255), 1, cv2.LINE_AA)
#%% Saving the output image
#scipy.misc.imsave('img_con.jpg',image_con)
cv2.imwrite('task2a_output.jpg',image_thre)
cv2.imshow('img_o',image_con)
cv2.waitKey(0)
cv2.imshow('img_t',image_thre)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

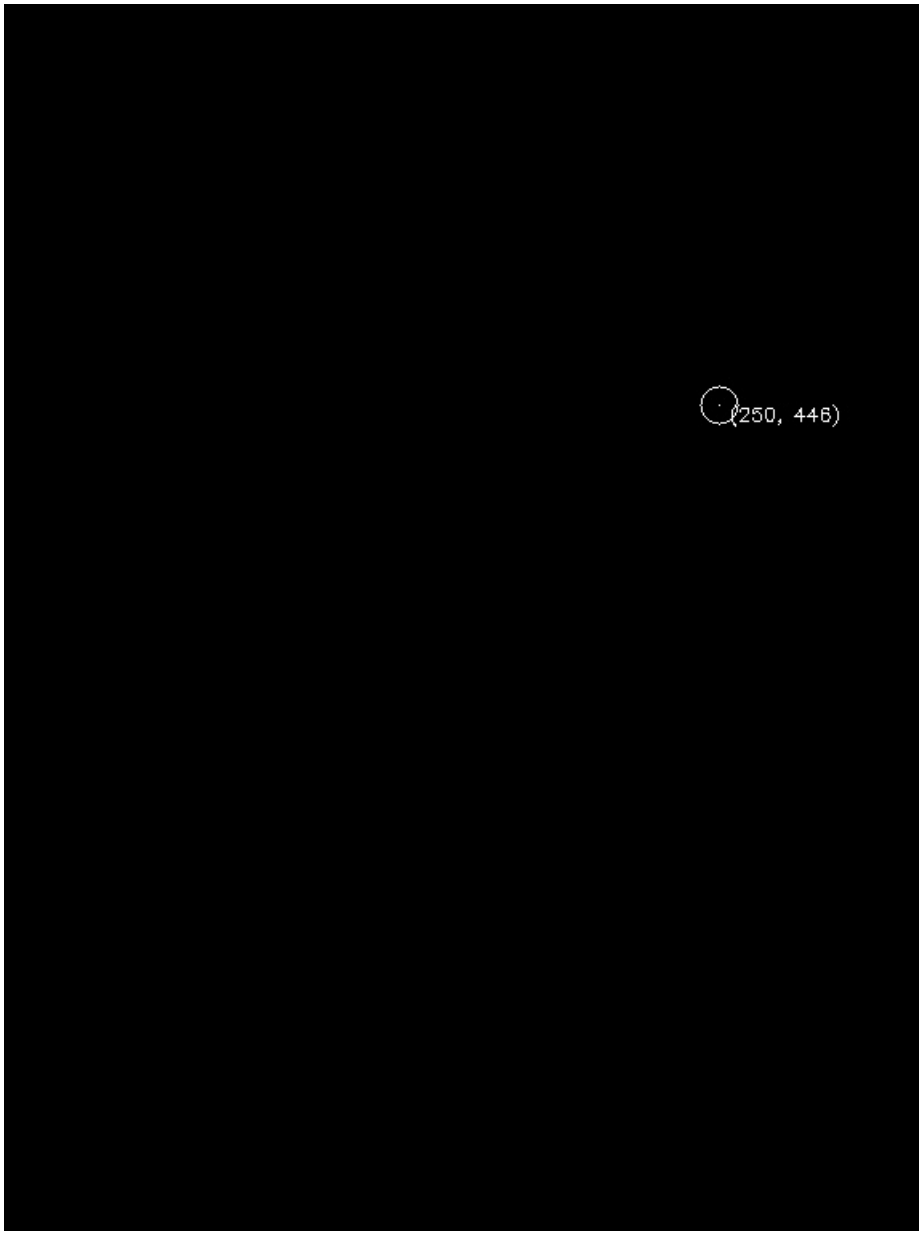
Task 2b:

```
#%% TASK2B
image_2B = cv2.imread('segment.jpg',0)
#%%
intensity = np.zeros(256)
h,w = image_2B.shape
for i in range(0,h):
    for j in range(0,w):
        intensity[image_2B[i,j]] += 1
#%%
    intensity = intensity.astype(int)
    print(intensity)
#%%
    intensity[0]=0
    intensity[0]=max(intensity)+100
#%%
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots()
ax.plot(intensity)
plt.xlabel('pixel value')
plt.ylabel('intensity')
plt.savefig('task2a_histogram.jpg')
plt.show()
#%%
np.set_printoptions(threshold=np.nan)
```

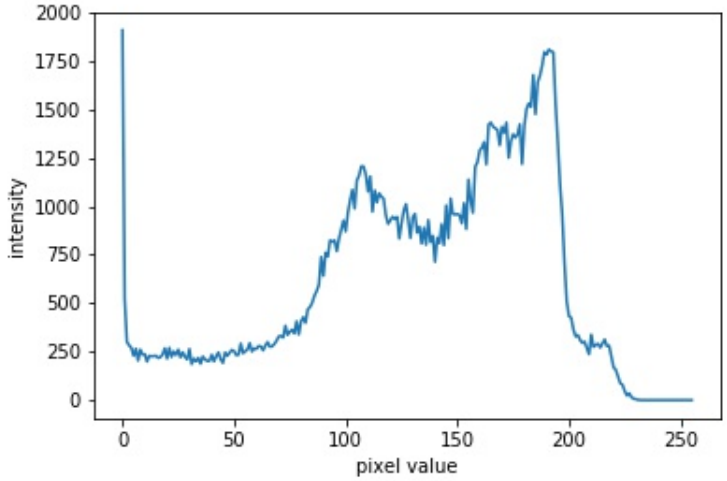
```
image_output=image_2B.copy()
#print(image_2B)
print('====================')

image_i = Thresholding(image_2B,205,'b')
image_i1 = Thresholding(image_2B,203,'w')
#%%
#%% Getting Co-ordinates for Bounding Box
row_max= 321
row_min = 0
col_max = 712
col_min =0
for i in range(0,h):
    for j in range(0,w):
       if image_i[i][j]>205:
          if row_max>i:
              row_max =i
          elif col_max > j:
              col_max =j
          elif row_min < i:
              row_min = i
          elif col_min < j:
              col_min =j
#%% Plotting and labelling Bounding Box
cv2.rectangle(image_i,(col_max,row_max),(col_min,row_min),
(255,255,255),2)
cv2.putText(image_i, str((row_min,col_min)),(col_min+6,row_min+10),
cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1, cv2.LINE_AA)
cv2.putText(image_i, str((row_max,col_max)),(col_max-80,row_max+10),
cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1, cv2.LINE_AA)
cv2.putText(image_i, str((row_min,col_max)),(col_max-80,row_min+10),
cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1, cv2.LINE_AA)
cv2.putText(image_i,str((row_max,col_min)),(col_min+6,row_max+10),
cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1, cv2.LINE_AA)
#%%
# show the output image
cv2.imwrite('task2b_output.jpg',image_i)
cv2.imshow('image_2B',image_2B)
cv2.waitKey(0)
cv2.imshow('image_output',image_i)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.3 Output:

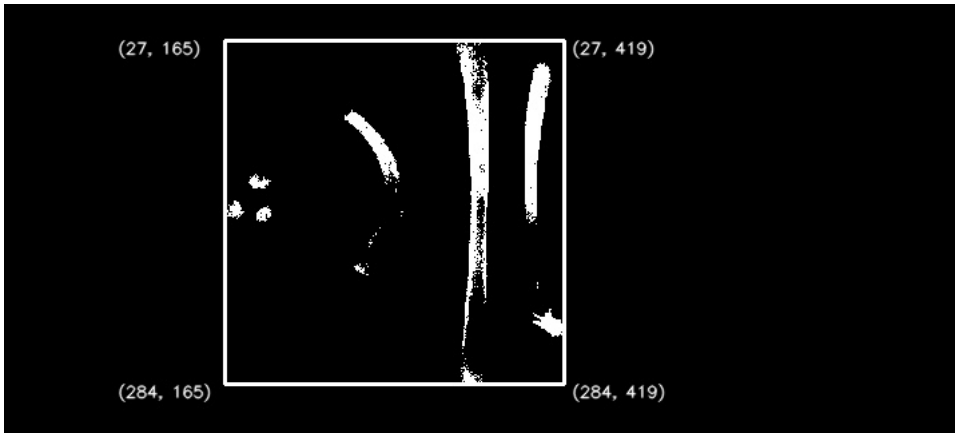
Output for Point Detection: Images task2aoutput.jpg and task2boutput.jpg



fig[1] : *Output Image for Point Detection*



$fig[2]:$ *Histogram of segment image*



$fig[3]:$ *Output image for Image Segmentation*

3 Task3 - Hough transform

Abstract

The goal of this task is stated point wise below:

1. Detect all the red lines in an image by using Hough transform algorithm and specify how many red lines are detected.
2. Use the same above algorithm and detect all the diagonal blue lines in a image.

3.1 Experimental setup:

1. Hough transform is a technique used to isolate features of a particular shape in an image such as lines, circles an ellipse and etc.,
2. Parametric line notion of hough transform is given by :

$$xcos\theta + ysin\theta = r$$

2.1. where r is the length of a normal from origin to this line and θ is orientation of r with respect to point (x,y) on line.
2.2. Here we are converting the pixel co-ordinates of an image to a point in hough space. If we plot all the possible (r,θ) values to point in hough space, The (x,y) point in the Cartesian image space maps to a curve in hough space, So this point-to-curve transformation is called hough transformation for straight line.

3. Algorithm for Hough Transformation:
3.1. Each (X_i, Y_i) coordinates in image space is converted into a discrete (r, θ) curve and the accumulator which line along the curve are incremented.
3.2. Transverse through the accumulator array and find out the peaks which results that there exists a straight line in the image.

3.2 code:

```
"""
Created on Fri Nov 30 17:43:37 2018
@author: manishre
"""
#%% LIBRARIES
import cv2
import numpy as np
import copy
import math
#%%Reading the image
image = cv2.imread('A:\\classes\\Intro to Computer vision and
Image Processing\\projrct\\project3\\task3\\hough.jpg',0)
width = image.shape[1]
height = image.shape[0]
print("Width :",width)
```

```
print('Height:',height)
#%% Functions
def edge_detection(pad_image):
    s_x = [[1,0,-1],[2,0,-2],[1,0,-1]]
    s_y =[[1,2,1],[0,0,0],[-1,-2,-1]]
    #Flip Operations
    for i in range (0,3):
        temp =s_x[i][0]
        s_x[i][0] =s_x[i][2]
        s_x[i][2] =temp
    print('s_x',s_x)
    for y in range (0,3):
        temp = s_y[0][y]
        s_y[0][y] =s_y[2][y]
        s_y[2][y] =temp
    img_x = np.asarray([[0 for x in range(0,width)] for y in range(0,height)])
    img_y = np.asarray([[0 for x in range(0,width)] for y in range(0,height)])
    # Convolution
    for x in range (0,height):
        for y in range (0,width):
            su_x=0
            su_y =0
            for k in range (0,3):
                for l in range (0,3):
                    a= pad_image[x-k,y-l]
                    w1 = s_x[k][l]
                    w2 = s_y[k][l]
                    su_x = su_x + (w1*a);
                    su_y = su_y + (w2*a)
            img_x[x,y] = abs(su_x)
            img_y[x,y] = abs(su_y)
    print(img_x)
    d = img_x.shape
    print('edge_'+str(i)+'width, height',d)
    return np.array(img_x,dtype='uint8'),np.array(img_y,dtype='uint8')
def hough_transform(img_bin):
  nR,nC = img_bin.shape
  theta = np.linspace(-90.0, 0.0, np.ceil(90.0) + 1.0)
  theta = np.concatenate((theta, -theta[len(theta)-2::-1]))
  print('1start')
  D = np.sqrt((nR - 1)**2 + (nC - 1)**2)
  q = np.ceil(D)
  nrho = 2*q + 1
  rho = np.linspace(-q, q, nrho)
  H = np.zeros((len(rho), len(theta)))
  for rowIdx in range(nR):
    for colIdx in range(nC):
      if img_bin[rowIdx, colIdx]:
        for thIdx in range(len(theta)):
          rhoVal = colIdx*np.cos(theta[thIdx]*np.pi/180.0)
```

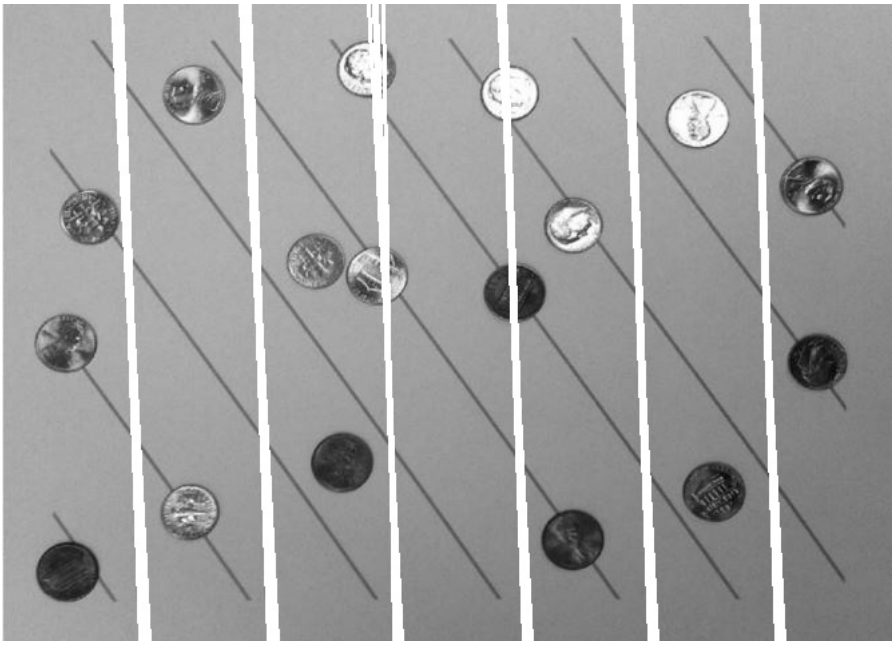
```
          + \ rowIdx*np.sin(theta[thIdx]*np.pi/180)
          rhoIdx = np.nonzero(np.abs(rho-rhoVal)
          == np.min(np.abs(rho-rhoVal)))[0]
          H[rhoIdx[0], thIdx] += 1
  print('1end')
  return rho, theta, H
#%%Padding the image
pad_image = np.asarray([[0 for x in range (width+2)]
for y in range (height+2)])
pad_image[1:-1,1:-1] = image
cv2.imshow('padded',np.asarray(pad_image,dtype = 'uint8'))
cv2.waitKey(0)
#g_x = np.asarray([[0 for x in range(0,width)] for y in range(0,height)])
g_x,g1 = edge_detection(pad_image)
cv2.imshow('g_x',abs(g_x))
cv2.waitKey(0)
cv2.imshow('g_y',abs(g1))
cv2.waitKey(0)
cv2.destroyAllWindows()
#g_y = np.asarray([[0 for x in range(0,width)] for y in range(0,height)])
#%%
im =image.copy()
edges = cv2.Canny(im, threshold1 = 0, threshold2 = 50, apertureSize = 3)
r,t,h = hough_transform(edges)
#%%
P =r
angle = t
space =h
#%%
def sort(P_space, n, rhos, thetas):
  print('2start')
  unstack = list(set(np.hstack(P_space)))
  unsorte = sorted(unstack, key = lambda n: -n)
  points = [(np.argwhere(P_space == x)) for x in unsorte[0:n]]
  rho_theta = []
  x_y = []
  for i in range(0, len(points), 1):
    co = points[i]
    for i in range(0, len(co), 1):
      n,m = co[i] # n by m matrix
      rho = rhos[n]
      theta = thetas[m]
      rho_theta.append([rho, theta])
      x_y.append([m, n]) # just to unnest and reorder coords_sorted
  print('2end')
  return [rho_theta[0:n], x_y]
#%%
  val,val1 = sort(space, 70, P, t)
  #%%
  print(val)
```

```
  #%% Voting and
def bound(pt, ymax, xmax):
  x, y = pt

  if x <= xmax and x >= 0 and y <= ymax and y >= 0:
    return True
  else:
    return False

def roundnum(tup):
  x,y = [int(round(num)) for num in tup]
  return (x,y)

def draw_Lines(target_im, pairs):
    tar0 = target_im.copy()
    tar1 = target_im.copy()
    h,w = np.shape(target_im)
    th0=[]
    print('len',len(pairs))
    pts0X=[]
    pts1X=[]
    pts0Y=[]
    pts1Y=[]
    for i in range(1,len(pairs),1):
        point =pairs[i]
        rho = point[0]
        theta = point[1]
        theta = point[1] *np.pi/180
        m = -np.cos(theta) / np.sin(theta)
        b = rho / np.sin(theta)
        left = (0, b)
        right = (w, w * m + b)
        top = (-b / m, 0)
        bottom = ((h - b) / m, h)
        #print(m)
        if (m>3):
           # print('greater')
           print('top',top)
           print('left',left)
           print('right',right)
           print('bottom',bottom)
           pts =[pt for pt in [left, right, top, bottom] if bound(pt, h, w)]
           pts0X.append(roundnum(pts[0]))
           pts1X.append(roundnum(pts[1]))
        else:
            #print('less')
            pts1 =[pt for pt in [left, right, top, bottom] if bound(pt, h, w)]
            pts0Y.append(roundnum(pts1[0]))
            pts1Y.append(roundnum(pts1[1]))
```

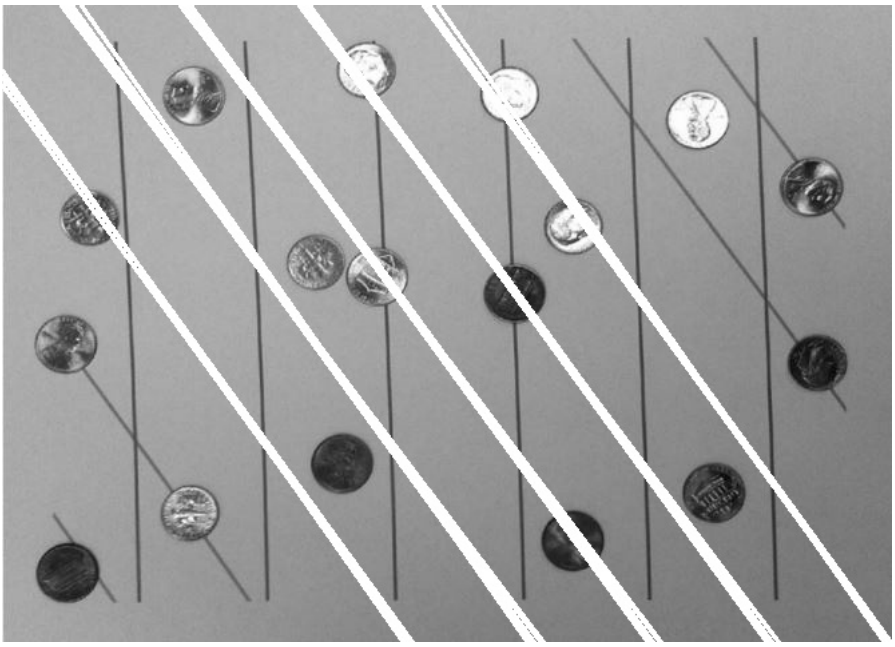
```
    return pts0X,pts1X,pts0Y,pts1Y

 #%%
p0,p1,p2,p3= draw_Lines(image,val)
#%%
imgr =image.copy()
imgb =image.copy()
for i in range(0,len(p0)):
 cv2.line(imgr, p0[i],p1[i] , (255,255,255), 2)
 cv2.line(imgb,p2[i],p3[i],(255,0,0),2)
#%% Output Images
cv2.imwrite('red_line.jpg',imgr)
cv2.imshow('red lines',imgr)
cv2.waitKey(0)
cv2.imwrite('blue_lines.jpg',imgb)
cv2.imshow('blue lines',imgb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.3 Output:

output for Hough Transform: Images redline.jpg and bluelines.jpg



fig[1] : *Red lines Detection using Hough Transform*



$fig[2]:$ *blue lines Detection using Hough Transform*

4 References:

1. https://www.imageeprocessing.com/2012/09/image-erosion-without-using-matlab.html –¿ nice Explanation
2. https://www.imageeprocessing.com/p/table-of-contents.html
3. https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/
4. https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.html
5. http://www.theobjects.com/dragonfly/deep-learning.html
6. https://nabinsharma.wordpress.com/2012/12/26/linear-hough-transform-using-python/