ECE 486 Robot Dynamics & Control

Gennaro Notomista

PROJECT DESCRIPTION

Due date: Aug 1, 2024

Contents

1 Introduction and objective

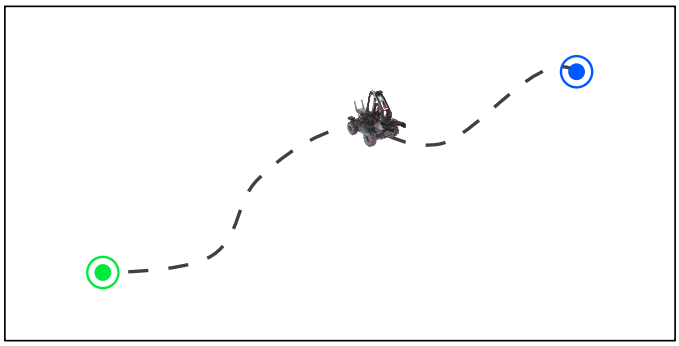

Figure 1: A mobile robot tracking a path from a start to a goal location.

Objective

The objective of this project is path planning and control for a mobile robot. The controller will leverage the property of differential flatness of the kinematic model of the mobile robot.

The approach will consist of the following tasks:

1. Kinematically feasible path planning
2. Feedforward controller based on the endogeneous transformation
3. Feedback controller based on pole placement

2 Instructions

2.1 Robot model

The robot is modeled as a unicycle, i.e. its kinematic model is

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega, \end{cases} \tag{1}$$

where x and y are the components of the position of the robot in the plane, θ is its heading, and v and ω are the longitudinal and angular velocity inputs, respectively.

2.2 Project tasks

2.2.1 Kinematically feasible path planning

Recall that the unicycle model is differentially flat, $z = [z_1, z_2]^T = [x, y]^T \in \mathbb{R}^2$ being the flat output. Therefore, from the knowledge of a smooth path $z(t)$, $t \in [0, T]$, $T > 0$, the control input signals $v(t)$ and $\omega(t)$ required to track it can be algebraically computed as follows:

$$\begin{aligned} v(t) &= \pm\sqrt{\dot{z}_1(t)^2 + \dot{z}_2(t)^2} \\ \omega(t) &= \frac{\ddot{z}_2(t)\dot{z}_1(t) - \ddot{z}_1(t)\dot{z}_2(t)}{\dot{z}_1(t)^2 + \dot{z}_2(t)^2}. \end{aligned} \tag{2}$$

Task 1

Plan a kinematically feasible path $z_d(t)$, $t \in [0, 10]$s, that starts at $z_s = [-2, -2]$m and ends at $z_g = [1.5, 2.5]$m. Assume the initial orientation of the robot is $\theta_s = 0$ and its final orientation is $\theta_g = -\frac{\pi}{6}$.

Plot the starting and goal locations together with the planned path in the simulator environment.

Hint: Since you need the path to be at least twice differentiable to compute $\omega(t)$ from (2), and you need to specify the initial and final orientation of the robot, you may want to consider a cubic spline interpolating the initial and final points.

2.2.2 Feedforward controller design

Task 2

Using the expression of the path $z_d(t)$ planned in Task 1, and the algebraic relations in (2), compute the input signals $v(t)$ and $\omega(t)$, $t \in [0, 10]$s required by the robot to track the planned path.

Plot the computed input signals over time in the interval $[0, 10]$s.

2.2.3 Feedback controller design

The controller evaluated in Task 2 is open loop as it is solely based on the planned path and does not account for the actual state in which the robot is at each point in time, but rather only the one in which it should be. If the kinematic model does not faithfully represent the motion of the robot or if there is a difference between the measured, or estimated, initial conditions of the robot and the actual ones, the open loop controller is doomed to fail.
We can then leverage state feedback control design to mitigate this problem. Let $w = [w_1^T, w_2^T]^T = [z^T, \dot{z}^T]^T \in \mathbb{R}^4$ and define its dynamics as follows:

$$\dot{w} = \underbrace{\begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}}_{A} w + \underbrace{\begin{bmatrix} 0 \\ I \end{bmatrix}}_{B} u_w, \tag{3}$$

where $u_w \in \mathbb{R}^2$ is the new control input.

Task 3

Design a state feedback controller of the following form:

$$u_w(t) = K_P(z_d(t) - z(t)) + K_D(\dot{z}_d(t) - \dot{z}(t)), \tag{4}$$

where $K_P > 0$ and $K_D > 0$ are controller gains.

Run 100 simulations, each of which with different initial conditions for w, $w_0 := w(0)$, drawn from a normal distribution with mean equal to $[z_s^T, 0^T]^T$ and standard deviation equal to 0.1 for all components, assumed to be independent, i.e.

$$w_0 \sim \mathcal{N}\left(\begin{bmatrix} z_s \\ 0 \end{bmatrix}, 0.1 \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \right). \tag{5}$$

Plot the trajectories of the robot over the course of the 100 simulations, overlapped, in the same simulator environment.

Hint: From the solution $w(t)$ of the dynamical system (3) with input u_w given by (4), use (2) to compute the input signals $v(t)$ and $\omega(t)$, $t \in [0, 10]s$ required by the robot to track the planned path.

2.3 Controller Implementation

The controller must be developed using the Python simulator provided on LEARN, under Content/Project/Code. The test script `test.py` shows how to use the functions provided in the simulator. In particular:

- ```
  robot = MobileManipulatorUnicycleSim(
                      robot_id=1,
                      robot_pose=[0.0, 0.0, 0.0],
                      pickup_location=[0.75, 0.75],
                      dropoff_location=[-0.75, -0.75],
                      obstacles_location=[[0.5, 0.5], [-0.5, -0.5]])
  ```
  initializes the robot at pose $[0, 0, 0]$. The `pickup_location` and `dropoff_location` variables should
  ```

be used to specify the starting and goal positions of the robot. Since obstacles are not used, you may move their location to be outside the environment by appropriately setting the variable `obstacles_location`.

- `robot.set_mobile_base_speed_and_gripper_power(`
  `v=0.1,`
  `omega=1.0,`
  `gripper_power=1.0)`
  moves the platform at 0.1 m/s forward, 1.0 rad/s counterclockwise. You may leave the `gripper_power` argument unset.

- `robot_pose, pickup_location, dropoff_location,`
  `obstacle_1_pose, obstacle_2_pose, obstacle_3_pose = robot.get_poses()`
  returns the poses of the robot, pick-up and drop-off locations, as well as of obstacles. Each pose is a list of x, y position coordinates, and orientation θ.

3 Performance evaluation

A successful implementation of the tasks described in the previous section consists of:

- A kinematically feasible path planned between the prescribed starting and final poses
- Two controllers, of feedforward and feedback type, respectively
- A report containing the detailed description of the approach adopted to solve the project tasks, alongside the required plots and references to the parts of the code which solve each part of the tasks
- The code written to solve the project tasks

Recall that the deliverables related to the project are as follows (including percentage of the overall course grade):

- Final project report: 15%
- Project code: 5%

More details on the format and the structure of the reports can be found on the syllabus.