ROB 520: Motion Planning
Winter 2022
Homework Assignment #1
Due 1/26/2022 at 2:59pm

Rules:

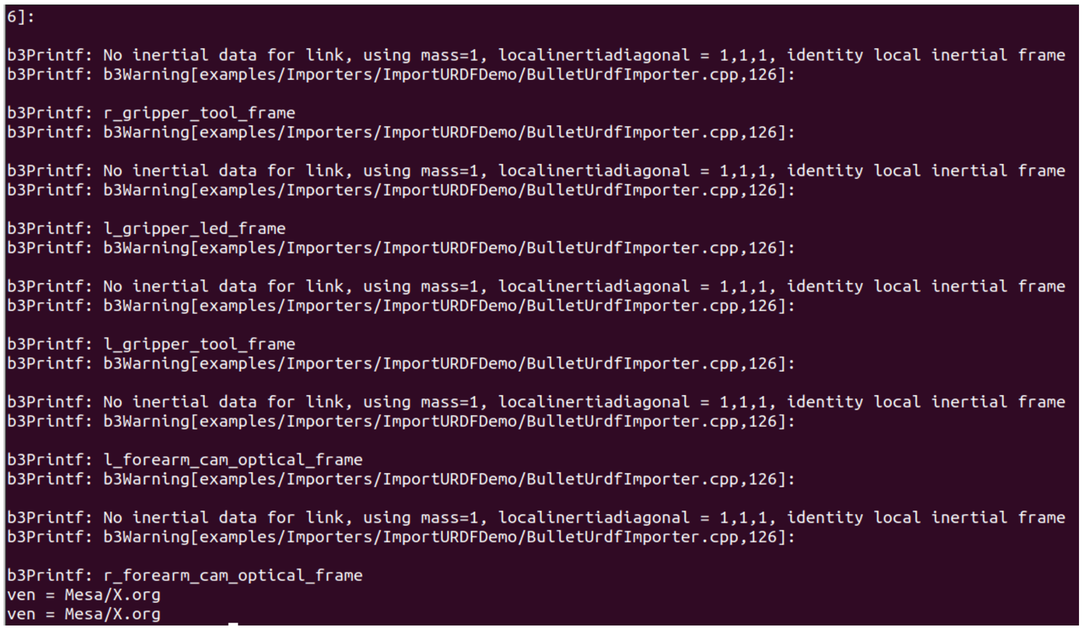
1. **All homework must be done individually, but you are encouraged to post questions on Piazza.**
2. No late homework will be accepted.
3. **You must show all your work to receive credit for your solutions.**
4. The goal of this homework is to familiarize you with openrave and some motion planning concepts.
5. Submit your python code along with a pdf of your answers in a zip file to Canvas.

Questions

1. (10 points) Chapter 4 exercise 5
2. (15 points) Chapter 4 exercise 8
3. (10 points) Chapter 4 exercise 14
4. (10 points) Chapter 4 exercise 16

Software

1. Install Ubuntu 20.04. It is best to use a separate partition, but if that is not possible, you can use a virtual machine. DO NOT USE ANY OTHER LINUX DISTRIBUTIONS OR VERSIONS OF UBUNTU FOR THIS COURSE.
2. Read through the python tutorial here. You will need to use Numpy to complete parts of this assignment. Some other resources are also available:
   - Numpy tutorial from Stanford's CS231n (link)
   - Numpy for `matlab` users (link)
3. Install `pybullet` by doing `sudo apt-get install python-pip` and then `pip3 install pybullet`
4. Download and unzip `pybullet_interface.zip`. Run `demo.py` and verify that you see the PR2 robot in a scene with a table. You will be asked to press enter multiple times to demo some `pybullet` capabilities. When you run `pybullet` with the PR2 robot, you will see warnings in the terminal that look like:



```
6]:

b3Printf: No inertial data for link, using mass=1, localinertiadiagonal = 1,1,1, identity local inertial frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: r_gripper_tool_frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: No inertial data for link, using mass=1, localinertiadiagonal = 1,1,1, identity local inertial frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: l_gripper_led_frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: No inertial data for link, using mass=1, localinertiadiagonal = 1,1,1, identity local inertial frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: l_gripper_tool_frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: No inertial data for link, using mass=1, localinertiadiagonal = 1,1,1, identity local inertial frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: l_forearm_cam_optical_frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: No inertial data for link, using mass=1, localinertiadiagonal = 1,1,1, identity local inertial frame
b3Printf: b3Warning[examples/Importers/ImportURDFDemo/BulletUrdfImporter.cpp,126]:

b3Printf: r_forearm_cam_optical_frame
ven = Mesa/X.org
ven = Mesa/X.org
```

These are issues with the definitions of the robot links/frames and will not affect your code in any way, so please ignore them.

Quick Information about PyBullet

`pybullet` is a physics engine that is used for simulating both soft and rigid bodies, including robots. We will NOT be using pybullet for physics simulation, but only for visualization and collision checking. To do this we've provided you with several helper functions in `utils.py` so that you can implement your motion planners and visualize the results. These helper functions should make it so that you don't have to worry about how `pybullet` works. You should never need to use an actual `pybullet` function to complete the assignment, the only interface should be through the helper functions we provide for you. To show you how to use these functions we've created a demo script called `demo.py`. You can refer to this script when writing your code to see how to use the helper functions.
Below are some tips for using `pybullet` that can help with your coding/debug process:

- **Helper Functions**: The helper functions are located in the `pybullet_tools` folder. Two files contain the main helper functions for you to use in this assignment: `utils.py` and `more_utils.py`, though you may use some of methods in `transformations.py`.
- **Accessing Robots and Obstacles**: An easy way to get all the robots and obstacles in the scene is to use the `load_env` function in `more_utils.py`. See the example code in `demo.py` to see how to use it.
- **Joint Groups**: A robot can have many degrees of freedom (e.g. a humanoid), but you may not want to plan for all of those DOFs at the same time. For instance, you may want to drive the mobile base of a robot but not move the robot's arms. You can see in `demo.py` where the names of the relevant joints are selected. Note that we sometimes refer to the "base joints" of the robot to denote the position and rotation of the base (a more accurate term would be "base DOF").

- **Viewer**: The `pybullet` viewer comes with some handy functionality to allow you to zoom in and out and focus on different parts of the scene:
  - Scroll Wheel or (CTRL (or ALT) + Right Mouse drag): Zoom camera in and out
  - CTRL (or ALT) + Left Mouse drag: Rotate camera
  - CTRL + Middle Mouse drag: Translate camera
- **Virtual Machines**: If you are running `pybullet` on a virtual machine (VM), make sure to allow the VM to use at least 2 cores and as much video memory as possible. This will make the rendering faster and will allow you to control the GUI more fluidly).

Implementation

The following implementation problems should be done in python. Feel free to look around the internet for example code for reference, but you should implement your own code from scratch unless explicitly stated otherwise.

1. (5 points) Make a new script called `tables.py`, and load in the `scenes/pr2doorway.json` scene. This will show the PR2 robot in an environment with several tables and a doorway. Set the poses of the tables so that they are all on PR2's side of the scene. You will need to use the `set_pose` function in `utils.py`. Include a screenshot of the scene in your `pdf`. Include this script in your `zip`.

2. (10 points) Make a script called `checkcollision.py` that loads in the `scenes/pr2doorway.json` scene. Set the pose of the PR2 so that it is near one of the tables, but is not touching it. Find joint values for the left arm of the PR2 robot so that the PR2 touches the table with any part of its left gripper. You may determine these joint values through experimentation. Confirm that the robot is touching the table by printing out the result of a call to `collision_fn` as in `demo.py`. Include a screenshot in your `pdf`. Include this script in your `zip`.

3. (10 points) Make a script called `drawing.py` that loads in the `scenes/pr2doorway.json` scene. Draw a set of red rectangles around the boundaries of the top of every table in the environment. You will need to use a series of lines to draw the rectangles. Hint: you can use the `get_aabb` function in `utils.py` to return the Axis-Aligned Bounding Box (AABB) of any body.

   Next draw 35 blue points in a circle that encompasses the environment shown in the scene. Make sure the points are large enough to be easily visible in a screenshot. Include a screenshot showing the points and the rectangles in your `pdf`. Include this script in your `zip`.

4. (20 points) Make a copy of your `checkcollision.py` script and call it `path.py`. Edit the script to execute a path for the left arm of the PR2 so that it starts below the table and moves above it without colliding. You should set the configurations in the path manually (see an example in `demo.py`). Execute the path on the robot and verify that it does not collide with the table. Finally, draw a line connecting the positions of the `l_gripper_tool_frame` poses for all configurations in the path. Include a screenshot of the scene in your `pdf`. Include this script in your `zip`.