TU Berlin Robotics WS2023/2024

Lab Assignment #5

Please upload your solution by **Sunday February 18, 2024 at 23:59**, using your ISIS account. Remember that this is a hard deadline, extensions impossible!

In this last assignment you will plan a collision-free path for the PUMA to maneuver an L-shaped gibbet end-effector through a small hole. We will provide you with accurate and complete models of the world and the PUMA robot, so you can work and test in a simulation. Finally, we will only execute the precomputed trajectory on the real robot. You will be supported by the *Robotics Library*[1].

RobLib - First Steps

The Robotics Library (RobLib) is an open source software that includes mathematics, kinematics, dynamics, hardware abstraction, visualization, collision detection, motion planning, and more.
To get started, follow the installation instructions (see slides and videos of the tutorial) and run the example:

```
cd <folder where you installed tutorialPlan>/build
./tutorialPlan
```

Press space—the example runs a planner, visualizes the search tree and the resulting path of `tutorialPlan/YourPlanner.cpp`. After planning, a path optimizer shortens the path.

Rapidly Exploring Random Trees (100 points)

In this assignment we investigate a family of motion planning algorithms called Rapidly-Exploring Random Trees[2] (RRT). In RobLib the basic RRT algorithm and some of its variants are already implemented. Your task is to develop and implement your own variant of an RRT algorithm. Your variant should improve the performance (= time until solution is found) without being overly tailored to the given problem (we will test your algorithm on a second, modified problem). You do not need to modify the path optimization method.

You can use the files `YourPlanner.h` and `YourPlanner.cpp` as a template for your own planner. Initially, `YourPlanner` implements the algorithm `RrtConConBase`, a bidirectional RRT-Connect. You can also take a look at other roblib planner implementations, for example `Rrt`, `RrtConCon` and `RrtExtExt`.

You can test your and any planner included in roblib with `TutorialPlanSystem.cpp`. You should go carefully through the code in there and test what happens when you use different planners and different parameters.

RRT Extensions (65 points)

Your main task is to come up with a good RRT planner—so be creative. To inspire you, here are some examples how you could extend the `RRTConConBase` variant:

- Change the distance metric.
- Change the nearest neighbor selection.
- Change the random sampling[3] strategy.
- Stop trying to extend exhausted nodes.

[1] http://www.roboticslibrary.org/

[2] `http://msl.cs.uiuc.edu/rrt/`

[3] Make sure to **adhere to joint limits**! You will be penalized if your submitted solution samples outside the joint limits of the robot.

Note that you are not only restricted to make changes in `YourPlanner.cpp`. For example, if you change the random sampling strategy you might want to edit `YourSampler.cpp`.

Points will be distributed for the following parts:

- Quality of extensions: Are the proposed extensions appropriate with respect to the given problem?
- Quantity of extensions: Think about different ways of improving the planner (**Minimum three**).
- Achieved improvements compared to `RrtConConBase` (see **Documentation**).
- The corresponding implementation of the extensions.

Documentation (35 points)

You are supposed to write a technical report with comments on the RRT-extensions you came up with. Be aware that points can be gained even for extensions that do not improve the runtime of the motion planner if you reason about the possible causes for this. The report should include at least the following sections:

- Explanation of the proposed extensions with a code snippet (e.g. screenshot) of your implementation.
- Reasoning about extensions that have not improved the runtime.
- Performance evaluation of your final algorithm (see below).
- The contribution table of your group (see section **Deliverables**).
- (Optional) 5 extra points can be gained by performing an ablation study of your extensions with regard to the runtime.

To evaluate the performance of your algorithm, compare your algorithm to `RRTConConBase`.
Use the following performance measures. benchmark.csv will contain the statistics for your evaluation:

- average computing time (`avgT`),
- standard deviation of computing time (`stdT`)
- average number of nodes in the computed search trees (`avgNodes`)
- average number of collision queries required for tree construction (`avgQueries`).

You should run your algorithm and the base code we give you (i.e., RRTConConBase) **ten times each**, and compare them in two ways: Using the start and end positions (planning from start to end), and using them reversed (planning from end to start).
Fill out the table below with your results:

	RRTConConBase	RRTConConBase (reversed)	Your Planner	Your Planner (reversed)
avgT				
stdT				
avgNodes				
avgQueries				

Deliverables

- A technical report as described in section **Documentation**.
- The adjusted scripts that we provided you, with plenty of meaningful comments! **Do not** add additional scripts to the project.
- Feel free to play around with all the settings in TutorialPlanSystem.cpp. But keep in mind, we will evaluate your code with the provided default version of TutorialPlanSystem.cpp, so be cautious not to change the interface to your planner class.
- The solution path planned by your algorithm, which we can execute on the PUMA. The solution path of the planner is written into the file `trajectory.txt` every time you run `tutorialPlan`. The file has the following format:

```
q1(t1) q2(t1) q3(t1) q4(t1) q5(t1) q6(t1)
q1(t2) q2(t2) q3(t2) q4(t2) q5(t2) q6(t2)
q1(t3) q2(t3) q3(t3) q4(t3) q5(t3) q6(t3)
....
q1(tn-1) q2(tn-1) q3(tn-1) q4(tn-1) q5(tn-1) q6(tn-1)
q1(tn)   q2(tn)   q3(tn)   q4(tn)   q5(tn)   q6(tn)
q1(tn-1) q2(tn-1) q3(tn-1) q4(tn-1) q5(tn-1) q6(tn-1)
....
q1(t3) q2(t3) q3(t3) q4(t3) q5(t3) q6(t3)
q1(t2) q2(t2) q3(t2) q4(t2) q5(t2) q6(t2)
q1(t1) q2(t1) q3(t1) q4(t1) q5(t1) q6(t1)
```

  with $q_i(t_j)$ being the angle (radians) of the i-th joint at time step t_j.
  (The path first goes forward - from initial configuration to goal - and then backwards - from goal to initial configuration.)

Explanation and template for implementation table

- Every group member needs to be able to **answer "high level" questions about *ALL* aspects** of the assignment.
  - We will check that during the presentations with general questions. Everyone needs to be able to answer these.
- For **implementation**, please **specify which team member** worked on which extension.
  - We will check that during the presentations with implementation specific questions.
  - You can split up the implementation of extension. But over all, every one needs to contribute equally to the implementation. (Writing the report does **not** count as *contributing to the implementation* and is accounted for differently as indicated in the table below.)
- Please use the following template in your submission. Differentiate between contributions to the implementation and documentation of your proposed extensions:

Student Name	Ext. 1 - Impl.	Ext. 1 - Doc.	···	···	Ext. n - Impl.	Ext. n - Doc.
Albert Albono	x	x				
Betty Barlow	x					x
Charlie Crockett			x	x		
Daisy Dolittle					x	x