University of California San Diego

Course #: MAE204 Robotics

Final Project: Milestone 2

March 12, 2024

Author	*Email*
Xuanbin Peng	ax009017@acsmail.ucsd.edu
Yixin Zhang	ax009074@acsmail.ucsd.edu

Contents

1 MileStone 2

1.1 Code

The Python file *next_state.py* is as follows:

```
"""
For this component, you will write a function called NextState that uses the kinematics of the youBot
(see MR Exercise 13.33), your knowledge of velocity kinematics, and your knowledge of the Euler method
 to predict how the robot will move in a small timestep given its current configuration and velocity.
 Thus, your function NextState should take as inputs:
 Inputs
• The current state of the robot (13 variables: 3 for chassis, 5 for arm, 4 for wheel angles, one for gripper
↪  state)
• The joint and wheel velocities (10 variables: 5 for arm   , 4 for wheels u, 1 for gripper state   )
• The timestep size t (1 parameter)
• The maximum joint and wheel velocity magnitude (1 parameter)
Outputs NextState should also produce the following outputs that describe the configuration of the robot
one timestep (t) later:
• The next state (configuration) of the robot (13 variables)

Approach The function NextState is based on a simple first-order Euler step: • new arm joint angles = (old arm
↪  joint angles) + (joint speeds)t
• new wheel angles = (old wheel angles) + (wheel speeds)t
• new chassis configuration is obtained from odometry, as described in Chapter 13.4
"""
import csv

import numpy as np
import modern_robotics as mr

def next_state(
    current_state, joint_and_wheel_velocities, delta_t, max_joint_and_wheel_velocity
):
    # exatract the current state
    odemetry_curr = current_state[0:3]
    arm_joint_angles_curr = current_state[3:8]
    wheel_angles_curr = current_state[8:12]
    # extract the joint and wheel velocities
    arm_joint_velocities = joint_and_wheel_velocities[0:5]
    wheel_velocities = joint_and_wheel_velocities[5:9]

    gripper_state = joint_and_wheel_velocities[9]
    #need to check the maximum joint and wheel velocity
    #if any of the joint or wheel velocity is greater than the maximum joint and wheel velocity, then clip
    ↪  this entry to be maximum joint and wheel velocity
    arm_joint_velocities = np.clip(arm_joint_velocities, -max_joint_and_wheel_velocity,
    ↪  max_joint_and_wheel_velocity)
    wheel_velocities = np.clip(wheel_velocities, -max_joint_and_wheel_velocity, max_joint_and_wheel_velocity)

    # specify the configuration of the chassis
    l = 0.47 / 2
    w = 0.3 / 2
    r = 0.0475

    # define the pseudoinverse of H for four-wheel mecanum drive
```

```
    F = (
        np.array(
            [
                [-1 / (l + w), 1 / (l + w), 1 / (l + w), -1 / (l + w)],
                [1, 1, 1, 1],
                [-1, 1, -1, 1],
            ]
        )
        * r
        / 4
    )
    V_chassis = np.dot(F, wheel_velocities) * delta_t
    # calculate the new odemetry
    wbz = V_chassis[0]
    vbx = V_chassis[1]
    vby = V_chassis[2]

    if wbz == 0:
        odemetry_new = odemetry_curr + np.array([0, vbx, vby])
    else:
        odemetry_new = odemetry_curr + np.array(
            [
                wbz,
                (vbx * np.sin(wbz) + vby * (np.cos(wbz) - 1)) / wbz,
                (vby * np.sin(wbz) + vbx * (1 - np.cos(wbz))) / wbz,
            ]
        )

    # calculate the new state

    # calculate the new arm joint angles
    arm_joint_angles_new = arm_joint_angles_curr + arm_joint_velocities * delta_t
    # calculate the new wheel angles
    wheel_angles_new = wheel_angles_curr + wheel_velocities * delta_t
    # calculate the new gripper state
    gripper_state_new = [gripper_state]

    # concatenate the new state
    new_state = np.concatenate((odemetry_new, arm_joint_angles_new, wheel_angles_new, gripper_state_new))

    return new_state

def main():
    #test the next_state function
    delta_t = 0.01
    N = 3000
    max_joint_and_wheel_velocity = 5
    initial_state = np.array([0,0,0,0,0,0,0,0,0,0,0,0,0])

    current_state = initial_state
    control_input_0 = np.array([1,0,0,0,0,0,0,0,0,0])
    control_input_1 = np.array([0,1,0,0,0,0,0,0,0,0])
    control_input_2 = np.array([0,0,1,0,0,0,0,0,0,0])
    control_input_3 = np.array([0,0,0,1,0,0,0,0,0,0])
    control_input_4 = np.array([0,0,0,0,1,0,0,0,0,0])
    control_input_5 = np.array([0,0,0,0,0,1,0,0,0,0])
```

```
    control_input_6 = np.array([0,0,0,0,0,0,1,0,0,0])
    control_input_7 = np.array([0,0,0,0,0,0,0,1,0,0])
    control_input_8 = np.array([0,0,0,0,0,0,0,0,1,0])
    #append the control input into a list
    control_input = []
    control_input.append(control_input_0)
    control_input.append(control_input_1)
    control_input.append(control_input_2)
    control_input.append(control_input_3)
    control_input.append(control_input_4)
    control_input.append(control_input_5)
    control_input.append(control_input_6)
    control_input.append(control_input_7)
    control_input.append(control_input_8)

    state_trajectory = []
    state_trajectory.append(current_state)

    for i in range(N):
        j = int(i/300) % 9
        new_state = next_state(current_state, control_input[j], delta_t, max_joint_and_wheel_velocity)
        state_trajectory.append(new_state)
        # print(new_state)
        current_state = new_state

    print(state_trajectory[0:10])
    #writing csv files in Python
    with open("../../data/state_trajectory_1.csv", "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        for config in state_trajectory:
            writer.writerow(config)

if __name__ == "__main__":
    main()
```

1.2 video link

Here is the link: https://drive.google.com/file/d/1dAmGYtKkv8koplsVUYQ5pqKTXeb-3b7Q/view?usp=sharing