Homework 0

Robot Autonomy
CMU 16-662, Spring 2016

TAs: Laura Herlant, Shushman Choudhury*

1 Introduction

In this homework, we will go through some basics for python and OpenRAVE. For this homework, you must write your own code and submit your own writeup/results, though you are free to work with anyone in the class. This homework is intended to get you acquainted with python and OpenRAVE, and learn how to find help online.

2 Python

For those without any python experience, we recommend this tutorial: `http://www-inst.eecs.berkeley.edu/~cs188/fa11/projects/tutorial/tutorial.html`. It goes through the basics very quickly. You **do not** need to submit any of the assignments there, but we find this to be a good reference.

You will find a file `python_tutorial.py` in the **code** directory. It initializes a dictionary of coffee shops and values. There are three places labeled `TODO`. You must write a sorting function in three different ways:

1. Manually sort using loops. No need to get fancy or be efficient with your sorting method, just get some practice iterating through data structures.
2. Sort using python's built in sorting functions, either list.sort or sorted. One neat thing about python is that you can specify which default values you are changing. So, for example, if you provide `sorted` with an additional argument `reverse=True`, then the sort will have the reverse order.
3. Sort using numpy's built in sorting function, either sort or argsort.

You can run the script with `python python_tutorial.py`. The output should sort **in descending order of value**. The solution is:

```
[['Espresso a Mano', '0.95'], ['21st Street', '0.9'], ["Tazza D'Oro", '0.75'],
    ['Voluto', '0.6'], ['Commonplace', '0.5'], ['Coffee Tree', '0.45'], ['
    Crazy Mocha', '0.35']]
```

Note that this is a list of lists - each solution should print either a list of tuples, or a list of lists. You need to make sure each of your three functions prints out the list in this order.

3 OpenRAVE

Next you'll find a script `openrave_tutorial.py` in the **code** directory. Most of the functions have a `TODO` tag meant for you to fill them in. Go ahead and run the script - you should see a window pop up with HERB over a green square. **Note**: There are two ways to run the script - with python, just type `python`

*Adapted from assignment by Kyle Strabala

`openrave_tutorial.py`, and for IPython, follow the instructions commented in the main method and start IPython with `ipython`, and then do `execfile('openrave_tutorial.py')` in the command prompt. This will show the popup window and give you a prompt to put commands in. Play with the buttons on the right. Here are some useful items:

- Click the top one (looks like a mouse pointer), then click on HERB. He should be surrounded by a green box, indicating he's collision free. Grab HERB and drag him, and you can move him into the ground, turning the box orange, indicating he's in collision. You can grab the box around HERB to rotate him too.
- With the mouse pointer, you can also see the coordinates of the object your mouse is over, and the name of the kinbody. Move it around HERB's arms to see the names of the different components.
- The glove goes back into the default mode of rotating the camera. You can also translate the camera with middle click.
- The flashlight (alternatively, hitting the 's' key) and clicking on a point zooms in, and centers the camera rotation frame there. Now when you rotate the camera, it rotates about that point.
- Click View -> Geometry -> Collision Only. HERB is actually approximated by geometries simpler than the entire Mesh, enabling faster collision checking.

Once you have become acquainted with the interface, it is time for some programming. Your tasks are:

1. Write the function `move_straight` to move the robot forward by the specified distance in the direction it is currently facing. See Fig 1.
2. Write the function `rotate_by` which rotates the robot about the z-axis by the specified angle. Note that there should be no translation - only rotation.
3. Write the function `go_around_square` which puts HERB at each corner of the red box, XYZ coordinates $(1, 1, 0), (1, -1, 0), (-1, 1, 0)$, and $(-1, -1, 0)$. Make HERB face the center of the square at each coordinate. No need to get complicated - you can hard code it (though it would be good practice to compute it!). You can use your other functions. See the left image in Fig 1 for an example of being at one corner facing the center.
4. OpenRAVE robots have specified Degrees of Freedom (DOFs). HERB has 24 of them. Figure out which indices correspond to the Right Arm, Right Hand, Left Arm, Left Hand, and Head.
5. Now we'll change one of the DOFs. Normally, OpenRAVE won't let you move beyond joint limits, which will keep the robot safe. This safety feature can be disabled, however. Make one of HERB's arms turn such that it intersects his body. Note that much like the additional `reverse=True` argument from sorting, you may find an optional argument for `SetDOFValues` useful.
6. Modify the `hw0.env.xml` file to add a sphere. Add it as position $X = 1, Y = 1, Z = 0.5$ with radius $0.2m$. It should look like the sphere in Fig 1. Note that this is what we're loading to add HERB to the environment, the floor, and lines.

See Section 6 for locations of examples and documentation. One thing to note - as good practice, you should use `with self.env` whenever modifying the robot (there are examples of this as well).

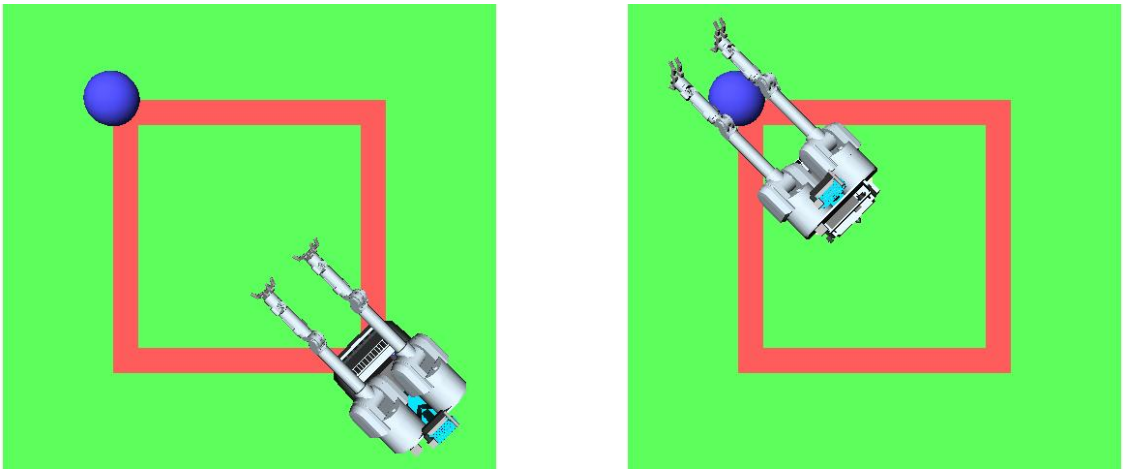

Figure 1: Example of the function usage. Here, we show one point on the square you should put the robot at, facing the center. The `move_straight` function is used to move HERB forward a specified amount. Finally, note the additional sphere placed in the world.

4 Deliverables Checklist

Homework submission instructions are located on the website. The following files are expected from you:

- A modified `python_tutorial.py`
- A modified `openrave_tutorial.py`
- A modified `hw0.env.xml`
- A writeup in pdf form (name it `hw0-<andrew-id>.pdf`)

5 Grading

This is how the assignment will be evaluated:

- Each of the sorting functions is worth one point (3 pts total). You don't need to write anything up for this - your code needs to run, and it needs to follow the rules for each function.
- Write the `move_straight`, `rotate_by`, and `go_around_square` (1 pt each). Include images in your writeup showing this working.
- Specify the 5 index ranges of DOFs for the Right Arm, Right Hand, Left Arm, Left Hand, and Head. (1 pt)
- Put HERB in self collision, and include an image in your writeup. (1 pt)
- Add the sphere, and include an image in the writeup showing it. (1 pt)
- Tell us how long this homework took you. (1 pt)

Some points will be deducted if you forget to lock the environment when altering the robot.

6 Where to get help

As always, you should try searching Google first. You will find that many of the hits point to the following pages.

- Quick python tutorial
- In-depth python tutorial
- OpenRAVE Main Page and C++ API. Many of the Python functions are only documented in the C++ API. I usually search in the C++ API first, then search on the main page which also searches through the examples (examples page 1, examples page 2).
- Eigen refernce manual: The Geometry page talks about transformations.
- ROS via Python We won't be using ros, but as many roboticists do, you may find useful information in the ros tutorials and message boards.
- OpenRAVE getting Started

Those are but a few of the pages that you will end up looking at. If you are truly stumped, or if you have clarification questions, then post a message on the HW0 Discussion Board on Blackboard. You should help each other out as best you can without giving away the answer.