

به نام خدا

اعضای گروه: امیرحسین نوری - کیمیا ایمنی - امیرمسعود شاکر

تمرین دوم

1.

آ.

```
import numpy as np
import matplotlib.pyplot as plt
import math

sonar_out = np.loadtxt('data.txt', delimiter=',')
sonar_out.shape

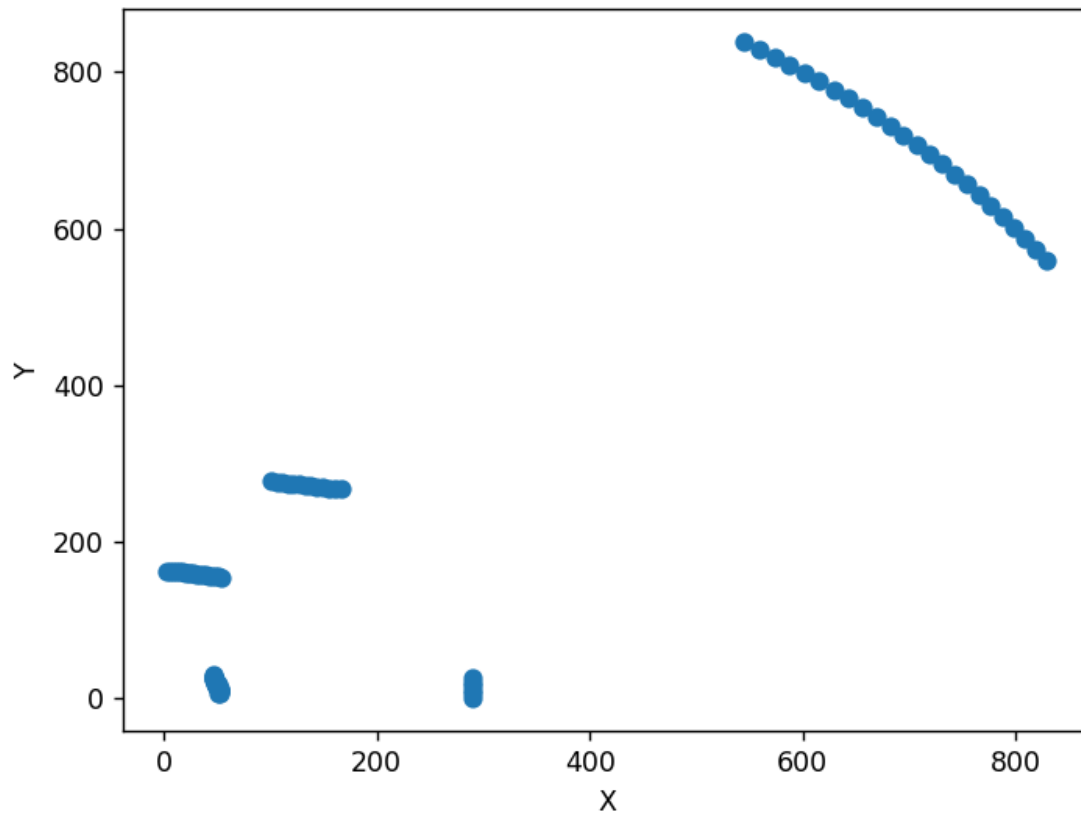
xs = []
ys = []

for i in range(90):
    theta = i * np.pi/180
    xs.append(float(sonar_out[i]) * np.cos(theta))
    ys.append(float(sonar_out[i]) * np.sin(theta))

plt.title("a. Plotting the data")
plt.xlabel('X')
plt.ylabel('Y')

plt.scatter(xs, ys)
plt.show()
```

a. Plotting the data



x=725. y=212.

```

import numpy as np
import matplotlib.pyplot as plt
import math

sonar_out = np.loadtxt('data.txt', delimiter=',')
sonar_out.shape

xs = []
ys = []

for i in range(90):
    theta = i * np.pi/180
    xs.append(float(sonar_out[i]) * np.cos(theta))
    ys.append(float(sonar_out[i]) * np.sin(theta))

def pseudo_inverse(X, y):

    W = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)

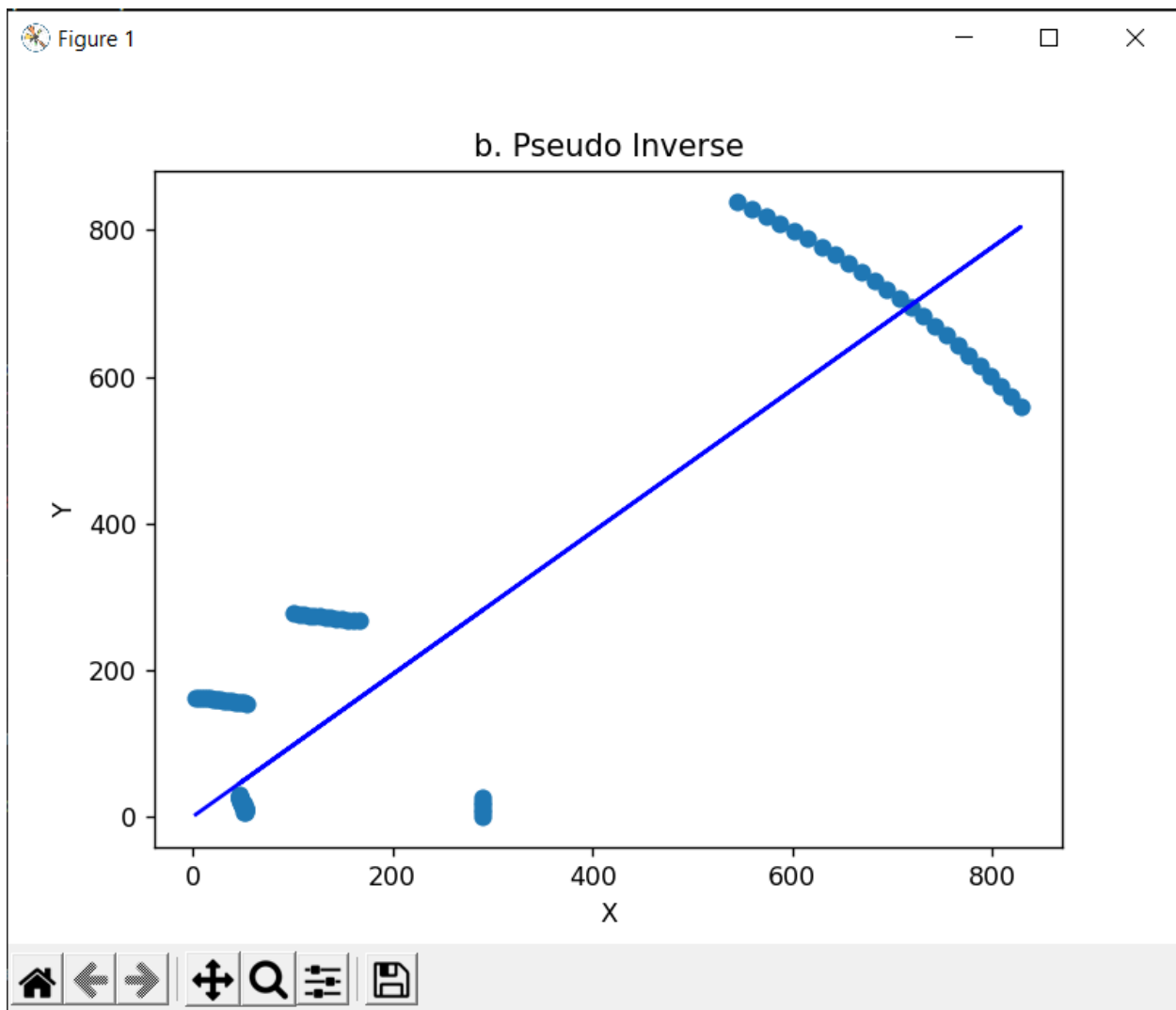
    return W

W_pseudo = pseudo_inverse(np.array(xs).reshape(-1,1),
np.array(ys).reshape(-1,1))

plt.title("b. Pseudo Inverse")
plt.xlabel('X')
plt.ylabel('Y')

plt.scatter(xs, ys)
plt.plot(xs, np.array(xs).reshape(-1,1).dot(W_pseudo), color='blue')
plt.show()

```



```

import numpy as np
import matplotlib.pyplot as plt
import math

sonar_output = np.loadtxt('data.txt', delimiter=',')
sonar_output.shape

xs = []
ys = []

for i in range(90):
    theta = i * np.pi/180
    xs.append(float(sonar_output[i]) * np.cos(theta))
    ys.append(float(sonar_output[i]) * np.sin(theta))

def pseudo_inverse(X, y):

    W = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)

    return W

W_pseudo = pseudo_inverse(np.array(xs).reshape(-1, 1),
                           np.array(ys).reshape(-1, 1))

def compute_MSE(X, y, y_hat):

    m = X.shape[0]

    MSE = (1 / (2 * m)) * np.sum((y - y_hat)**2)
    return MSE

def ransac(X, y, num_sample, threshold):

    iterations = math.inf

```

```

iterations_done = 0

max_inlier_count = 0
best_model = None

prob_outlier = 0.5
desired_prob = 0.95

total_data = np.column_stack((X, y))
data_size = len(total_data)

iterations_done = 0

while iterations > iterations_done:

    np.random.shuffle(total_data)

    sample_data = total_data[:num_sample, :]
    X = sample_data[:, :-1]
    y = sample_data[:, -1:]

    estimated_model = pseudo_inverse(X, y)

    y_hat = X.dot(estimated_model)
    MSE = compute_MSE(X, y, y_hat)
    inlier_count = np.count_nonzero(MSE < threshold)

    if inlier_count > max_inlier_count:
        max_inlier_count = inlier_count
        best_model = estimated_model

    prob_outlier = 1 - inlier_count / data_size
    iterations = math.log(1 - desired_prob) / \
        math.log(1 - (1 - prob_outlier) ** num_sample)
    iterations_done += 1

return best_model

```

```

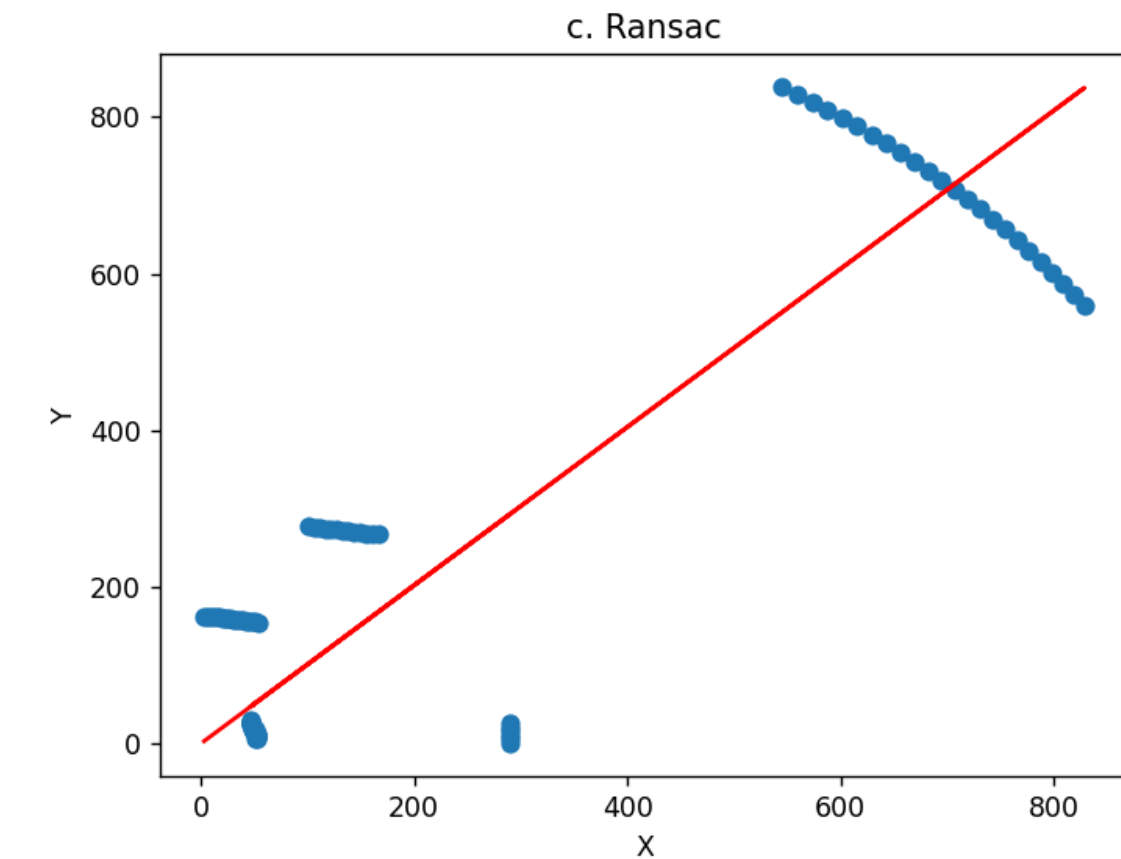
W_ransac = ransac(np.array(xs).reshape(-1, 1),
                  np.array(ys).reshape(-1, 1), num_sample=2,
                  threshold=1000000)

plt.title("c. Ransac")
plt.xlabel('X')
plt.ylabel('Y')

plt.scatter(xs, ys)
plt.plot(xs, np.array(xs).reshape(-1, 1).dot(W_ransac), color='red')
plt.show()

```

Figure 1



```

import numpy as np
import matplotlib.pyplot as plt
import math

sonar_out = np.loadtxt('data.txt', delimiter=',')
sonar_out.shape

xs = []
ys = []

for i in range(360):
    theta = i * np.pi/180
    xs.append(float(sonar_out[i]) * np.cos(theta))
    ys.append(float(sonar_out[i]) * np.sin(theta))

def pseudo_inverse(X, y):

    W = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)

    return W

def compute_MSE(X, y, y_hat):

    m = X.shape[0]

    MSE = (1 / (2 * m)) * np.sum((y - y_hat)**2)
    return MSE

def ransac(X, y, num_sample, threshold):

    iterations = math.inf
    iterations_done = 0

    max_inlier_count = 0
    best_model = None

```



```

prob_outlier = 0.5
desired_prob = 0.95

total_data = np.column_stack((X, y))
data_size = len(total_data)

iterations_done = 0

while iterations < iterations_done:

    np.random.shuffle(total_data)

    sample_data = total_data[:num_sample, :]
    X = sample_data[:, :-1]
    y = sample_data[:, -1:]

    estimated_model = pseudo_inverse(X, y)

    y_hat = X.dot(estimated_model)
    MSE = compute_MSE(X, y, y_hat)
    inlier_count = np.count_nonzero(MSE < threshold)

    if inlier_count > max_inlier_count:
        max_inlier_count = inlier_count
        best_model = estimated_model

    prob_outlier = 1 - inlier_count / data_size
    iterations = math.log(1 - desired_prob) / \
        math.log(1 - (1 - prob_outlier) ** num_sample)
    iterations_done += 1

return best_model

W_ransac = ransac(np.array(xs).reshape(-1, 1),
                  np.array(ys).reshape(-1, 1), num_sample=2,
                  threshold=1000000)

```

```

def point_line_distance(point, line):
    p1, p2 = np.array(line[0]), np.array(line[1])
    p3 = np.array(point)

    d = np.linalg.norm(np.cross(p2-p1, p1-p3))/np.linalg.norm(p2-p1)

    return d

point_line_distance((6, 7), ((6, 7), (4, 6)))

lines = []
splitted_lines = []
points = []
for i in range(len(xs)):
    points.append((xs[i], ys[i]))

lines.append((0, len(points)-1))
threshold = 10

while len(lines) != 0:
    current_line = lines.pop(0)
    max_dist, furthest_point, idx = 0, None, None
    for i in range(current_line[0], current_line[1]+1):
        distance = point_line_distance(point=points[i],
                                       line=(points[current_line[0]],
points[current_line[1]]))
        if max_dist < distance:
            max_dist = distance
            furthest_point = points[i]
            idx = i

    if max_dist > threshold:
        lines.append((current_line[0], idx))
        lines.append((idx, current_line[1]))
    else:
        splitted_lines.append(current_line)

```

```

splitted_lines

plt.scatter(xs, ys)
for line in splitted_lines:
    x_values = [points[line[0]][0], points[line[1]][0]]
    y_values = [points[line[0]][1], points[line[1]][1]]
    plt.plot(x_values, y_values, color='blue')

plt.title("Split")
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

line_without_outliers = []
OUTLIERTHRESHOLD = 200
for line in splitted_lines:

    xL1, yL1, xL2, yL2 = points[line[0]][0], points[line[0]]
                                ][1],
    points[line[1]][0], points[line[1]][1]
    Llength = math.sqrt((yL2-yL1)**2 + (xL2-xL1)**2)
    if not(abs(line[0]-line[1]) <= 3 or Llength > OUTLIERTHRESHOLD):
        line_without_outliers.append(line)

def find_sum_distance_point(current_line):
    sum_distance = 0
    for i in range(current_line[0], current_line[1]+1):
        dist = point_line_distance(point=points[i], line=(
            points[current_line[0]], points[current_line[1]]))
        sum_distance = sum_distance + dist
    return sum_distance

find_sum_distance_point(splitted_lines[0])

splittedlines = line_without_outliers.copy()
index = 0

```

```

thresholdMerge = 0.2
while index != len(splittedlines)-1:
    lineL = splittedlines[index]
    lineR = splittedlines[index+1]
    xL1, yL1, xL2, yL2 = points[lineL[0]][0], points[lineL[0]
                                                ][1],
points[lineL[1]][0], points[lineL[1]][1]
    xR1, yR1, xR2, yR2 = points[lineR[0]][0], points[lineR[0]
                                                ][1],
points[lineR[1]][0], points[lineR[1]][1]
    lineLeftError = find_sum_distance_point(lineL)
    lineRightError = find_sum_distance_point(lineR)
    mergedLine = (lineL[0], lineR[1])
    mergedLineError = find_sum_distance_point(mergedLine)
    Llength = math.sqrt((yL2-yL1)**2 + (xL2-xL1)**2)
    Rlength = math.sqrt((yR2-yR1)**2 + (xR2-xR1)**2)
    Mlength = math.sqrt((yR2-yL1)**2 + (xR2-xL1)**2)
    LeftError = lineLeftError / Llength
    RightError = lineRightError / Rlength
    MergedError = mergedLineError / Mlength

    if LeftError + RightError > MergedError + thresholdMerge:
        splittedlines[splittedlines.index(lineL)] = (lineL[0],
lineR[1])
        splittedlines.remove(lineR)
        index = index - 2

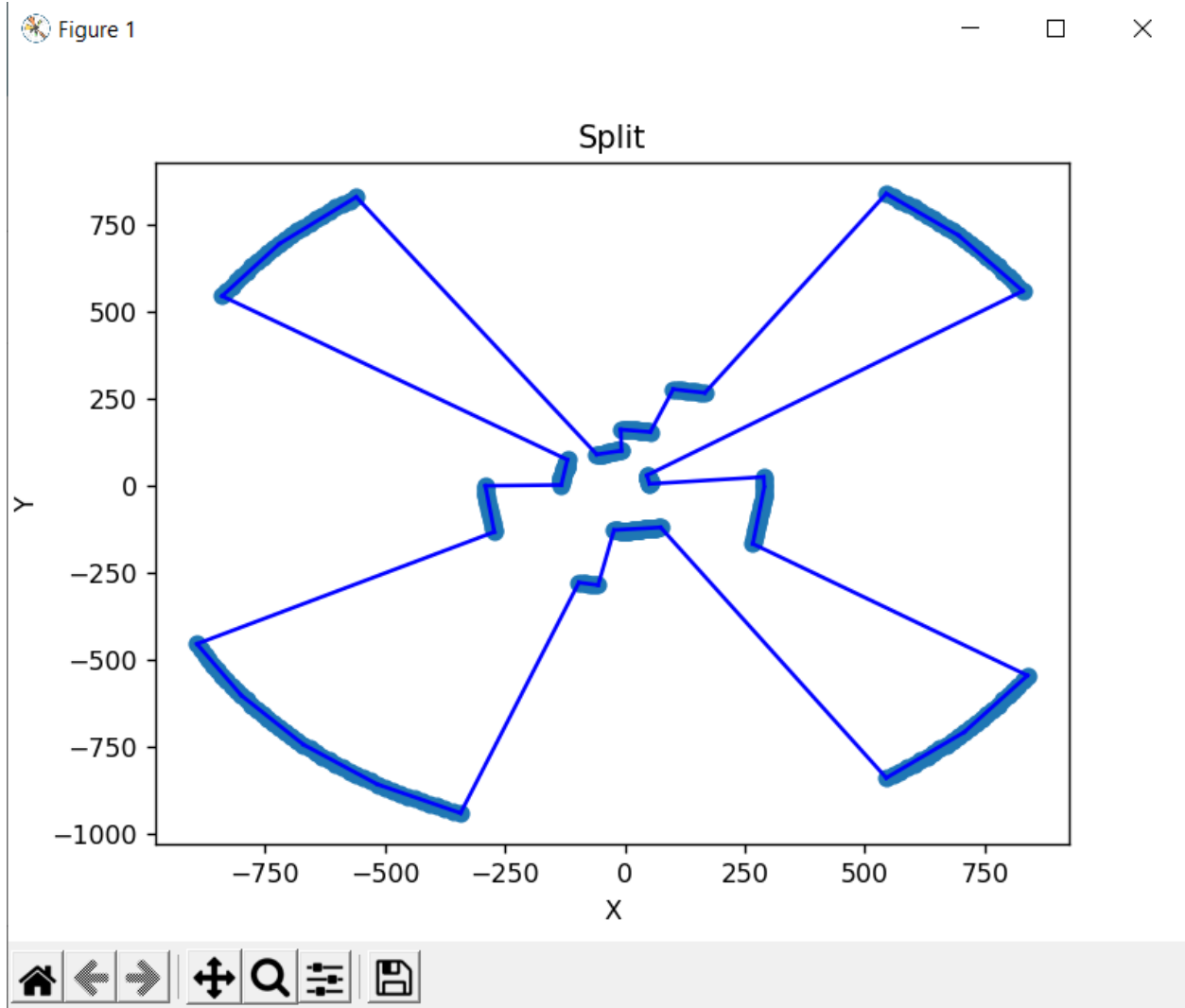
    index = index+1
    if(index < 0):
        index = 0

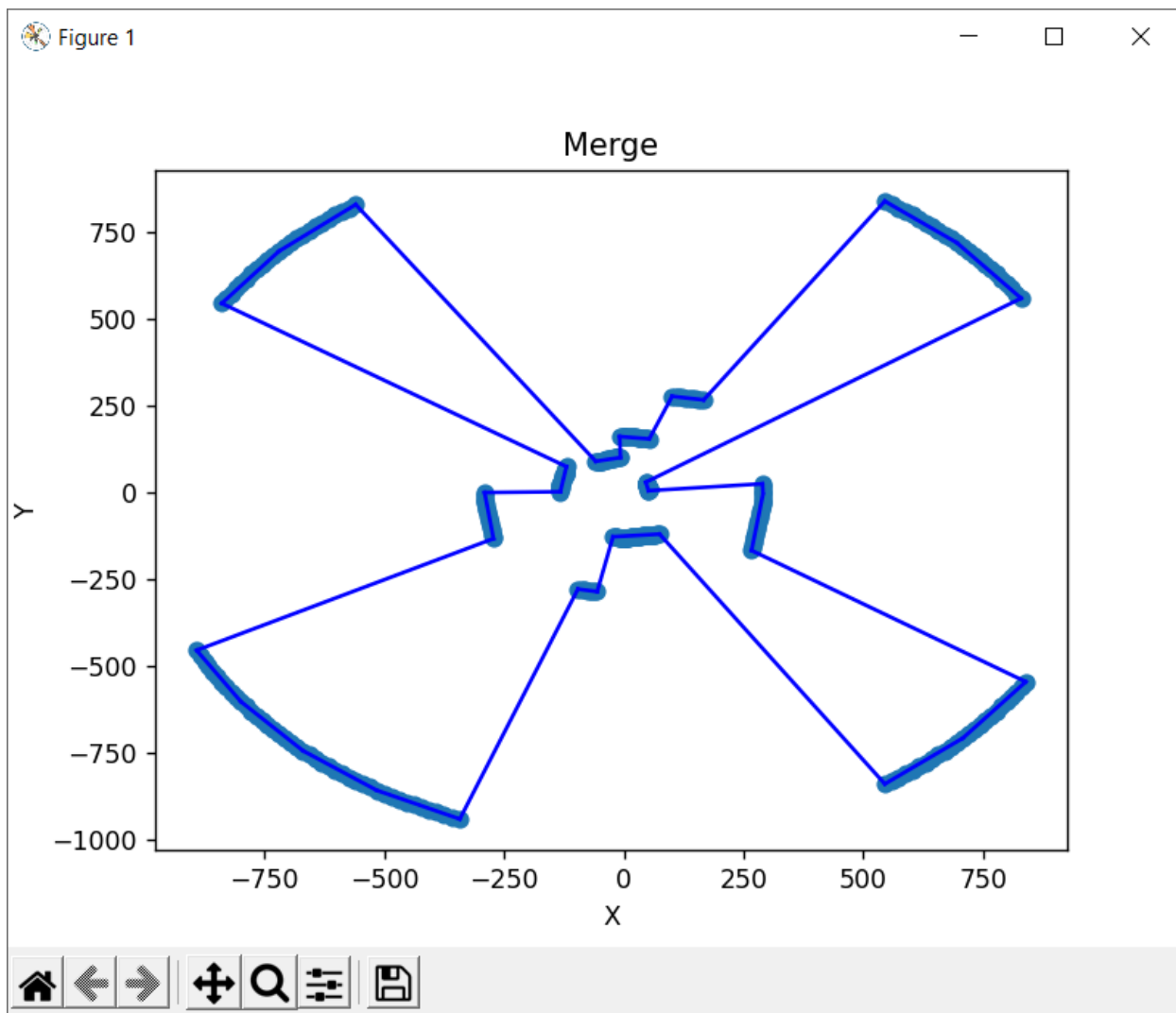
plt.scatter(xs, ys)
for line in splitted_lines:
    x_values = [points[line[0]][0], points[line[1]][0]]
    y_values = [points[line[0]][1], points[line[1]][1]]
    plt.plot(x_values, y_values, color='blue')

plt.title("Merge")
plt.xlabel('X')

```

```
plt.ylabel('Y')  
plt.show()
```





2.

بخش اول:

لینک کولب کد CNN:

<https://colab.research.google.com/drive/1QDxWclP79USON8mgFIRIEE1csa7g15k?usp=sharing>

توضیح کد:

پس از unzip کردن داده ها و import های لازم، ابتدا آدرس داده های آموزشی و تست را تعیین میکنیم و img_size را برابر 28 قرار میدهیم (تصاویر 28*28 هستند).

سپس با استفاده از ImageDataGenerator، یک datagen میسازیم و تنظیمات مربوط به آن (مانند چرخش افقی و عمودی داده ها و نسبت تقسیم داده ها ی train , validation) را انجام میدهیم.

با استفاده از datagen و متد flow_from_directory، داده های آموزشی و تست را میسازیم.

سپس مدل خود را با استفاده از ماژول keras.models Sequential آن میسازیم.

دو لایه Conv2d و دو لایه MaxPooling میسازیم. سپس لایه Flatten و پس از آن دو لایه Dense.

برای لایه های Conv2d و لایه Dense اول از تابع فعالسازی 'relu' استفاده کرده و برای لایه Dense آخر از 'softmax' استفاده میکنیم.

پس از ساخت مدل، آن را با تابع خطای 'categorical_crossentropy' و بهینه ساز 'Adam' کامپایل میکنیم.

سپس مدل را با داده های آموزشی و تست، fit میکنیم و با تعداد epoch = 20، مدل را آموزش میدهیم:

```
D: Epoch 1/20
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
after removing the cwd from sys.path.
1179/1179 [=====] - 20s 16ms/step - loss: 0.1049 - accuracy: 0.9640 - val_loss: 0.0218 - val_accuracy: 0.9952
Epoch 2/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0025 - accuracy: 0.9994 - val_loss: 6.1326e-04 - val_accuracy: 1.0000
Epoch 3/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0074 - accuracy: 0.9978 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 4/20
1179/1179 [=====] - 19s 16ms/step - loss: 4.1634e-04 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 0.9990
Epoch 5/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0037 - accuracy: 0.9990 - val_loss: 2.1864e-04 - val_accuracy: 1.0000
Epoch 6/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0053 - accuracy: 0.9985 - val_loss: 1.2291e-04 - val_accuracy: 1.0000
Epoch 7/20
1179/1179 [=====] - 19s 16ms/step - loss: 7.9803e-04 - accuracy: 0.9997 - val_loss: 2.4533e-04 - val_accuracy: 1.0000
Epoch 8/20
1179/1179 [=====] - 19s 16ms/step - loss: 1.3778e-04 - accuracy: 1.0000 - val_loss: 8.1204e-06 - val_accuracy: 1.0000
Epoch 9/20
1179/1179 [=====] - 19s 16ms/step - loss: 1.2014e-05 - accuracy: 1.0000 - val_loss: 5.5142e-06 - val_accuracy: 1.0000
Epoch 10/20
1179/1179 [=====] - 19s 16ms/step - loss: 3.9035e-05 - accuracy: 1.0000 - val_loss: 9.4563e-05 - val_accuracy: 1.0000
Epoch 11/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0037 - accuracy: 0.9988 - val_loss: 0.0213 - val_accuracy: 0.9981
Epoch 12/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0020 - accuracy: 0.9996 - val_loss: 5.0525e-05 - val_accuracy: 1.0000
Epoch 13/20
1179/1179 [=====] - 19s 16ms/step - loss: 5.7065e-06 - accuracy: 1.0000 - val_loss: 2.2824e-04 - val_accuracy: 1.0000
Epoch 14/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0018 - accuracy: 0.9992 - val_loss: 2.4542e-04 - val_accuracy: 1.0000
Epoch 15/20
1179/1179 [=====] - 19s 16ms/step - loss: 1.0269e-04 - accuracy: 1.0000 - val_loss: 4.9230e-06 - val_accuracy: 1.0000
Epoch 16/20
1179/1179 [=====] - 19s 16ms/step - loss: 3.4116e-06 - accuracy: 1.0000 - val_loss: 9.4569e-04 - val_accuracy: 0.9990
Epoch 17/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 1.5161e-04 - val_accuracy: 1.0000
Epoch 18/20
1179/1179 [=====] - 19s 16ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 5.8271e-06 - val_accuracy: 1.0000
Epoch 19/20
1179/1179 [=====] - 19s 16ms/step - loss: 9.3742e-06 - accuracy: 1.0000 - val_loss: 9.1635e-07 - val_accuracy: 1.0000
Epoch 20/20
1179/1179 [=====] - 19s 16ms/step - loss: 6.6876e-04 - accuracy: 0.9997 - val_loss: 0.0056 - val_accuracy: 0.9981
```

سپس یک تابع برای تشخیص شکل تصویر ورودی مینویسیم که پس از انجام پیش پردازش های لازم روی تصویر ورودی، متد predict از مدل را روی تصویر ورودی صدا میزنند.

بعد با استفاده از چند شرط if روی پیش بینی مدل، یکی از مقادیر 'circle'، 'square'، 'star' و 'triangle' را خروجی میدهیم.

در نهایت خروجی تابع نوشته شده روی همه داده های تست را در یک فایل csv طبق فرمت خواسته شده مینویسیم.

بخش دوم:

کد کنترلر:

```
from controller import Robot
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from keras.models import load_model

def make_prediction(img_dir):
    model =
load_model('E:/University/Term_8/Robotix/Assignments/HW3/ex3/Q2/worlds
/CNN_q2.h5')

    test_img = image.load_img(img_dir, target_size=(img_size,
img_size))
    test_img = test_img.convert(mode='RGB')
    test_img = tf.keras.preprocessing.image.img_to_array(test_img)
    test_img = np.expand_dims(test_img, axis=0)
    test_img =
tf.keras.applications.inception_v3.preprocess_input(test_img)

    result = model.predict(test_img)

    if result[0][0] > result[0][1] + result[0][2] + result[0][3]:
        return 'circle'
    elif result[0][1] > result[0][0] + result[0][2] + result[0][3]:
        return 'square'
    elif result[0][2] > result[0][0] + result[0][1] + result[0][3]:
        return 'star'
```



```

        elif result[0][3] > result[0][0] + result[0][1] + result[0][2]:
            return 'triangle'

img_size = 28
timestep = 64

robot = Robot()

left_motor = robot.getDevice('left wheel motor')
right_motor = robot.getDevice('right wheel motor')

left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))

left_motor.setVelocity(0)
right_motor.setVelocity(0)

camera = robot.getDevice('camera_1')
camera.enable(timestep)

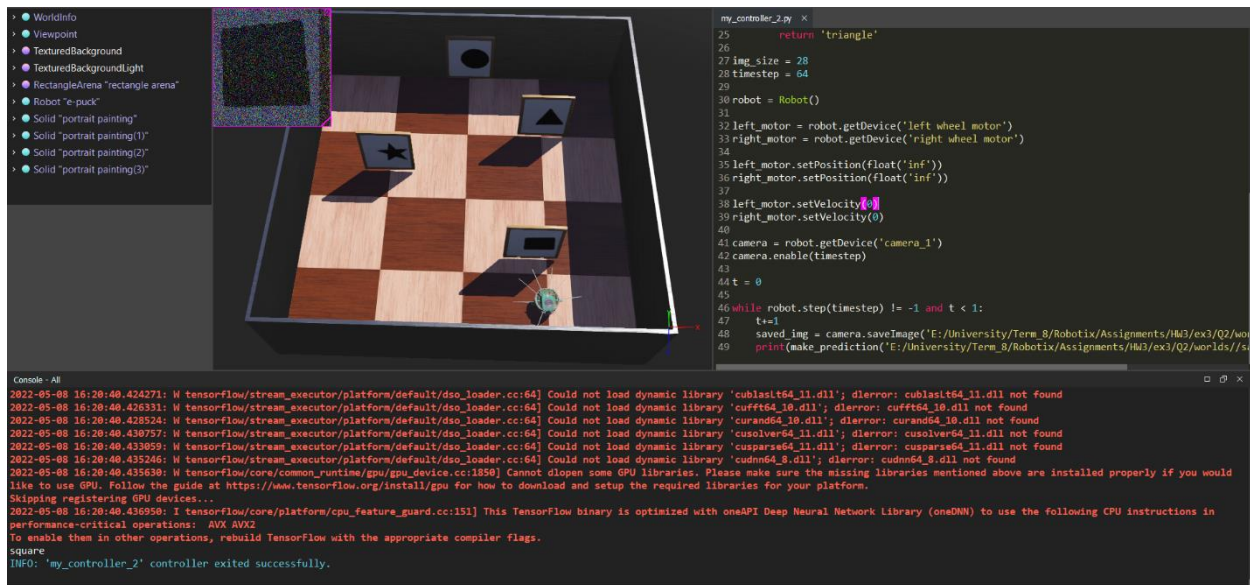
t = 0

while robot.step(timestep) != -1 and t < 1:
    t+=1
    saved_img =
camera.saveImage('E:/University/Term_8/Robotix/Assignments/HW3/ex3/Q2/
worlds//saved_img.png', 100)
    print(make_prediction('E:/University/Term_8/Robotix/Assignments/HW
3/ex3/Q2/worlds//saved_img.png'))

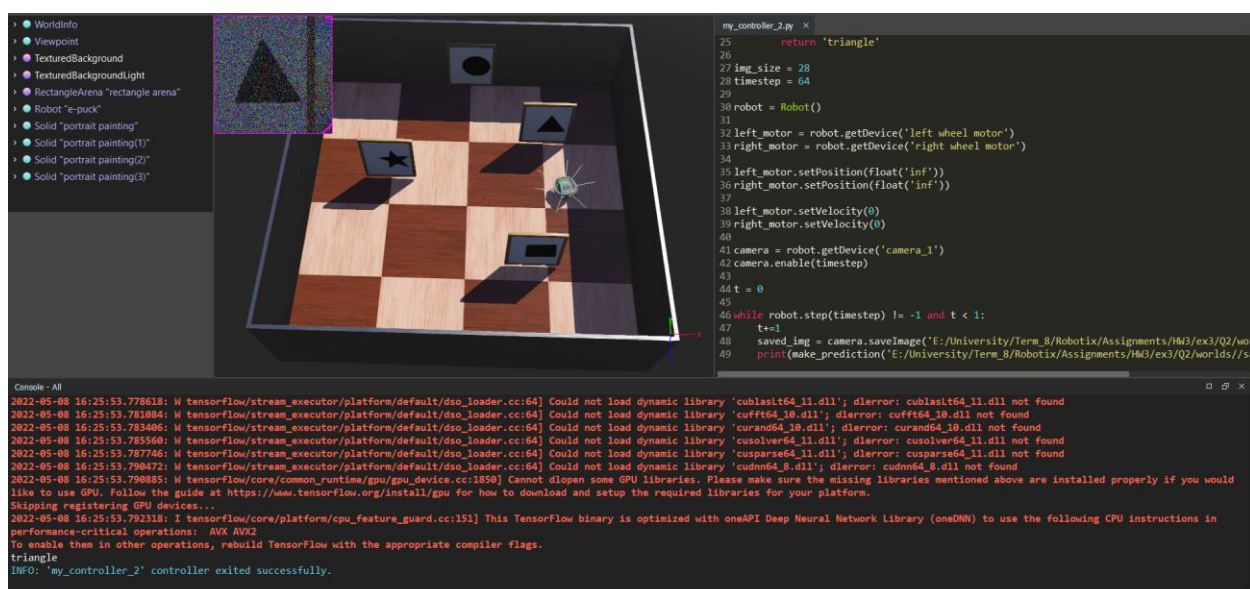
```

چند نمونه از خروجی ها:

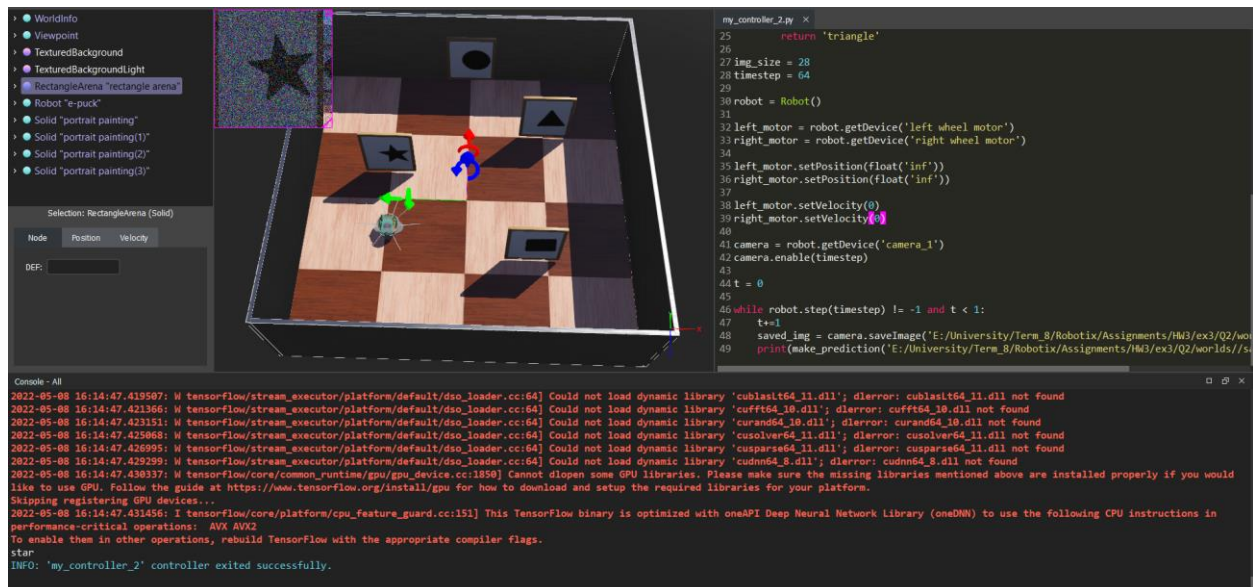
تشخیص مربع:



تشخیص مثلث:



تشخيص ستاره:



The screenshot displays a ROS2 simulation environment. On the left, a sidebar shows a tree view with nodes like 'WorldInfo', 'Viewpoint', 'TexturedBackground', and 'RectangleArena'. The top center features a 3D view of a robot with a camera, situated in a checkered arena. The right sidebar contains a code editor showing Python code for a robot controller. The bottom console window displays system logs, including warnings about missing dynamic libraries and a successful exit message for the 'my_controller_2' node.

```
my_controller_2.py
25     return 'triangle'
26
27 img_size = 28
28 timestep = 64
29
30 robot = Robot()
31
32 left_motor = robot.getDevice('left wheel motor')
33 right_motor = robot.getDevice('right wheel motor')
34
35 left_motor.setPosition(float('inf'))
36 right_motor.setPosition(float('inf'))
37
38 left_motor.setVelocity(0)
39 right_motor.setVelocity(0)
40
41 camera = robot.getDevice('camera_1')
42 camera.enable(timestep)
43
44 t = 0
45
46 while robot.step(timestep) != -1 and t < 1:
47     t+=1
48     saved_img = camera.saveImage('E:/University/Term_8/Robotix/Assignments/PM3/ex3/Q2/worlds/s
49     print(make_prediction('E:/University/Term_8/Robotix/Assignments/PM3/ex3/Q2/worlds/s
```

Console - All

```
2022-05-08 16:14:47.419507: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublaslt64_11.dll'; dlerror: cublaslt64_11.dll not found
2022-05-08 16:14:47.421366: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cufft64_10.dll'; dlerror: cufft64_10.dll not found
2022-05-08 16:14:47.423151: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'curand64_10.dll'; dlerror: curand64_10.dll not found
2022-05-08 16:14:47.423968: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusolver64_11.dll'; dlerror: cusolver64_11.dll not found
2022-05-08 16:14:47.426995: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cuspars64_11.dll'; dlerror: cuspars64_11.dll not found
2022-05-08 16:14:47.429299: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found
2022-05-08 16:14:47.430337: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1850] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2022-05-08 16:14:47.431456: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
star
INFO: 'my_controller_2' controller exited successfully.
```