EN2550_Exercise7_190621M

March 23, 2022

0.1 Exercise-07

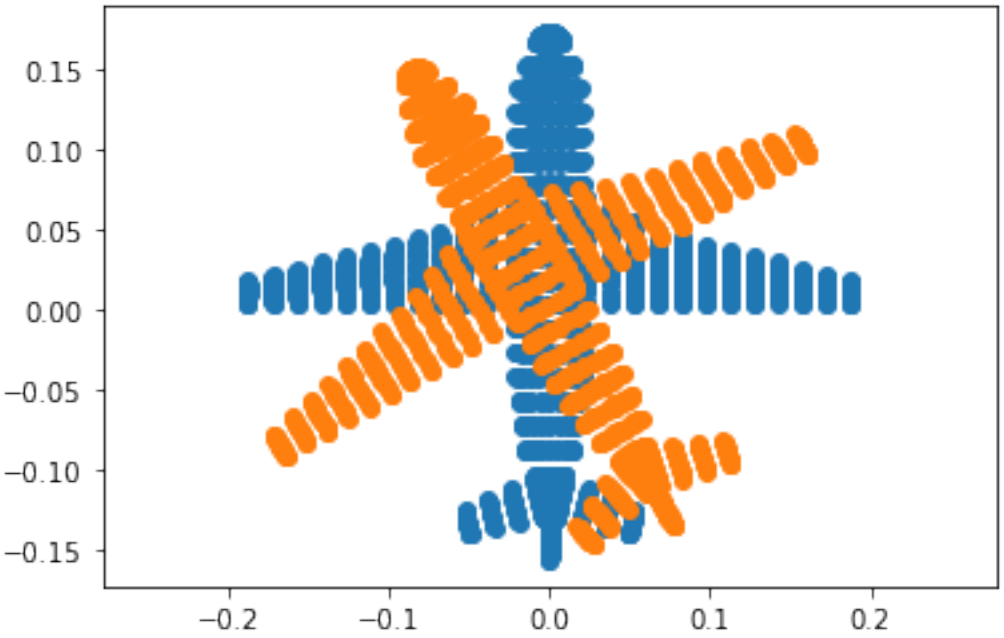
0.2 Index No - 190621M

0.3 Name - K. Thanushan

0.3.1 Question 1.

[ ]:
```
import numpy as np
from plyfile import PlyData, PlyElement
import matplotlib.pyplot as plt

pcd = PlyData.read('airplane.ply')
assert pcd is not None

points = np.concatenate((pcd['vertex']['x'].reshape(1, -1), pcd['vertex']['y'].
 ↪reshape(1, -1), pcd['vertex']['z'].reshape(1, -1)), axis=0)
points = points - np.mean(points, axis=1).reshape(3,1)

ones = np.ones((1, points.shape[1]))
X = np.concatenate((points, ones), axis = 0)

R = np.array([[1,0,0],[0,1,0],[0,0,1]])
K = np.array([[1,0,0],[0,1,0],[0,0,1]])
t = np.array([[0], [0], [-4000]])

P1 = K@np.concatenate((R,t), axis = 1)

R2 = np.array([[0.8660,-0.5,0],[0.5,0.8660,0],[0,0,1]])
K2 = np.array([[1,0,0],[0,1,0],[0,0,1]])
t2 = np.array([[0], [0], [-4000]])

P2 = K2@np.concatenate((R2,t2), axis = 1)
x1 = P1@X
x1 = x1/x1[2,:]
x2 = P2@X
x2 = x2/x2[2,:]

fig, ax = plt.subplots(1,1, sharex = True, sharey=True)
```

```
ax.scatter(x1[0,:], x1[1, :])
ax.scatter(x2[0,:], x2[1, :])
ax.axis('equal')
plt.show()
```

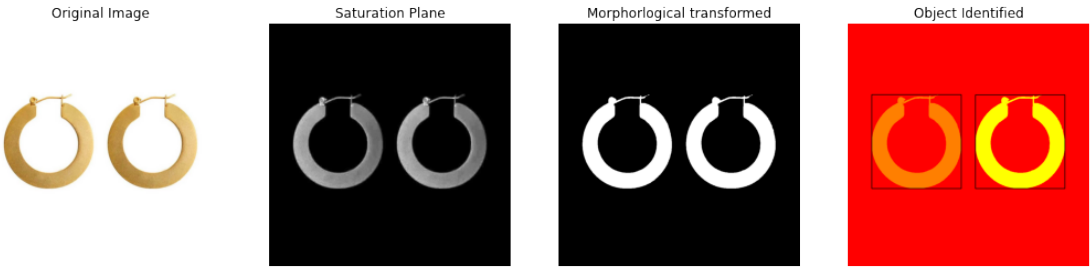


0.3.2 Question 3.

[ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

Image = cv.imread("earrings.jpg", cv.IMREAD_COLOR)
assert Image is not None
hsv = cv.cvtColor(Image, cv.COLOR_BGR2HSV)
th, bw = cv.threshold(hsv[:, :, 1], 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

#remove dots in the object foreground using closing
w = 5
kernel = np.ones((w,w), np.uint8)
opened = cv.morphologyEx(bw, cv.MORPH_CLOSE, kernel)

retval, labels, stats, centroids = cv.connectedComponentsWithStats(bw)
colormapped = cv.applyColorMap((labels/np.amax(labels)*255).astype('uint8'), cv.
 ↪COLORMAP_AUTUMN)
z = 720 #mm
```

```
f = 8 #mm

for i, s in enumerate(stats):
    if i !=0:
        print('Item', i, "area in pixels = ", s[4])
        print('Item', i, "area in mm^2 = ", s[4]*(2.2e-3)**2*(z*z)/(f*f))
        colormapped = cv.rectangle(colormapped, (s[0], s[1]), (s[0]+s[2],␣
 ↪s[1]+s[3]), (0, 0, 0), 2)

fig, ax = plt.subplots(1,4, figsize = (18,6))
Imageplot = cv.cvtColor(Image, cv.COLOR_BGR2RGB)
ax[0].imshow(Imageplot)
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(hsv[:,:,1], cmap = 'gray')
ax[1].set_title('Saturation Plane')
ax[1].axis('off')
Imageplot2 = cv.cvtColor(opened, cv.COLOR_BGR2RGB)
ax[2].imshow(Imageplot2)
ax[2].set_title('Morphorlogical transformed')
ax[2].axis('off')
Imageplot3 = cv.cvtColor(colormapped, cv.COLOR_BGR2RGB)
ax[3].imshow(Imageplot3)
ax[3].set_title('Object Identified')
ax[3].axis('off')
plt.show()
```

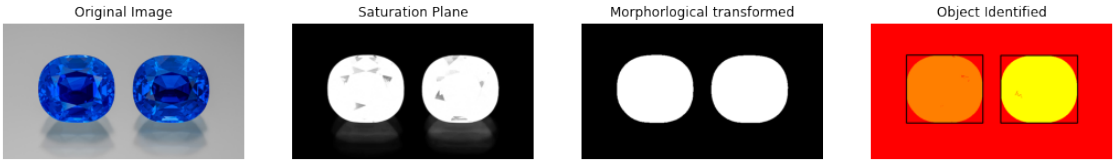
```
Item 1 area in pixels =  59143
Item 1 area in mm^2 =  2318.642172
Item 2 area in pixels =  59211
Item 2 area in mm^2 =  2321.3080440000003
```



[ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
```

```
Image = cv.imread("sapphire.jpg", cv.IMREAD_COLOR)
assert Image is not None
hsv = cv.cvtColor(Image, cv.COLOR_BGR2HSV)
th, bw = cv.threshold(hsv[:, :, 1], 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

#remove dots in the object foreground using closing
w = 5
kernel = np.ones((w,w), np.uint8)
opened = cv.morphologyEx(bw, cv.MORPH_CLOSE, kernel)

retval, labels, stats, centroids = cv.connectedComponentsWithStats(bw)
colormapped = cv.applyColorMap((labels/np.amax(labels)*255).astype('uint8'), cv.
 ↪COLORMAP_AUTUMN)
z = 720 #mm
f = 8 #mm

for i, s in enumerate(stats):
    if i !=0:
        print('Item', i, "area in pixels = ", s[4])
        print('Item', i, "area in mm^2 = ", s[4]*(2.2e-3)**2*(z*z)/(f*f))
        colormapped = cv.rectangle(colormapped, (s[0], s[1]), (s[0]+s[2],␣
 ↪s[1]+s[3]), (0, 0, 0), 2)

fig, ax = plt.subplots(1,4, figsize = (18,6))
Imageplot = cv.cvtColor(Image, cv.COLOR_BGR2RGB)
ax[0].imshow(Imageplot)
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(hsv[:,:,1], cmap = 'gray')
ax[1].set_title('Saturation Plane')
ax[1].axis('off')
Imageplot2 = cv.cvtColor(opened, cv.COLOR_BGR2RGB)
ax[2].imshow(Imageplot2, cmap = 'gray')
ax[2].set_title('Morphorlogical transformed')
ax[2].axis('off')
Imageplot3 = cv.cvtColor(colormapped, cv.COLOR_BGR2RGB)
ax[3].imshow(Imageplot3)
ax[3].set_title('Object Identified')
ax[3].axis('off')
plt.show()
```

```
Item 1 area in pixels =  30056
Item 1 area in mm^2 =  1178.315424
Item 2 area in pixels =  30066
Item 2 area in mm^2 =  1178.707464
```



0.3.3 Question 4.

[ ]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
file_name = 'allenkeys.jpg'
im = cv. imread(file_name , cv.IMREAD_REDUCED_GRAYSCALE_2)
assert im is not None
canny = cv.Canny(im , 50 , 150)

#Copy edges to the images that will display the results in BGR
canny_color = cv.cvtColor(canny, cv.COLOR_GRAY2BGR)

lines = cv.HoughLines(canny, 1 ,np.pi/180, 170, None, 0, 0)

if lines is not None :
    for i in range(0, len(lines)):
        rho = lines[i][0][0]
        theta = lines[i][0][1]
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b * rho
        pt1 = (int(x0 + 1000*(-b)),int(y0 + 1000*(a)))
        pt2 = (int(x0 + 1000*(-b)),int(y0 + 1000*(a)))
        cv.line(canny_color, pt1, pt2,(0, 0, 255), 1,cv.LINE_AA)


fig, ax = plt.subplots(1,4, figsize = (18,6))
ax[0].imshow(im, cmap = 'gray')
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(canny, cmap = 'gray')
ax[1].set_title('Edge Detected Image')
ax[1].axis('off')
ax[2].imshow(canny_color, cmap = 'gray')
ax[2].set_title('Lines Detected Image')
ax[2].axis('off')
```

```
r = cv.selectROI('Image', canny_color, showCrosshair = True , fromCenter =␣
 ↪False)
print(r)

x0, y0 = int(r[0] + r[2]/2), int(r[1] + r[3]/2)
m = b/a #Gradient
m = np.tan(np.median(lines[ : , 0 , 1]))
c = y0 - m*x0 # Intercept


cv.line(canny_color, (0, int(c)), (im.shape[0], int(m*im.shape[0] + c)), (0,␣
 ↪255, 0), 2 , cv.LINE_AA)
ax[3].imshow(canny_color)
ax[3].set_title('Object Identified')
ax[3].axis('off')
plt.show()

dy = 1
y_sub_pixel = np.arange(0, im.shape[0] - 1, dy )
f_sub_pixel = np.zeros_like(y_sub_pixel)
f_sub_pixel_nn = np.zeros_like(y_sub_pixel)
#https://youtu.be/v9CFu4r6tPY

#for i, y in enumerate(y_sub_pixel):



#fig,ax = plt.subplots(figsize = (30 ,5))
#ax.plot(f_sub_pixel_nn)

#Your code hear to compute the widths. Keep in mind of the angle.
```

```
(0, 0, 0, 0)
```