ECE 486 / 687 Robot Dynamics & Control

Gennaro Notomista

PROJECT DESCRIPTION

Due date: Aug 4, 2025

Contents

1 ECE 486

1.1 Introduction and objective

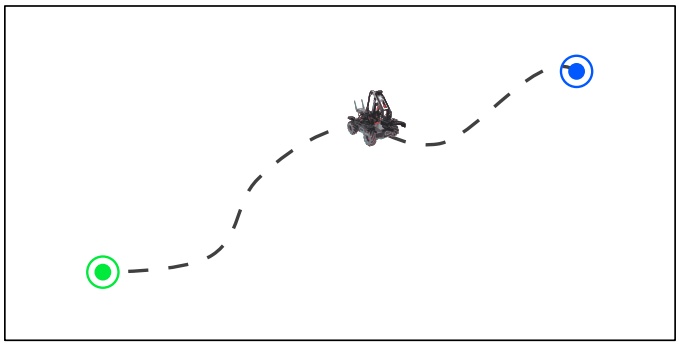


Figure 1: A mobile robot tracking a path from a start to a goal location.

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

1.2 Instructions

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. See also reference frames in Fig. 3b.

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 $z_d(t)$, $t \in [0, 20]$s, that starts at z_s and ends at z_g, for arbitrarily chosen z_s and z_g. 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.

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.

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 $\omega(t)$, $t \in [0, 20]$s required by the robot to track the planned path.

Plot the computed input signals over time in the interval $[0, 20]$s.

Plot the motion of the robot controlled using the computed input signals.

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.

Plot the motion of the robot controlled using the state feedback controller above.

Plot the input signals over the course of the experiment.

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 $\omega(t)$, $t \in [0, 20]s$ required by the robot to track the planned path.

1.3 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: 20%
- Project code: 5%

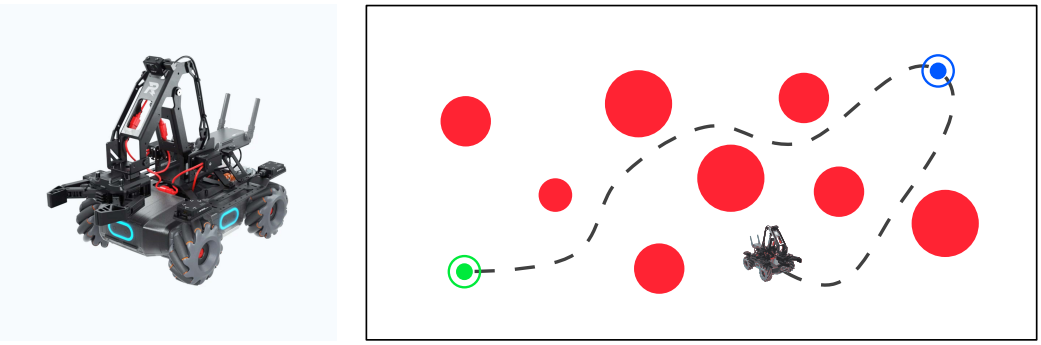
The format and the structure of the report must be as follows:

- Maximum length: 4 pages
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Proposed approach
  - Section II: Results
  - Section III: Discussion

The reports must include also the detailed description of the work carried out by each member of the group, including what sections of the report were written by whom.

2 ECE 687

2.1 Introduction and objective



(a) The mobile manipulator DJI RoboMaster EP used to execute the pick-and-place task.

(b) A pick-and-place scenario with a pick-up location (blue), a drop-off location (green), and obstacles (red).

Figure 2: Mobile manipulator robot and pick-and-place scenario of the project.

Objective

The objective of this project is to program the mobile manipulator DJI RoboMaster EP (Fig. 2a) to perform a pick-and-place task consisting in (see Fig. 2b):

- Navigating to a known pick-up location
- Picking up an object
- Navigating to a known drop-off location
- Droping off the object
- Avoiding obstacles placed in the environment at known locations

The mobile manipulator is comprised of a mobile base controlling using longitudinal and angular velocity inputs, carrying a robotic arm with a gripper mounted on its end-effector. The technical specifications of the robot are available here: `https://www.dji.com/ca/robomaster-ep/specs`.

2.2 Instructions

Tasks

Complete the following steps to achieve the pick-and-place task:

1. Assuming the mobile manipulator is at a known pick-up (resp. drop-off) location, write a function to pick up (resp. drop off) an object at that location
2. Design control Lyapunov functions (CLFs) to drive the robot to known pick-up and drop-off locations
3. Design control barrier functions (CBFs) to avoid obstacles placed at known locations in the environment
4. Synthesize the control input for the mobile platform using a quadratic program (QP) formulation to achieve safe navigation
5. Combine the controllers for the manipulator arm and the mobile platform developed above to realize the desired pick-and-place task

2.3 Performance evaluation

The robot should be able to

- Navigate to a known pick-up location
- Pick up an object
- Navigate to a known drop-off location
- Drop off the object
- Avoid obstacles placed in the environment at known locations

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

- Final project report: 40%
- Project code: 10%

The format and the structure of the report must be as follows:

- Maximum length: 4 pages
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Proposed approach
  - Section II: Results
  - Section III: Discussion

The reports must include also the detailed description of the work carried out by each member of the group, including what sections of the report were written by whom.
A short video (maximum 1 minute) to supplement the results may also be attached.

2.4 Alternative for MASc and PhD students

The project will consist of the solution to a problem in the student's research area using the techniques covered during the course.
In addition to the final deliverables, a proposal should be submitted following the instructions below:

- Maximum length: 1 page
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Problem description
  - Section II: Novelty and/or impact
  - Section III: How robot dynamics and control techniques play a key role
  - Section IV: Technical challenges
  - Section V: Metric for success
  - Section VI: Timeline
- Deadline: **June 1**

The format and the structure of the final report must be as follows:

- Maximum length: 4 pages
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Introduction
  - Section II: Literature review
  - Section III: Materials and methods
  - Section IV: Results
  - Section V: Discussion

A short video (maximum 1 minute) to supplement the results may also be attached.

3 Code

The controller must be developed on your PC using Python starting from the files provided on LEARN, under Content/Project/Code.

3.1 Robot API

The description of the files is as follows:

- `main.py`: Skeleton script to write project code
- `mobile_manipulator_unicycle.py`: Implementation of the class `MobileManipulatorUnicycle`—inheriting from `Robot` implemented in `robot.py`—to control the base and the arm of the mobile manipulator
- `README.md`: Instructions to get started with the code
- `robot.py`: Implementation of the class `Robot` handling the interface with the backend to receive robot poses from the camera system in the Robohub and to send control inputs to the robot
- `robot_control_comms.py`: Utility classes and functions related to communication with the backend
- `test.py`: Script to test the interface with the backend to send control inputs to the robot and receive its pose from the camera system in the Robohub

Note that you must, and need to, only modify the file `main.py` to solve the project. Everything else must not be changed. In the provided `main.py` script, the robot ID has to be set on line 4. For example, if you would like to use robot 3, then line 4 should be

```
robot = MobileManipulatorUnicycle(robot_id=3, backend_server_ip="192.168.0.2")
```

The object `robot` gives you an interface to the backend—which in turn interfaces with the robot and the tracking camera system of the Robohub—through the following methods:

- `set_mobile_base_speed_and_gripper_power`

Description	Send longitudinal and angular speeds, as well as gripper commands, to the mobile platform
Example of use	`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 about the local z direction (see also Fig. 3b), and opens the gripper
Example of use	`robot.set_mobile_base_speed_and_gripper_power(v=-0.05, omega=-2.0, gripper_power=-1.0)` moves the platform at 0.05 m/s backward, 2.0 rad/s clockwise about the local z direction (see also Fig. 3b), and closes the gripper

- `set_leds`

Description	Send RGB color to the robot LEDs
Example of use	`robot.set_leds(128, 32, 32)` to set the color of the LEDs to (128, 32, 32) in RGB format

- `set_arm_pose`

Description	The two input arguments control the position of the end effector
Example of use	`robot.set_arm_pose(25.0, 25.0)` moves the arm 25 mm forward and 25 mm upward

- `get_poses`

Description	Get poses of the robot, pick-up and drop-off locations, as well as of obstacles, in the global reference frame of the Robohub (see also Fig. 3b). Each pose is a list of x, y position coordinates, and orientation θ (e.g. `robot_pose` is the list containing the position and orientation of the robot, $[x_R, y_R, \theta_R]$)
Example of use	`robot_pose, pickup_location, dropoff_location, obstacle_1_pose, obstacle_2_pose, obstacle_3_pose = robot.get_poses()`

3.2 Simulator

You are encouraged to test your controller in simulation first. The test script `test_sim.py` shows how to use the functions provided in the simulator. These are a replica of the functions to interface with the real robot, so that the only line you'll need to change to test your controller in the Robohub is the constructor of the `MobileManipulatorUnicycle`. In particular:

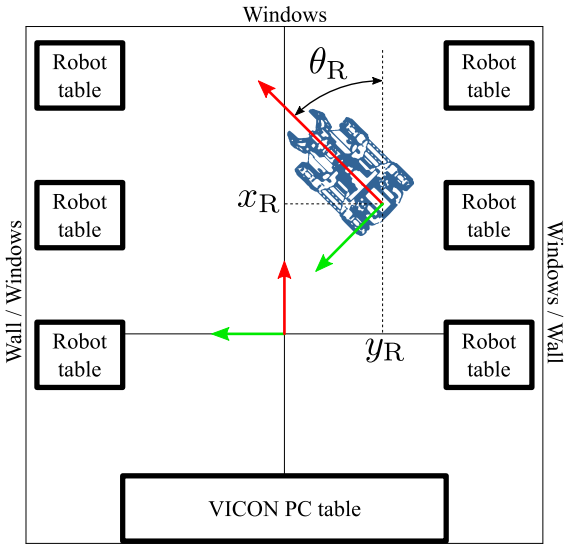
- ```
  robot = MobileManipulatorUnicycleSim(
                      robot_id=1,
                      robot_pose=[0.0, 0.0, 0.0],
                      pickup_location=[0.75, 0.75],
                      dropoff_location=[-0.75, -0.75],
                      obstacles_location=[[0.5, 0.5], [-0.5, -0.5]])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

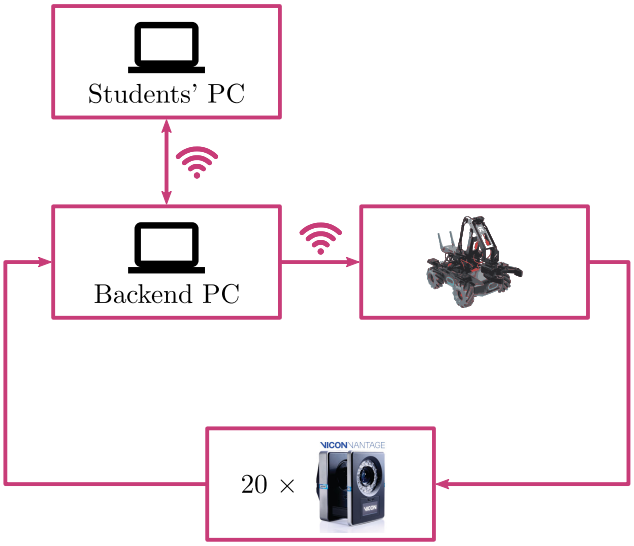
- ```
  robot_pose, pickup_location, dropoff_location,
  obstacle_1_pose, obstacle_2_pose, obstacle_3_pose = robot.get_poses()
  ```
  returns the poses of the robot, pick-up and drop-off locations, as well as of obstacles. The pose of the robot is a list of x, y position coordinates, and orientation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

4 Robohub schedule

(a) A humanoid, a mobile manipulator, and the ceiling camera system in the Robohub.



(b) Global reference frame in the Robohub (red and green arrows are x and y directions) and local reference frame of the robot (red and green arrows centered at the robot).



(c) Block diagram of the closed-loop control of the mobile manipulators in the Robohub.

Figure 3: The Robohub at University of Waterloo.

The Waterloo Robohub (Fig. 3a) is a collaborative robotics research facility located on the ground floor of Engineering 7. It hosts a diverse fleet of robots (humanoids, quadrupeds, manipulators, ground, and aerial mobile platforms) and it is equipped with an indoor positioning system comprised of a set of 20 Vicon Vantage V5 cameras. The RoboMaster EP robots are controlled according to the feedback control loop shown in Fig. 3c.

The schedule to perform project-related activities in the Robohub is reported in the table below (as well as in the course syllabus).

Date	**Time**	**Suggested activity**	
		ECE 486	**ECE 687**
Jun 16	8:30–11:30	Intro to RoboMaster EP	Intro to RoboMaster EP
Jun 30	8:30–11:30	Intro to RoboMaster EP	Intro to RoboMaster EP
Jul 7	8:30–11:30	Path planning / Quidditch	Object pick-up and drop-off
Jul 14	8:30–11:30	Feedforward control / Quidditch	Optimization-based control (navigation)
Jul 21	8:30–11:30	Feedback control / Quidditch	Optimization-based control (safety)
Jul 28	8:30–11:30	Feedback control / Quidditch	Optimization-based control