Tutorial 4: Motion Planning using OMPL

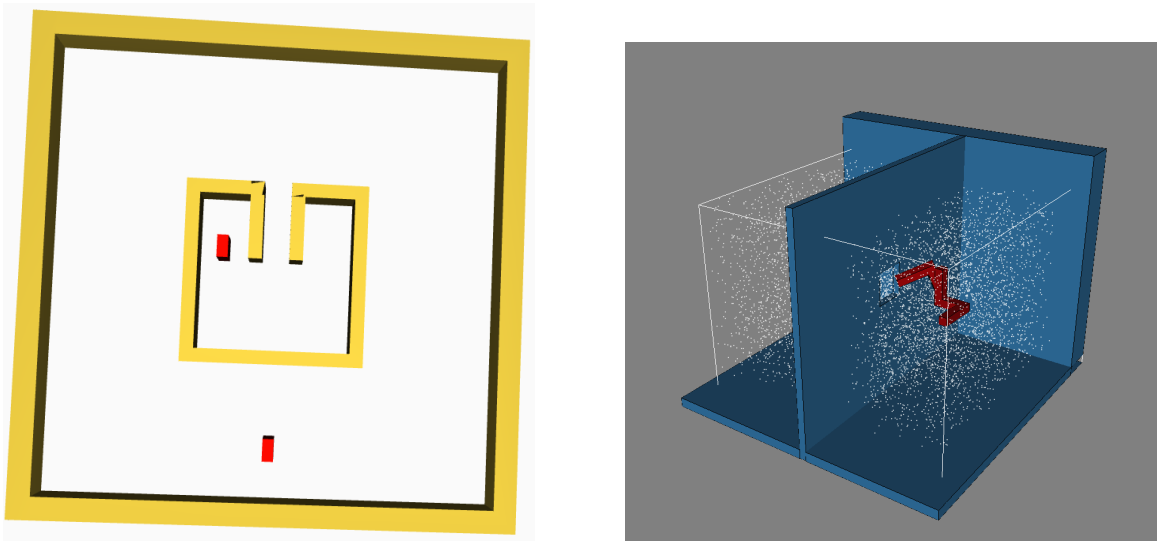

Figure 1: Example environments in OMPL

Introduction

In this tutorial, you will gain experience with the two sampling-based motion planning algorithms taught in the lectures—*Rapidly-exploring Random Trees* (RRT) and *Probabilistic Roadmaps* (PRM). To do so, you will use the *Open Motion Planning Library* (OMPL), a widely-used software package for planning in robotics.

The work is to be done individually and the answers to the questions posed are to be submitted via BlackBoard by the announced deadline.

Open Motion Planning Library (OMPL)

OMPL is a motion planning library consisting of implementations of numerous planning algorithms. It is written in C++ and has Python bindings. It is widely used in academia and industry, for example the *Robot Operating System* (ROS) uses OMPL as its default planner. The popular robot simulator V-REP has also recently switched to OMPL as its recommended planning infrastructure.

To get familiar with the basics of motion planning using OMPL, please refer to the OMPL Primer document here: http://ompl.kavrakilab.org/OMPL_Primer.pdf

There are two ways for you to use OMPL:

1. Do a local installation on your personal machine (only for Linux or Mac OS) following the instructions here: http://ompl.kavrakilab.org/installation.html
2. Use the desktop PCs in our lab (Roderic Hill 362) which we have pre-installed with OMPL.

To start using OMPL, open a terminal on your Linux or Mac OS machine, and use the command `ompl_app` from the directory where the OMPL is installed, under the subdirectory `gui`. To learn how to use the OMPL.app GUI, please refer to the brief introduction here: http://ompl.kavrakilab.org/gui.html

Question 1: Rigid Body Motion Planning

Set up the OMPL environment by opening the problem configuration file `Barriers.cfg`. This is located in the subdirectory `/resources/2D` in the directory with the OMPL. Verify that the OMPL parameters are set as follows:

- Robot type: `Rigid body planning (2D)`
- Start pose: `(X=35, Y=-75, Rotation=0)`
- Goal pose to `(X=600, Y=-300, Rotation=-180)`
- Optimization objective: `length`
- Time: `10.0`
- Collision checking resolution: `0.01`
- Goal bias: `0.05`

a) Planning with RRT [25%]

Set the planner to `RRT`.

- Press solve and animate showing both states and edges. Provide a screenshot of the resulting exploration tree showing both the explored states (nodes) and the edges.
- Clear and solve the task several times with the same parameters. Describe your observation and explain how that can be explained based on the RRT planning algorithm.
- Please try using different values for the parameters of your planner and re-run it several times to see the resulting motion plans.

b) Planning with PRM [25%]

Set the task parameters to their defaults for the problem configuration `Barriers.cfg`. Set the planner to `PRM` and verify that the `Max nearest neighbors = 8`.

- Press solve and animate showing both states and edges. Provide a screenshot of the resulting states and edges.
- Set `Max nearest neighbors = 30` and solve the task. Provide the same screenshot with the updated parameter and describe how the results differ and why.

c) Generic [20%]

- Explain what information each generated state (a.k.a. node or milestone) should contain for the motion plan to result in a feasible collision-free trajectory.
- Explain whether the original RRT and PRM algorithms are applicable to dynamic environments which require real-time motion planning? Explain why.

Question 2: Kinodynamic Motion Planning

Set up the OMPL environment by opening the problem configuration file `BugTrap_dcar.cfg`. Verify that the OMPL task parameters are set to the following:

- Robot type: `Dynamic car`
- Start pose: `(X=6, Y=-12, Rotation=0)`
- Goal pose to `(X=-40, Y=0, Rotation=0)`
- Optimization objective: `length`
- Time: `10.0`
- Propagation step size: `0.20`
- Control duration: `0-20`
- Goal bias: `0.05`

a) Tuning the parameters [15%]

Press solve and animate showing both states and edges using the RRT algorithm. Tune the parameters of your planner and clear and repeat solving the task several times until you have a successful motion plan. Provide a screenshot of the successful plan and provide the parameters that resulted in the successful plan.

b) Understanding [15%]

Please describe the observed differences between the motion plans generated for the rigid body problem and the kinodynamic problem. How do they differ and why?