CS 4610/5335: Robotic Science and Systems (Spring 2025) X. Zhi Tan
Northeastern University version: 1.0.0

HW 3: Sample Based Motion Planning

Please remember the following policies:

- Submissions should be made electronically via the Canvas. Please ensure that your solutions for both the written and/or programming parts are present and zipped into a single file.
- You are welcome to discuss the programming questions with other students in the class. However, you must understand and write all code yourself. Also, you must list all students (if any) with whom you discussed your solutions to the programming questions.
- Any use of generative AI must follow the class' generative AI policy.

Depending on the configuration of your computer, there are different ways to run the code in this homework. We recommend using conda with the attached environment file.

This homework uses the same Conda environemnt as homework 2. If you do not have the environemnt set up, you can do so by following these steps:

- Install the latest version of mini-conda from conda's website: conda.io
- Download `exercise3.zip` from Canvas. Unzip the file to your preferred directory (e.g., `/Users/Zhi/RSS/HW3`).
- Navigate to the folder using your terminal
- Create a conda environment using the following command: `conda env create --name "rss-hw3" --file=rss_env.yml`
- Open the folder using your favourite IDE and select `rss-hw3` as your intepreter.

The following are common build or run errors that you might run into:

- `AttributeError: module 'pybullet' has no attribute 'performCollisionDetection'`
  You are likely using an older version of pybullet. Make sure the pybullet version is 3.2.6.

Submission Instructions

Your submission on Canvas should be a single zip file containing your work. Your writeup should be a single PDF file. Most formats other than pdf cannot be viewed in the Canvas grading portal. The zip should have the same file structure as the `exercise3.zip` download. Your submission PDF submission should not be larger than 50mb. If the size limit is an issue, please check with a TA. At the top of each file, list your name, email, and other students with whom you discussed this homework.

The following practices are banned, unless specifically requested in the homework, and may result in a points deduction.

- Including spaces in path or file names. Use underscores.
- Adding matplotlib `plt.show()` commands other than the ones specified in the homework.
- Making code chages outside of designated locations
- Importing libraties not included in the anaconda env.
- Printing diagnositics info inside a loop. Your homework should not print thousands of lines of code.
- Using compression formats other than zip.

Doing any of the above makes grading take longer.

1 Sampling-based Motion Planning

In the `motion_planning` folder, you will find:

- `hw3_motion.py`: Main file - run `python hw3_motion.py -q <questionNum>` with an appropriate question number (0–6) to run the code for that question. Do not modify this file! (Feel free to actually make changes, but check that your code runs with an unmodified copy of `hw3_motion.py`.)
- `M0.py` – `M5.py`: Code stubs that you will have to fill in for the respective questions.
- `robot.py` – Defines the robot and its basic IK/FK/collision checking functionality.
- `\Robot` – the URDF and additional scripts for the robot.

In this homework you will implement two sampling-based motion planning algorithms, the probabilistic roadmap (PRM) and the rapidly exploring random tree (RRT). A 4-DOF 2-link arm has been defined, together with a spherical obstacle. The cylinders depicting the arm's links should not collide with the obstacle.

Note, sampling based algrothims can fail to find a solution, even when working corretly. The TA's will run your code several times using a script. For both the PRM and RRT algorithms, your solution should be able to find a path at least 80-90% of the time. Your code should run without errors every time. If implemented efficiently, your code should run in a few seconds or less. If it takes longer than 20 seconds you likely have an issue and should try to improve. The TA's will have to run each assigment several times, so submitting code that takes a very long time to run will slow down grading and may result in a points deduction.

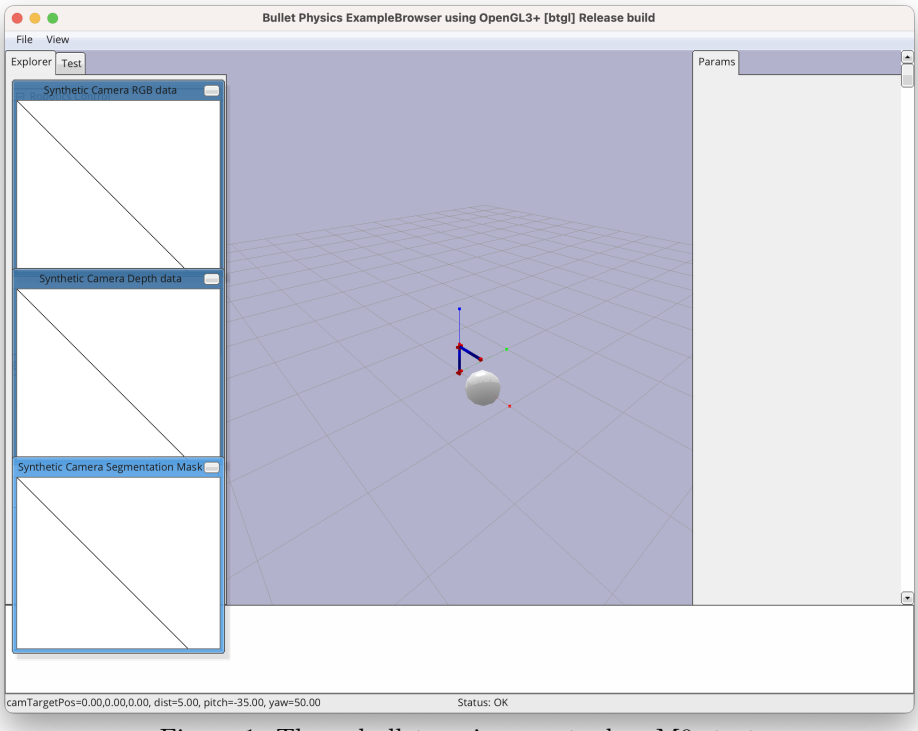

Figure 1: The pybullet environment when M0 starts

M0. **0 points.** Explore M0 and see how the robot works. The code will open pybullet and move to the joint angles given by M0. As you may notice, the robot ignores all collision.

M1. **1 points.** Given the provided joint limits, generate uniformly distributed samples from configuration space. The configurations should be within the joint limits, but they can be in collision. You would not be able to verify the correctness of this function until later questions.

M2. **2 points.** Implement the PRM algorithm to construct a roadmap for this environment. Your graph should contain `num_samples` collision-free configurations within joint limits. For each vertex, consider the nearest `num_neighbors` neighbors, and add an edge if the segment between them is collision-free. We will use `networkx.Graph` to store the roadmap. NetworkX has some visualization tools that can be used to check the connectivity of the graph, but you should avoid leaving any `plt.show()` commands in your work because this can make it annoying to run the homework in a script. Note: we do collision checking by setting the robot joint in the simulator. This means you will see the robot "jump" between multiple positions as it does collision checking.

M3. **2 points.** Use the roadmap to find a collision-free path from the start configuration to the goal configuration. You will need to select appropriate points to get "on" and "off" the roadmap from the start and goal respectively.

M4. **2 points.** Implement the RRT algorithm to find a collision-free path from the start configuration to the goal configuration. Choose appropriate hyperparameter values for the step size and frequency of sampling q_{goal}. Remember to traverse the constructed tree to recover the actual path. Do not use NetworkX to implement your RRT (it will make your code run very slowly when building a large tree). You should use an appropriate tree data structure. One simple way to do this is to implement your tree as two lists, one to store nodes, and another to keep track of the index of the parent of each node.

M5. **2 points.** The path found by RRT likely has many unnecessary waypoints and motion segments in it. Smooth the path returned by the RRT by removing unnecessary waypoints. One way to accomplish this is to check non-consecutive waypoints in the path to see if they can be connected by a collision-free edge.

M6. **0 points.** If M4 and M5 have been implemented correctly, this question should require no further implementation. Watch your algorithm work on a more challenging motion planning problem.

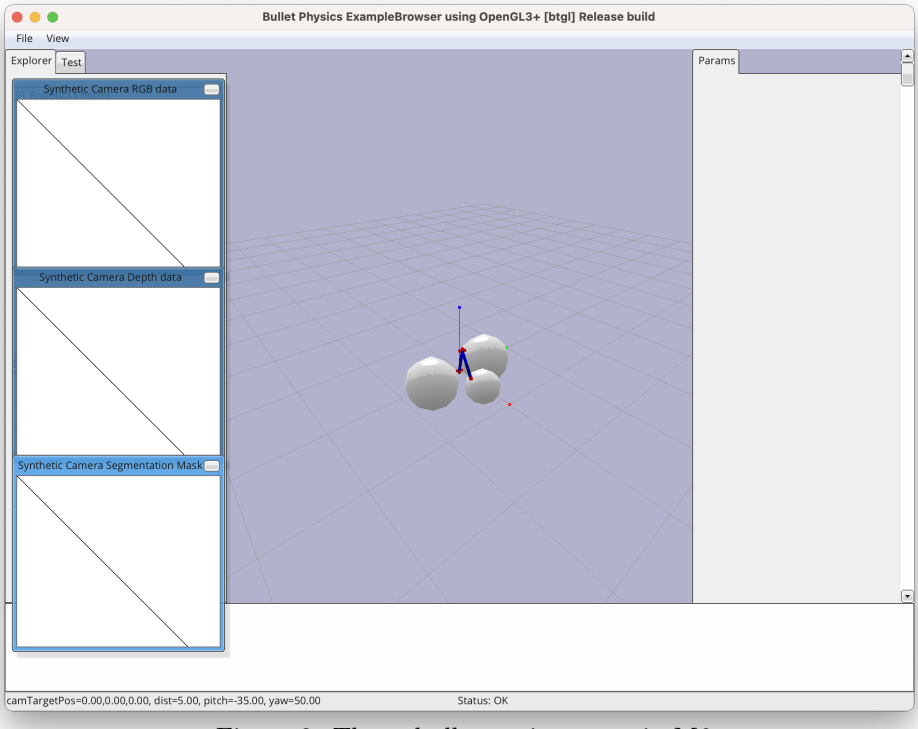

Figure 2: The pybullet environment in M6