Perception and Decision Making in Intelligent Systems
Homework 3: A Robot Navigation Framework
Announce: 11/7 , Deadline: 12/5 23:59

Introduction

In the HW2, we have reconstructed a 3D semantic map of **apartment_0**. Our next goal is to enable a robot to move to a desired destination (for example, **navigate the robot to find a specific item**). Therefore, in this homework, we focus on how to navigate from point A to B using the RRT algorithm. (You only need to do on first floor of apartment_0)

There are three main tasks:

1. **2D semantic map construction**
   In homework 2, we already have a semantic point cloud. You can use your result to get 2D semantic map for navigation.

2. **RRT algorithm implementation**
   We use the RRT algorithm to find a navagitable path from starting point A to goal point B (a specific item).

3. **Robot navigation**
   An agent can navigate automatically by following the path calculated by RRT to find specific items.

In this homework, you only need to do on first floor of apartment_0.
You can use either .py or .ipynb to show your results.

Implementation

Part 1: 2D semantic map construction

The following describes the steps for generating a 2D semantic map.

1. Remove ceiling and floor points of the point cloud constructed in HW2

2. Save coordinates and colors of the points (the color should be the same as color map for 101 categories we use in HW2)
3. Use any libraries (e.g. matplotlib) to plot a scatter graph (x-coordinate, z-coordinate in the point clouds) with with the points you saved
4. Save the map as **"map.png"**

We provide a 2D semantic map of the first **floor of apartment_0 for your reference**. You can download it from this link. If you choose to use the provided 2D semantic map, you can start from **Step 3**. Of course, feel free to use your own reconstructed map.

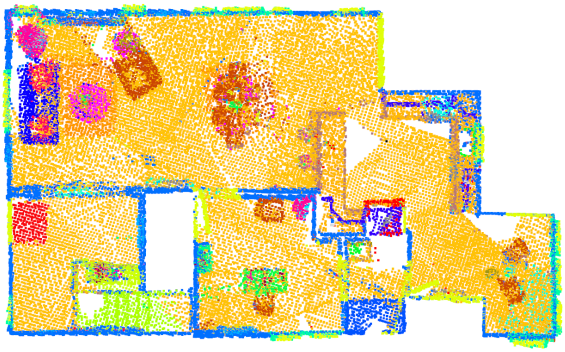
Notes for Step 3:

- **Scale relationship** between coordinates in **point.npy** and coordinates in apartment_0 is: **apartment_0 = points array * 10000. / 255.**

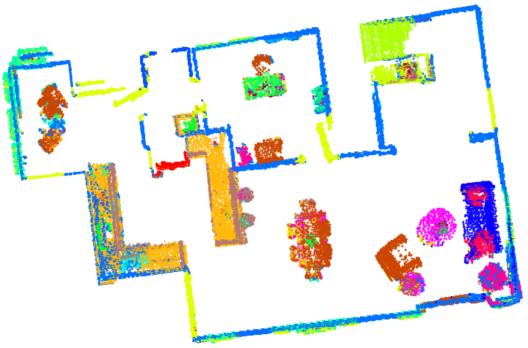
- There are two versions of the color array, the value of rgb in **color_01.npy** is in range [0, 1] while **color_0255.npy** is in range [0, 255]. If you have problems using **color_0.npy** to find labels because of floating-point errors, you can use **color_0255.npy.**

Note: You need to find the relationship between the map (pixel) coordinate and the Habitat coordinates, it's necessary for Part 3. **You can add extra points to calculate the scale, or create different map to help you**.

Example map:



[After removing ceiling]



[Removing ceiling and floor]

Part 2: RRT Algorithm

You will implement the RRT algorithm, which calculates a navigable path from the **"map.png"**. For more detail of the algorithm, please check details on this link RRT or the slide on the class.

You need to search for these target categories, including **rack, cushion, lamp, stair, and cooktop.** For the corresponding colors, you can check color map for 101 categories .

Input:

1. **A target category you want to search**
   Just input a string (ex: **rack**)

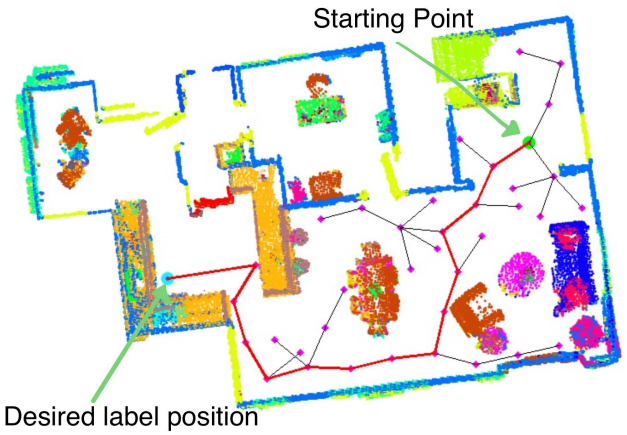
2. **Choose a starting point on the map**
   You can reference the way we use in Example.

Output:

1. **A map with a path from a starting point to a target point**
   The target point is a point **in front of the target item**. You can use any method to decide the location of the point.

2. **Path points on this route**
   The xy-coordinate calculated by RRT is in the pixel coordinate. To navigate in Habitat, you need to convert them into the coordinate of **the first floor of apartment_0.**

Example of a RRT output:



Part 3: Robot navigation

1. Based on the path you get in Part 2, let the agent move using these three commands : “move_foward”, “turn_left”, “turn_right”.
   - You can modify load.py to do navigation.
   - The xy-coordinate calculated by RRT is in the pixel coordinate. We need to convert them from pixel coordinate to xyz-coordinate in Habitat.
   - We can define the amount of how much the agent should move and turn for each step (By adding the following lines in function: make_simple_cfg)

```
# Here you can specify the amount of displacement in a forward action and the turn angle
agent_cfg.action_space = {
    "move_forward": habitat_sim.agent.ActionSpec(
        "move_forward", habitat_sim.agent.ActuationSpec(amount = 0.25)
    ),
    "turn_left": habitat_sim.agent.ActionSpec(
        "turn_left", habitat_sim.agent.ActuationSpec(amount = 10.0)
    ),
    "turn_right": habitat_sim.agent.ActionSpec(
        "turn_right", habitat_sim.agent.ActuationSpec(amount = 10.0)
    ),
}
```

2. Please highlight the target with a **transparent mask** while navigating, so that we can clearly visualize the target category.
   - Ref: tutorial_adding_images

3. During the agent moving, collect the RGB images and save them as a video (or gif) **“{target_name}.mp4”**

Example of transparent mask

Example

Example Demo Video (Target object: refrigerator)

Grading

Online Demo 30%

We will specify a target label and a starting point, you should show us the corresponding navigation result on the **first floor of aprartment_0**.

Report 70%

Your report should include the following content:

1. **Implementation (50%)**
   a. **Code**
      Detailed explanation of your implementation.
      For example:
      - How do you do RRT algorithm?
      - How do you convert routes to discrete actions?
   b. **Result and Discussion**
      - Show and discuss the results from the RRT algorithm with different start points and targets.
      - Discuss about the robot navigation results
      - Anything you want to discuss

2. **Questions (20%)**
   a. In RRT algorithm, you can adjust the step size and bias (the number balance between exploration and exploitation). Please explain how the two numbers affect the RRT sampling result. (15%)
   b. If you want to apply the indoor navigation pipeline in the real world, what problems you may encounter? (5%)

Bonus 10%

Try to improve the RRT algorithm, compare and discuss it with the original one.

Submission

Due Date: 2023/11/28 23:59

Please directly compress your code files and report (.pdf) into {STUDENT ID}_hw3.zip and submit it to the New E3 System.

The file structure should look like:

```
{student_id}_hw3.zip
├ README.md (Explain how to run your code)
├ report.pdf
├ src
│  ├ main.py or main.ipynb
│  └ (other code files if you have)
└ results
   ├ rack.mp4 or rack.gif
   …
   └ cooktop.mp4 or cooktop.gif
```

Wrong submission format leads to -10 point

Late submission leads to -20 points per day