ECE780_Assignment_4_Q4cib_python_code

July 23, 2024

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

x0 = np.array([[8],[8],[np.pi/2],[0]]) # state vector (x, y, theta, delta)
u = np.array([[0],[0]]) # control input (v, omega)
sd = np.array([[0],[0]]) # target position
L = 0.1 # point distance
l = 2 # vehicle length
alpha = 0.1 # gain

# Simulation parameter
dt = 0.1
T = 1000
arrow_length = 0.1
num_steps = int(T / dt)

# Plot the trajectory
plt.scatter(x0[0][0], x0[1][0], color='green', label='Start')
plt.scatter(sd[0][0], sd[1][0], color='blue', label='Target')

def dynamic(x0, u, dt):
    v = u[0][0]
    omega = u[1][0]
    x, y, theta, delta = x0[0][0], x0[1][0], x0[2][0], x0[3][0]
    x = x + v*np.cos(delta)*np.cos(theta) * dt
    y = y + v*np.cos(delta)*np.sin(theta) * dt
    theta = theta + v/l*np.sin(delta) * dt
    delta = delta + omega * dt
    x0 = np.array([[x],
                   [y],
                   [theta],
                   [delta]])
    return x0

def S(x,y,theta,delta):
    xp = x + l*np.cos(theta) + L*np.cos(delta+theta)
    yp = y + l*np.sin(theta) + L*np.sin(delta+theta)
```

```
    s = np.array([[xp],
                  [yp]])
    return s-sd[:2]

def g(theta,delta):
    return np.array([[np.cos(delta+theta)-L/l*np.sin(delta)*np.sin(delta+theta), -L*np.sin(delta+theta)],
                     [np.sin(delta+theta)+L/l*np.sin(delta)*np.cos(delta+theta), L*np.cos(delta+theta)]])

# Initialize lists to store the trajectory
xp_trajectory = []
yp_trajectory = []

# Add yaw angle arrow representation for the start position
x, y, theta, delta = x0[0][0], x0[1][0], x0[2][0], x0[3][0]
s = S(x,y,theta,delta)
plt.quiver(s[0][0], s[1][0], np.cos(x0[2])*arrow_length, np.sin(x0[2])*arrow_length, color='black', angles='xy', scale_units='xy', scale=0.1, width=0.01)

for _ in range (num_steps):
    x, y, theta, delta = x0[0][0], x0[1][0], x0[2][0], x0[3][0]
    u_star = -alpha/2 * np.linalg.pinv(g(theta, delta)) @ S(x,y,theta,delta)
    x0 = dynamic(x0, u_star, dt)
    x, y, theta, delta = x0[0][0], x0[1][0], x0[2][0], x0[3][0]
    s = S(x,y,theta,delta)
    # Store the new coordinates
    xp_trajectory.append(s[0][0])
    yp_trajectory.append(s[1][0])

# Add yaw angle arrow representation for the last position
plt.quiver(s[0][0], s[1][0], np.cos(x0[2])*arrow_length, np.sin(x0[2])*arrow_length, color='black', angles='xy', scale_units='xy', scale=0.1, width=0.01, label="Heading")

plt.plot(xp_trajectory, yp_trajectory, marker='o', color="red", label="Trajectory of Xp and Yp")
plt.xlabel('X Position')
plt.ylabel('Y Position')
plt.title('Point Trajectory')
plt.grid(True)
plt.legend()
plt.savefig('script\\assignment\\assignment_4_picture\\point_trajectory.png')
plt.show()
```