ECE780_Assignment_4_Q4cii_python_code

July 23, 2024

[ ]:
```
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 = 200
arrow_length = 0.1
num_steps = int(T / dt)

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]
```

```python
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
x_trajectory = []
y_trajectory = []
xp_trajectory = []
yp_trajectory = []
theta_trajectory = []
delta_trajectory = []

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
    x_trajectory.append(x)
    y_trajectory.append(y)
    xp_trajectory.append(s[0][0])
    yp_trajectory.append(s[1][0])
    theta_trajectory.append(theta)
    delta_trajectory.append(delta)

time = np.linspace(0, T*dt, num_steps)

plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)
plt.plot(time, x_trajectory, label='x')
plt.xlabel('Time ()')
plt.ylabel('X Position')
plt.title('X Position over Time')
plt.legend()
plt.grid(True)

plt.subplot(2, 3, 2)
plt.plot(time, y_trajectory, label='y')
plt.xlabel('Time')
plt.ylabel('Y Position')
plt.title('Y Position over Time')
plt.legend()
plt.grid(True)
```

```
plt.subplot(2, 3, 3)
plt.plot(time, xp_trajectory, label='xp')
plt.xlabel('Time')
plt.ylabel('XP Position')
plt.title('XP Position over Time')
plt.legend()
plt.grid(True)

plt.subplot(2, 3, 4)
plt.plot(time, yp_trajectory, label='yp')
plt.xlabel('Time')
plt.ylabel('YP Position')
plt.title('YP Position over Time')
plt.legend()
plt.grid(True)

plt.subplot(2, 3, 5)
plt.plot(time, theta_trajectory, label='theta')
plt.xlabel('Time')
plt.ylabel('Theta')
plt.title('Theta over Time')
plt.legend()
plt.grid(True)

plt.subplot(2, 3, 6)
plt.plot(time, delta_trajectory, label='delta')
plt.xlabel('Time')
plt.ylabel('Delta')
plt.title('Delta over Time')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.savefig('script\\assignment\\assignment_4_picture\\vehicle_state_trajectory.
  ↪png')
plt.show()
```