Perception and Decision Making in Intelligent Systems
Homework 2: A Robot Navigation Framework
Announce: 10/1 , Deadline: 10/14 23:59

Introduction

In this assignment, we will use a semantic map of **apartment_0** to enable a robot to move to a desired destination (for example, **navigate the robot to find a specific item**). Therefore, this homework focuses on navigating from point A to B using the RRT algorithm. (You only need to do it on the first floor of apartment_0)

There are three main tasks:

1. **2D semantic map construction**
   Use the provided semantic point cloud to get a 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 it on the 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. We provide a 3D semantic map of the first floor of apartment_0. You can download it from this link. Remove ceiling and floor points of the point cloud.
2. Save coordinates and colors of the points (the color should be the same as color map for 101 categories)
3. Use any libraries (e.g. matplotlib) to plot a scatter graph (x-coordinate, z-coordinate in the point clouds) with the points you saved
4. Save the map as **"map.png"**

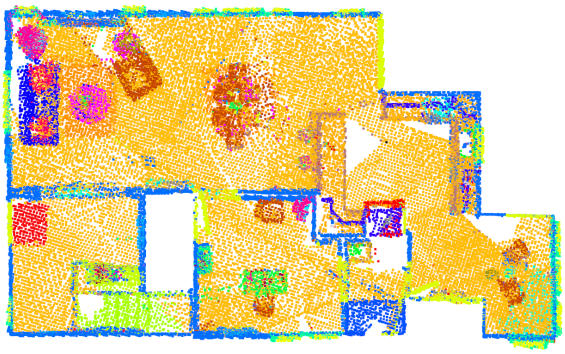
Notes for Step 3:

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

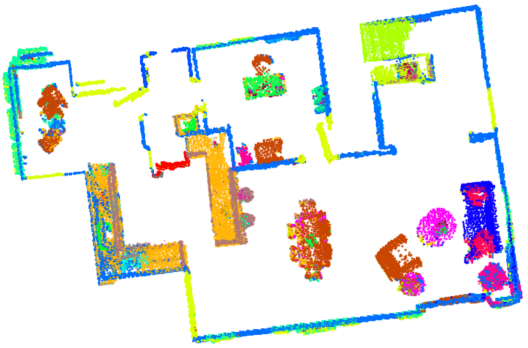
- There are two versions of the color array. The value of RGB in **color_01.npy** is in the range [0, 1] while **color_0255.npy** is in the range [0, 255]. If you have problems using **color_0.npy** to find labels because of floating-point errors, you can use **color_0255.npy.**

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 maps 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 details on the algorithm, please check details on this link, RRT, or the slide in the class.

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

Input:

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

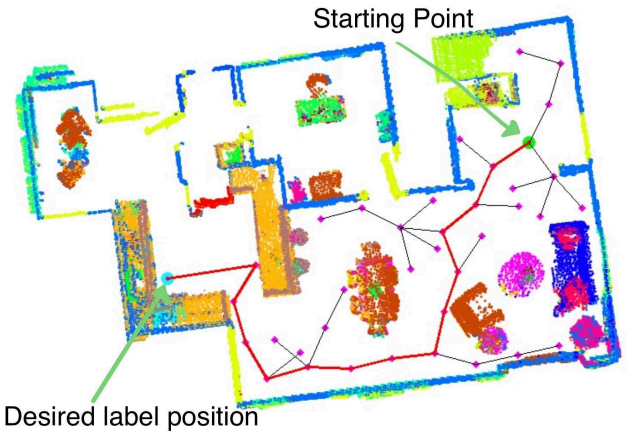
2. **Choose a starting point on the map**
   You can reference the way we use it in Example.

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 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”, and “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 coordinates 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 the 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”**

Examples of a transparent mask

Example

Example Demo Video (Target object: cooktop)

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 apartment_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 the 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
   c. **Reference**
      - Any reference you take

2. **Questions (20%)**
   a. In the 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 may you encounter? (5%)

Bonus 10%

Try to improve the RRT algorithm by comparing and discussing it with the original one.

QA page

The following link is to the QA page for hw2. If you have any questions, please ask them on this Notion page. We will answer them as soon as possible.
https://lopsided-soursop-bec.notion.site/HW2-QA-Sheet-bd7cc22e59604320a12849244894e7b9?pvs=4

Submission

Due Date: 2024/10/14 23:59

Please directly compress your code files and report (.pdf) into {STUDENT ID}_hw2.zip and submit it to the New E3 System.

The file structure should look like:

```
📂{student_id}_hw2.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

Plagiarism is strictly prohibited, including assignments written on GitHub. If we find any instances, it will result in a zero score without any room for objection.