University of California San Diego

Course #: MAE204 Robotics

Final Project: Milestone 1

March 12, 2024

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

Contents

1 MileStone 1

1.1 Code

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

```python
import csv
from math import pi
import numpy as np
import modern_robotics as mr


def TrajectoryGenerator(
    Tse_initial, Tsc_initial, Tsc_final, Tce_grasp, Tce_standoff, k
):
    # Calculate the total time for the trajectory

    """
    Segement:
    1 A trajectory to move the gripper from its initial configuration to a "standoff" configuration a few cm above the block.
        #Trajectory segments 1 and 5 are longer motions requiring motion of the chassis.
        #Segment 1 is calculated from the desired initial configuration of the gripper to the
        first standoff configuration
        #The gripper trajectories could correspond to constant screw motion paths
        t = 3
        gripper state : 0
        Tse_initial -> Tse_standoff_1

    2 A trajectory to move the gripper down to the grasp position.

        # simple up or down translations of the gripper of a fixed distance. Good
        trajectory segments would be cubic or quintic polynomials taking
        a reasonable amount of time (e.g., one second)

        t = 1
        gripper state : 0

        Tse_standoff_1 -> Tse_grasp_1

    3 Closing of the gripper.

        t = 0.65
        gripper state: closed 0 -> 1

    4 A trajectory to move the gripper back up to the "standoff" configuration.

        t = 1
        gripper state : 1
        Tse_grasp_1 -> Tse_standoff_1

    5 A trajectory to move the gripper to a "standoff" configuration above the final configuration.

        t = 3
        gripper state : 1
        Tse_standoff_1 -> Tse_standoff_2
```

```

    6 A trajectory to move the gripper to the final configuration of the object.
        t = 1
        gripper state : 1
        Tse_standoff_2 -> Tse_grasp_2

    7 Opening of the gripper.
        t = 0.65
        gripper state: closed 1 -> 0

    8 A trajectory to move the gripper back to the "standoff" configuration.
        t = 1
        gripper state : 0
        Tse_grasp_2 -> Tse_standoff_2

    For each line return:
    r11, r12, r13, r21, r22, r23, r31, r32, r33, px, py, pz, gripper state
    """
    # Define the waypoints for the trajectory
    Tse_standoff_1 = np.dot(Tsc_initial, Tce_standoff)
    Tse_grasp_1 = np.dot(Tsc_initial, Tce_grasp)
    Tse_standoff_2 = np.dot(Tsc_final, Tce_standoff)
    Tse_grasp_2 = np.dot(Tsc_final, Tce_grasp)

    # Segment 1: Move the gripper to the standoff configuration above the block
    """
        t = 3
        gripper state : 0
        Tse_initial -> Tse_standoff_1
    """
    t1 = 3
    N1 = int(t1 * k / 0.01)  # Number of reference configurations for segment 1
    gripper_state_1 = 0
    traj_segmt1_list = generate_segment_trajectory(Tse_initial, Tse_standoff_1, t1, N1, 5 ,gripper_state_1)

    # Segment 2: Move the gripper down to the grasp position
    """
        t = 1
        gripper state : 0
        Tse_standoff_1 -> Tse_grasp_1
    """
    t2 = 1
    N2 = int(t2 * k / 0.01)  # Number of reference configurations for segment 2
    gripper_state_2 = 0
    traj_segmt2_list = generate_segment_trajectory(Tse_standoff_1, Tse_grasp_1, t2, N2, 5 ,gripper_state_2)
    # Segment 3: Close the gripper
    """
        t = 0.65
        gripper state: closed 0 -> 1
    """
    t3 = 0.65
    N3 = int(t3 * k / 0.01)  # Number of reference configurations for segment 3
    gripper_state_3 = 1
    traj_segmt3_list = generate_segment_trajectory(Tse_grasp_1, Tse_grasp_1, t3, N3, 5 ,gripper_state_3)

    # Segment 4: Move the gripper back up to the standoff configuration
```

```python
    """
        t = 1
        gripper state : 1
        Tse_grasp_1 -> Tse_standoff_1
    """
    t4 = 1
    N4 = int(t4 * k / 0.01)  # Number of reference configurations for segment 4
    gripper_state_4 = 1
    traj_segmt4_list = generate_segment_trajectory(Tse_grasp_1, Tse_standoff_1, t4, N4, 5 ,gripper_state_4)

    # Segment 5: Move the gripper to the standoff configuration above the final configuration
    """
        t = 3
        gripper state : 1
        Tse_standoff_1 -> Tse_standoff_2
    """
    t5 = 3
    N5 = int(t5 * k / 0.01)  # Number of reference configurations for segment 5
    gripper_state_5 = 1
    traj_segmt5_list = generate_segment_trajectory(Tse_standoff_1, Tse_standoff_2, t5, N5, 5
    ↪  ,gripper_state_5)

    # Segment 6: Move the gripper to the final configuration of the object
    """
        t = 1
        gripper state : 1
        Tse_standoff_2 -> Tse_grasp_2
    """
    t6 = 1
    N6 = int(t6 * k / 0.01)  # Number of reference configurations for segment 6
    gripper_state_6 = 1
    traj_segmt6_list = generate_segment_trajectory(Tse_standoff_2, Tse_grasp_2, t6, N6, 5 ,gripper_state_6)

    # Segment 7: Open the gripper
    """
        t = 0.65
        gripper state: closed 1 -> 0
    """
    t7 = 0.65
    N7 = int(t7 * k / 0.01)  # Number of reference configurations for segment 7
    gripper_state_7 = 0
    traj_segmt7_list = generate_segment_trajectory(Tse_grasp_2, Tse_grasp_2, t7, N7, 5 ,gripper_state_7)

    # Segment 8: Move the gripper back to the standoff configuration
    """
        t = 1
        gripper state : 0
        Tse_grasp_2 -> Tse_standoff_2
    """
    t8 = 1
    N8 = int(t8 * k / 0.01)  # Number of reference configurations for segment 8
    gripper_state_8 = 0
    traj_segmt8_list = generate_segment_trajectory(Tse_grasp_2, Tse_standoff_2, t8, N8, 5 ,gripper_state_8)

    # combine all the segments list into one list
    reference_configs = (
```

```python
        traj_segmt1_list
        + traj_segmt2_list
        + traj_segmt3_list
        + traj_segmt4_list
        + traj_segmt5_list
        + traj_segmt6_list
        + traj_segmt7_list
        + traj_segmt8_list
    )

    # Write the reference configurations to a .csv file
    with open("reference_trajectory_1.csv", "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        for config in reference_configs:
            writer.writerow(config)

    return reference_configs

def rearrange_back(list):
    """
    rearrange the 1x12 vector to a matrix T
    r11, r12, r13, r21, r22, r23, r31, r32, r33, px, py, pz, gripper state
    """
    return np.array(
        [
            [list[0], list[1], list[2], list[9]],
            [list[3], list[4], list[5], list[10]],
            [list[6], list[7], list[8], list[11]],
            [0, 0, 0, 1],
        ]
    )

def rearrange(T, gripper_state):
    """
    rearrange the matrix T to a 1x12 vector
    r11, r12, r13, r21, r22, r23, r31, r32, r33, px, py, pz, gripper state
    """
    return [
        T[0, 0],
        T[0, 1],
        T[0, 2],
        T[1, 0],
        T[1, 1],
        T[1, 2],
        T[2, 0],
        T[2, 1],
        T[2, 2],
        T[0, 3],
        T[1, 3],
        T[2, 3],
        gripper_state,
    ]

def generate_segment_trajectory(Xstart, Xend, Tf, N, method, gripper_state):
    """Computes a trajectory as a list of N SE(3) matrices corresponding to
      the screw motion about a space screw axis
```

```python
    :param Xstart: The initial end-effector configuration
    :param Xend: The final end-effector configuration
    :param Tf: Total time of the motion in seconds from rest to rest
    :param N: The number of points N > 1 (Start and stop) in the discrete
              representation of the trajectory
    :param method: The time-scaling method, where 3 indicates cubic (third-
                   order polynomial) time scaling and 5 indicates quintic
                   (fifth-order polynomial) time scaling
    :return: The discretized trajectory as a list of N matrices in SE(3)
             separated in time by Tf/(N-1). The first in the list is Xstart
             and the Nth is Xend.
             rearrange the matrix T to a 1x12 vector
    r11, r12, r13, r21, r22, r23, r31, r32, r33, px, py, pz, gripper state
    """
    N = int(N)
    timegap = Tf / (N - 1.0)
    traj = [[None]] * N
    for i in range(N):
        if method == 3:
            s = mr.CubicTimeScaling(Tf, timegap * i)
        else:
            s = mr.QuinticTimeScaling(Tf, timegap * i)

        T_curr = np.dot(
            Xstart, mr.MatrixExp6(mr.MatrixLog6(np.dot(mr.TransInv(Xstart), Xend)) * s)
        )
        traj[i] = rearrange(T_curr, gripper_state)

    return traj

def main():
    # Define the end-effector frame {e}
    M_0e = np.array([[1, 0, 0, 0.033], [0, 1, 0, 0], [0, 0, 1, 0.6546], [0, 0, 0, 1]])
    Tb_0 = np.array([[1, 0, 0, 0.1662], [0, 1, 0, 0], [0, 0, 1, 0.0026], [0, 0, 0, 1]])
    x = 0
    y = 0
    phi = 0
    T_sb = np.array([[np.cos(phi), -np.sin(phi), 0, x], [np.sin(phi), np.cos(phi), 0, y], [0, 0, 1, 0.0963],
    ↪  [0, 0, 0, 1]])
    T0_e = np.dot(Tb_0, M_0e)
    T_se = np.dot(T_sb, T0_e)
    Tse_initial = T_se

    Tsc_initial = np.array(
        [[1, 0, 0, 1], [0, 1, 0, 0], [0, 0, 1, 0.025], [0, 0, 0, 1]]
    )  # Initial configuration of the cube
    Tsc_final = np.array(
        [[0, 1, 0, 0], [-1, 0, 0, -1], [0, 0, 1, 0.025], [0, 0, 0, 1]]
    )  # Final configuration of the cube

    s45 = np.sin(np.pi/4)
    Tce_grasp = np.array(
        [[-s45, 0, s45, 0], [0, 1, 0, 0], [-s45, 0, -s45, 0], [0, 0, 0, 1]]
    )  # Configuration of the end-effector relative to the cube while grasping
    Tce_standoff = np.array(
        [[-s45, 0, s45, 0], [0, 1, 0, 0], [-s45, 0, -s45, 0.05], [0, 0, 0, 1]]
```

```
    )  # Standoff configuration of the end-effector above the cube
    k = 1  # Number of trajectory reference configurations per 0.01 seconds
    trajectory = TrajectoryGenerator(Tse_initial, Tsc_initial, Tsc_final, Tce_grasp, Tce_standoff, k)

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

1.2 video link

Here is the link: https://drive.google.com/file/d/1orBTCStKuJ3ZcXcjsz37vek52pOaVdMH/view?usp=sharing