# کد برای حرکت دایره ای

import numpy as np

import time

import matplotlib.pyplot as plt

from FuzzyLibrary import JointFuzzyController

from ur5 import UR5

# تعریف ربات و کنترلر فازی

ur5 = UR5()

fuzzy\_controller = JointFuzzyController('Joint1')

# تنظیمات مسیر دایره‌ای و مانع

radius = 0.7 # شعاع دایره

obstacle\_radius = 0.2 # شعاع مانع

center\_x, center\_y, center\_z = 0, 0, 0.5 # مرکز دایره

obstacle\_position = np.array([0.6, 0, 0.5]) # موقعیت مانع

num\_steps = 200

# حد پایین و بالای چرخش مفصل 1 (مثال)

joint1\_min = -np.pi / 2

joint1\_max = np.pi / 2

# تولید نقاط مسیر دایره‌ای بر اساس حداقل و حداکثر چرخش مفصل 1

theta\_values = np.linspace(joint1\_min, joint1\_max, num\_steps)

path\_points = np.array([

    [center\_x + radius \* np.cos(theta),

     center\_y + radius \* np.sin(theta),

     center\_z] for theta in theta\_values

])

# لیست ذخیره مسیر طی شده و مختصات گریپر

gripper\_trajectory = []

gripper\_positions = [] # لیست جدید برای ذخیره موقعیت دقیق گریپر

def check\_collision(position, obstacle\_position, obstacle\_radius):

    """ بررسی برخورد ربات با مانع """

    distance = np.linalg.norm(np.array(position) - np.array(obstacle\_position))

    return distance <= obstacle\_radius

# تابع تصحیح مسیر برای دور زدن مانع

def adjust\_path\_to\_avoid\_obstacle(point, obstacle\_position):

    adjusted\_point = np.array(point) + np.array([0, 0, 0.2])

    return adjusted\_point

# شروع حرکت ربات از حد پایین چرخش مفصل 1

ur5.move\_to\_config([joint1\_min, 0, 0, 0, 0, 0])

print("✅ Robot Started from Lower Joint Limit!")

# شروع حرکت ربات در مسیر دایره‌ای

for i, point in enumerate(path\_points):

    # بررسی برخورد با مانع برای تمامی مفاصل

    collision\_detected = check\_collision(point, obstacle\_position, obstacle\_radius)

    # محاسبه سرعت کنترلر فازی

    velocity = fuzzy\_controller.compute\_velocity(0.1, 0.05)

    # ذخیره مسیر هدف و موقعیت واقعی گریپر

    gripper\_trajectory.append(point.copy())

    # دریافت موقعیت فعلی گریپر و ذخیره آن

    current\_gripper\_position = ur5.get\_ground\_truth()[:3, 3]

    gripper\_positions.append(current\_gripper\_position.copy())

    # اگر مانع در مسیر باشد

    if collision\_detected:

        print(f"⚠️ Obstacle Detected at Distance!")

        point = adjust\_path\_to\_avoid\_obstacle(point, obstacle\_position)

        ur5.move\_to\_pose(point, [0, 0, np.pi/2]) # تغییر مسیر به سمت بالا

        print(f"Bypassing Obstacle at Point: {point} | Velocity Adjusted!")

    else:

        # حرکت معمولی روی مسیر دایره‌ای

        ur5.move\_to\_pose(point, [0, 0, np.pi/2]) # چنگک به سمت بیرون

        print(f"Moving on Circle: {point} | Velocity: {velocity:.2f}")

    # بررسی جهت چرخش برای پیشگیری از برخورد در تغییر جهت کامل

    if i > 0:

        prev\_point = path\_points[i - 1]

        angle\_change = np.linalg.norm(np.array(point) - np.array(prev\_point))

        if angle\_change > np.pi / 2: # اگر تغییر زاویه بیش از حد باشد

            print(f"⚠️ Large Angle Change Detected, Rechecking Collision...")

            if check\_collision(point, obstacle\_position, obstacle\_radius):

                point = adjust\_path\_to\_avoid\_obstacle(point, obstacle\_position)

                ur5.move\_to\_pose(point, [0, 0, np.pi/2])

    time.sleep(0.1)

# بازگشت ربات به حد بالای چرخش مفصل 1

ur5.move\_to\_config([joint1\_max, 0, 0, 0, 0, 0])

print("✅ Reached Upper Joint Limit Successfully!")

# 📊 رسم مسیر طی شده توسط گریپر و موقعیت واقعی گریپر

gripper\_trajectory = np.array(gripper\_trajectory)

gripper\_positions = np.array(gripper\_positions) # تبدیل به آرایه برای رسم

fig = plt.figure(figsize=(12, 10))

ax = fig.add\_subplot(111, projection='3d')

# رسم مسیر هدف و موقعیت واقعی

ax.plot(gripper\_trajectory[:, 0], gripper\_trajectory[:, 1], gripper\_trajectory[:, 2], label='Target Gripper Path', linewidth=2, linestyle='--')

ax.plot(gripper\_positions[:, 0], gripper\_positions[:, 1], gripper\_positions[:, 2], label='Actual Gripper Path', linewidth=2, color='purple')

ax.scatter(obstacle\_position[0], obstacle\_position[1], obstacle\_position[2], color='red', s=150, label='Obstacle')

ax.scatter(gripper\_trajectory[0, 0], gripper\_trajectory[0, 1], gripper\_trajectory[0, 2], color='green', s=150, label='Start Point')

ax.scatter(gripper\_trajectory[-1, 0], gripper\_trajectory[-1, 1], gripper\_trajectory[-1, 2], color='blue', s=150, label='End Point')

# تنظیمات نمودار

ax.set\_xlabel('X (meters)')

ax.set\_ylabel('Y (meters)')

ax.set\_zlabel('Z (meters)')

ax.set\_title('3D Gripper Trajectory and Actual Position with Obstacle Avoidance')

ax.legend()

plt.show()

# کد برای حرکت به سمت نقاط دلخواه

import numpy as np

import time

import matplotlib.pyplot as plt

from FuzzyLibrary import JointFuzzyController

from ur5 import UR5

# تعریف ربات و کنترلر فازی

ur5 = UR5()

fuzzy\_controller = JointFuzzyController('Joint1')

# تنظیمات مانع

obstacle\_radius = 0.2 # شعاع مانع

obstacle\_position = np.array([0.1, 0.1, 0.5]) # موقعیت جدید مانع

# تعریف موقعیت‌های هدف

target\_positions = [

    np.array([-0.1, -0.2, 0.7]),

    np.array([0.5, 0, 0.5]),

    np.array([-0.3, 0.2, 0.65]) # موقعیت جدید سوم

]

# لیست ذخیره مسیر طی شده

gripper\_trajectory = []

def check\_collision(position, obstacle\_position, obstacle\_radius):

    """ بررسی برخورد ربات با مانع """

    distance = np.linalg.norm(np.array(position) - np.array(obstacle\_position))

    return distance <= obstacle\_radius

# تابع تصحیح مسیر برای دور زدن مانع

def adjust\_path\_to\_avoid\_obstacle(point, obstacle\_position):

    vector\_to\_obstacle = np.array(obstacle\_position) - np.array(point)

    adjusted\_point = np.array(point) + np.array([0, 0, 0.2]) # حرکت عمودی به سمت بالا

    return adjusted\_point

# حرکت ربات به هر موقعیت هدف

for target\_position in target\_positions:

    print(f"🚀 Moving towards: {target\_position}")

    # بررسی برخورد و محاسبه سرعت

    collision\_detected = check\_collision(target\_position, obstacle\_position, obstacle\_radius)

    velocity = fuzzy\_controller.compute\_velocity(0.1, 0.05)

    # ذخیره مسیر برای رسم

    gripper\_trajectory.append(target\_position.copy())

    # اگر مانع شناسایی شد، مسیر را اصلاح کن

    if collision\_detected:

        print(f"⚠️ Obstacle Detected at {target\_position}!")

        target\_position = adjust\_path\_to\_avoid\_obstacle(target\_position, obstacle\_position)

        ur5.move\_to\_pose(target\_position, [0, 0, np.pi/2]) # تغییر مسیر به بالا

        print(f"Bypassing Obstacle at: {target\_position} | Velocity Adjusted!")

    else:

        # حرکت معمولی به نقطه هدف

        ur5.move\_to\_pose(target\_position, [0, 0, np.pi/2])

        print(f"✅ Moved to: {target\_position} | Velocity: {velocity:.2f}")

    time.sleep(0.1)

# بازگشت ربات به موقعیت اولیه

ur5.move\_to\_config([0, 0, 0, 0, 0, 0])

print("✅ Trajectory Completed Successfully!")

# 📊 رسم مسیر طی شده توسط گریپر

gripper\_trajectory = np.array(gripper\_trajectory)

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

# رسم مسیر

ax.plot(gripper\_trajectory[:, 0], gripper\_trajectory[:, 1], gripper\_trajectory[:, 2], label='Gripper Trajectory', linewidth=2)

ax.scatter(obstacle\_position[0], obstacle\_position[1], obstacle\_position[2], color='red', s=150, label='Obstacle')

ax.scatter(gripper\_trajectory[0, 0], gripper\_trajectory[0, 1], gripper\_trajectory[0, 2], color='green', s=150, label='Start Point')

ax.scatter(gripper\_trajectory[-1, 0], gripper\_trajectory[-1, 1], gripper\_trajectory[-1, 2], color='blue', s=150, label='End Point')

# تنظیمات نمودار

ax.set\_xlabel('X (meters)')

ax.set\_ylabel('Y (meters)')

ax.set\_zlabel('Z (meters)')

ax.set\_title('3D Gripper Trajectory with Obstacle Avoidance')

ax.legend()

plt.show()