**CS449/549: Learning for Robotics**
Bilkent University
Fall 2024
Due date: November 05

# Homework 2

## Submission

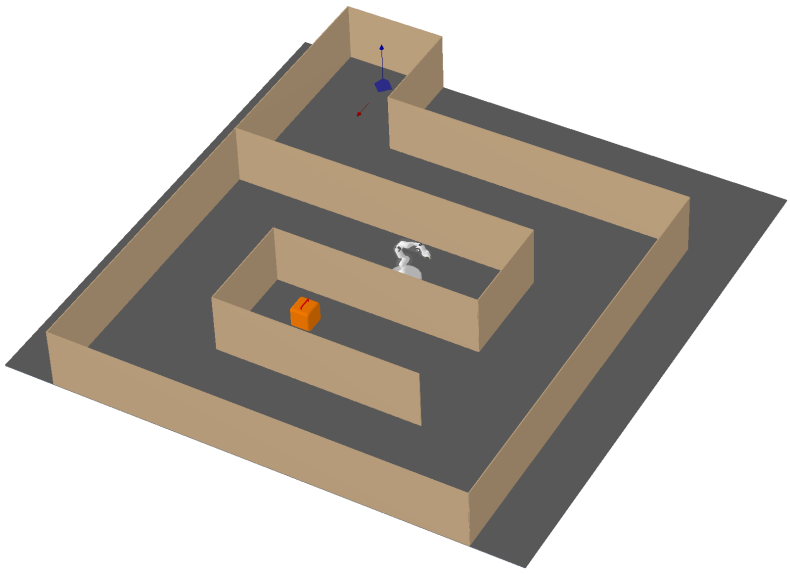
Please follow this instruction to submit your work.

- The coding will be done using the RAI (robotics) package introduced in the tutorials.
- You should upload all the Python files (.py and .ipynb), along with any '.g' files you created.
- You should include recordings demonstrating your solutions in the simulation.
- For your reports, you should upload a single PDF file containing your answers.
- You should submit a single ".zip" file, which involves all of the files in the working form. **Please ensure that all the files run smoothly without any need for modification when executed.**
- For your ."zip" files, please use this name convention "StudentNumber_HW2.zip"

## 1 Cargobot Manipulation

Congratulations! You and your team have been selected for the Cargobot project at a company. The engineering team has successfully designed a mobile cargobot. Now, it's your turn to implement the path planning algorithm for the robot. Your objective is to find the (optimal) path that enables the robot to pick up the cargo from its initial position in the map and place it in the designated goal area (shown by a large marker in the following figure).

**IMPORTANT:** In this question, you are STRICTLY PROHIBITED from adding intermediate waypoints by yourself. You should adhere to only using the available frames, when formulating your KOMO problem. You are provided with the necessary .g files in folder titled "Question-01" attached to this homework. Looking at the maze map for this question (see following figure), it is clear that this problem cannot be directly solved using KOMO, as there are numerous local minima that the robot might get trapped in. So, we will need RRT (Recall Tutorial 2) to get the correct solution for this problem. Be careful! In all your solutions, neither the robot nor the cargo should penetrate any walls!



## 1.1 Part A

Import all the .g files given in the directory Question-01/environment into your RAI environment. By doing so, you should be able to see a cargobot and cargo in the maze as shown in the figure above. You shall first obtain the inverse kinematics solutions for your cargobot. In this part, there are two tasks for you. In the first task, you need to formulate a KOMO problem that moves the cargobot gripper to cargo handle's position. Subsequently in the second task, you are required to formulate a KOMO problem that moves the cargobot from cargo position to designated goal area. Note that you don't need to get the whole path for this part, just the final joint states that place the robot at (1) "cargo position" and (2) "designated goal area" through inverse kinematics would be sufficient. **So, prescribe your KOMO problem/constraints using appropriate features (click here to get the list of features) accordingly.** Be careful! The Cargobot must not penetrate any walls in these solutions, or you will be unable to solve the following parts of this question.

## 1.2 Part B

Now, you'll use the RRT algorithm to find a path that moves the robot from its initial position to the cargo and then from the cargo's position to the goal area. Use the solutions from Part A as inputs to RRT. You should be able to obatain two paths:

1. From the Initial Position to the Cargo.
2. From the Cargo's Position to the Goal Area

**NOTE:**

1. You can skip modeling the pick-place mode switches/frame attachments in this part. A simple visualization of the robot's motion along the initial-cargo-goal trajectory is sufficient. That means you are allowed to skip visualizing the cargo moving along with the robot.

2. It is okay if RRT cannot find any feasible solution in your first try. Please design your overall solution/algorithm in a way to call your motion planner several times until it finds a feasible solution (approximately, 5-10 trial seems to be enough, but keep on trying until it finds a feasible solution).

### 1.3 Part C

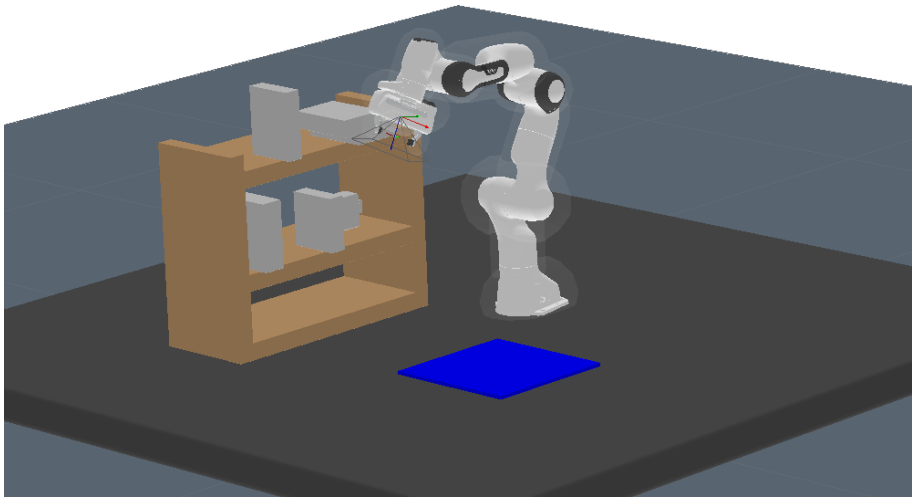
After you finish the path planning, it is now time for cargo delivery. Through the solution of this part, the cargobot should now be able to move to cargo's position, grasp the cargo from its handle and then take it along with itself to the goal area. You might want to check out the functions Config.attach() and Frame.unLink() for implementation of pick/place action primitives in this part. Your code should output the visualization of full trajectory of cargobot from its initial position to cargo, picking up cargo and then placing it in the designated goal area. We expect you to deliver a video demonstration of your solution along with the code for this part.

**Some Helpful Tips:**

Be careful with the usage of frame attachment/detachment functions. Make sure that both the cargobot gripper and cargo are at correct relative positions before you call these functions in the code. Before running the trajectory visualization loop, set the frame and joint states to their initial positions. Remove all temporary links used to simulate grasps, and recreate them at the appropriate times for the pick/place visualization.

## 2 Bookshelf Manipulation

In this question, you are provided with a simple bookshelf containing various objects. Your objective is to grasp each object in a natural way, similar to how a human would typically hold it. You should add task constraints to KOMO that guide the robot to solve the problem in a way resembling human behavior. After grasping the object, the robot should place it in the designated goal area.
You will focus on object manipulation using KOMO (K-order Markov Optimization). Your task is to manipulate objects exclusively through KOMO solutions without relying on predefined waypoints. You are expected to incorporate various features in your solutions to achieve this.
For more information on features, you can visit config_2_features.ipynb. While you may refer komo_3_manipulation.ipynb and manipulation.py for guidance, you are **prohibited from directly using the ManipulationModelling class and its methods**. Instead, you are expected to develop your own functions to perform pick-and-place tasks. A key requirement of this assignment is that your code should be generalizable. It should not only solve the specific problem presented but also be adaptable for future tasks with minimal modifications. For example, instead of hard-coding the grasping axis based on the name of the reference frame, your algorithm can decide the optimal approach direction for the gripper when grasping an object. You may code a heuristic method for grasping along different axes. Ensure that there are no collisions between the objects, environment, and the robot during the process.



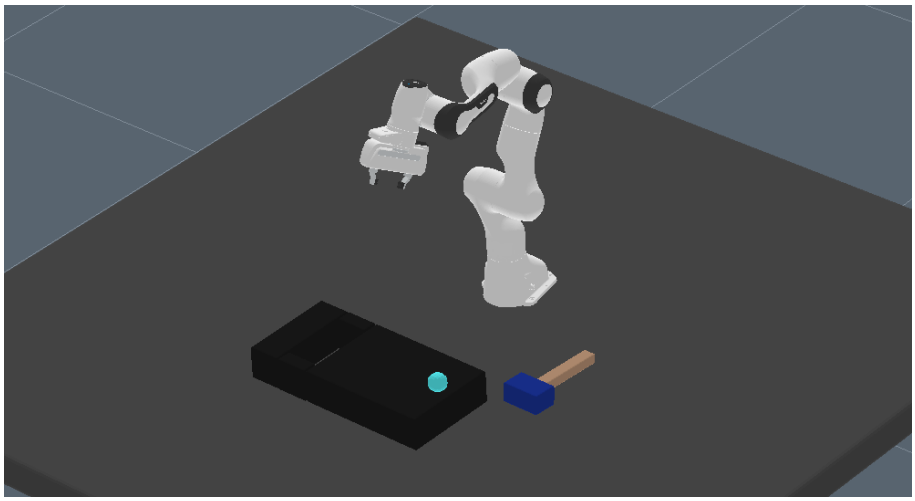
Below is a step-by-step guide for the task:

1. Add the "book_shelf.g" file to your configuration. Various objects are defined in the file. The objects you will work with are [book1, book2, book3, book4, and mug].
2. In each run, your algorithm should randomly select one of the objects and change the color of the selected object.
3. The robot should then grasp the object in a natural manner.
4. The robot should place the object in the goal area.
5. Finally, the robot should return to its home position.

For this question, we expect you to create a video demonstrating the solution to the pick-and-place task using KOMO for different objects. You can save screencasts for the different objects and combine them into a single video. Secondly, we expect you to write a report explaining your algorithm, the features you used, and the challenges you faced.

# 3 Mini-Golf

In this question, you will use BotOp to solve a mini-golf task by manipulating a tool. Like in real mini-golf, you must use a tool to move the ball; you cannot touch or manipulate it directly, but you can push the ball with the tool. You are required to create a heuristic method capable of solving the task. You can use waypoints in your heuristic. We have provided the "mini_golf.g" file for the mini-golf platform and a golf-club-like hammer to use.
Given that this simulation aims to mimic real-world conditions, you are restricted from directly attaching the tool. To grasp the tool, you should use the gripper. For more information on BotOp, you can visit botop_1_intro.ipynb.



For this question, we expect you to create a video demonstrating the solution to the mini-golf task using your heuristic. Secondly, we expect you to write a report explaining your heuristic. You may also include the pseudo-code of your heuristic.