Perception and Decision Making in Intelligent Systems
Homework 1: 3D Scene Reconstruction
Announce: 9/17, Deadline: 10/15 23:55

1. Introduction

In this homework, we will use the **Habitat simulator** as our main platform and **Replica** as our scene dataset.
In first homework you need to reconstruct a 3D map of a scene which uses the data collected by yourself in the apartment_0, one of the scenes from replica datasets (https://github.com/facebookresearch/Replica-Dataset)

There are three main tasks to complete: **Data Collection**, **Point Cloud Alignment**, and **Camera Pose Estimation**.

1. Data Collection:
   To reconstruct a scene, you need to control the agent walking through the whole scene, meanwhile saving the observation of the agent(RGB images, depth images), and the poses of the sensors.

2. Point Cloud Alignment and Reconstruction:
   Since the agent is moving, the position of the sensor is changing by the time, so we need to align them using the ICP algorithm.
   (There will be **two versions** of alignment, one based on your implementation, the other uses open3d function. For the further description please check out **Tasks** part 2.)

3. Camera Pose Estimation and Visualization:
   Visualizing the trajectory and show the difference between the trajectory we computed by icp algorithm and ground truth trajectory.

2. Requirements

The requirement of the development environments:

- OS : ubuntu 18.04 (Do not use virtual machine)
- Python 3.6 (You can use conda to create new environment)
- opencv-contrib-python
- Open3d
- Habitat lab and sim
  following the link https://github.com/facebookresearch/habitat-lab
  to install it.

3. Tasks

Part 1: Data Collection

1. Download the replica datasets from the link below.
Apartment_0 :
https://drive.google.com/file/d/1zHA2AYRtJOmlRaHNuXOvC_OaVxHe56M4/view?usp=sharing
Note : You can change **agent_state.position** to set the agent in the first or second floor ((0, 0, 0) for first floor and (0, 1, 0) for second floor.

2. Use the load.py to explore the scene and do data collection. (This is just a simple code to view the scene)
https://drive.google.com/file/d/1grlijScjDNAnQIO4JyfUjr3u03TyuS2I/view?usp=sharing

Part 2: Point Cloud Alignment and Reconstruction:

Pipeline:

1. Unproject depth images t_i and t_{i+1} to reconstruct two point clouds
2. Apply global registration, which are used as initialization of the local methods (you may need to do **voxelization to your point cloud for reducing the number of points**)
3. Apply local registration, using ICP to obtain the transformation matrix
4. Align point cloud at t_{i+1} to point cloud at t_i
5. Apply same process to the whole trajectory

In this part, you need to implement **two** functions:
(**depth_image_to_point_cloud** and **local_icp_algorithm)**
For the second function, you should first use **open3d icp** to do it, and then implement it by yourself. (you will have two versions of local icp) And for the rest, you can use the open3d library.

1. **depth_image_to_point_cloud**: get point cloud from rgb and depth image.
   input: rgb image, depth image, intrinsic matrix
   output: point cloud list with color
   Note: For intrinsic parameters, the agent camera uses a pinhole camera model (resolution 512 x 512) and horizontal and vertical FOV are 90 degrees.
   depth_scale = 1000

2. a.) **Using open3d local ICP (i.e. point to point ...)**
   b.) **local_icp_algorithm** : obtain the transformation matrix
   input: point cloud_1, point cloud_2, an initial transformation matrix, threshold
   output: transformation matrix
   Here is the reference you can read:
   https://cs.gmu.edu/~kosecka/cs685/cs685-icp.pdf

While finishing these functions, we can simply reconstruct a 3D map of the environment by the transformation matrices .

Part 3: Camera Pose Estimation and Visualization

1. Add the ground truth trajectory into our 3D scene
2. Add the estimated trajectory into our 3D scene
3. Transform the trajectories and the 3D scene into same coordinate system (be aware of the direction of coordinate and the scale)
4. Compute the Euclidean distance between estimated camera poses (x y z) and groundtruth camera poses (x y z), and calculate the mean and output on the screen.

After finishing the above 3 parts, we can obtain a 3D map with trajectories. Finally, Please remove the ceiling and you can compare the difference between your work and ground truth.
(Note : you will have **two versions of results**. First one is using open3d local icp, the other one is your own icp algorithm)

4. Example results

Red line: The estimated camera pose
Black line: The ground truth camera pose
Note: the below results are the screenshots.

The first floor of apartment_0:

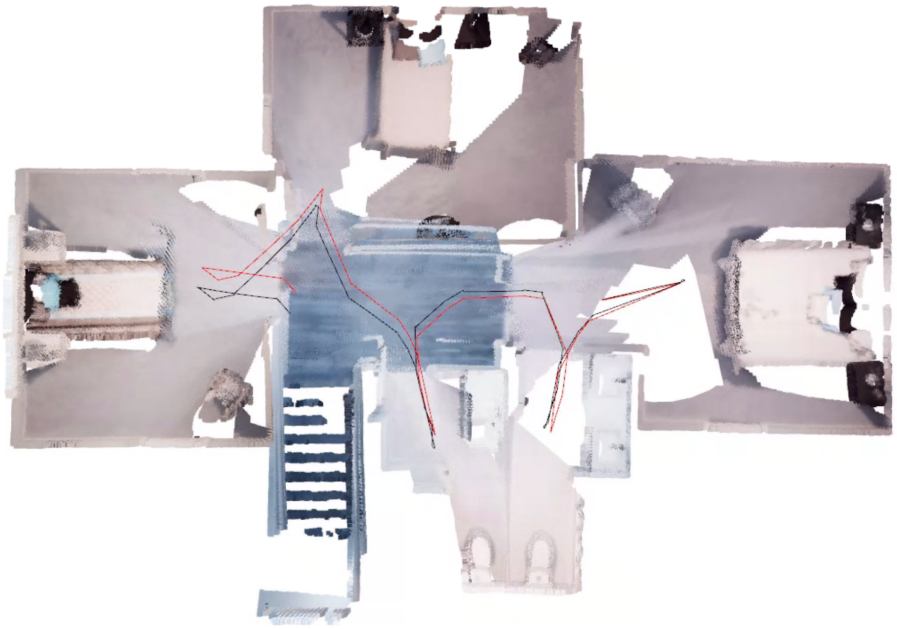
Data size = 152

Data size = 129

The above two results show the importance of the data collection step. Although collecting data in the same scene, with different data size, trajectory, distance to the object, the reconstructed result differs a lot.

The second floor of apartment_0:

Data size = 213



5. Submission

1. **File Formate:**

```
HW1_StudentID ----- load.py      # only for exploring the scene and save the data
                  | --- reconstruct.py # put your implementation code here
                  | --- README
                  | --- Data_collection |---first_floor    ----- rgb    ----- 1.png
                                                                      | ---  2.png
                                                                      | --- ...
                                                          | --- depth   -----  1.png
                                                                      | ---  2.png
                                                                      | --- …
                                                          | --- GT_Pose.txt
                                        |---second_floor  -----  rgb    ----- 1.png
                                                                      | ---  2.png
                                                                      | --- ...
                                                          | --- depth   -----  1.png
                                                                      | ---  2.png
                                                                      | --- …
                                                          | --- GT_Pose.txt
```

Readme: Please describe how to run your program.
Please put all the files into folder **HW1_StudentID** and compress your folder into **one "zip" file**. Submit it to New E3 System with the format **HW1_StudentID.zip.**

2. The **deadline** for this homework is **10/15 23:55.**
3. Late submission leads to **-20 points per day**.
4. Online Demo: we will run **reconstruct.py** directly, please give a relative path inside **reconstruct.py.**

6. Grading

1. Online Questions (explain depth unprojection and ICP) 30%
2. Online Demo (3D reconstruction and pose overlay) 70%
   We expected that using Open3d ICP will get a good result. But for your own ICP, you don't need to do as perfect as Open3d ICP.
3. Bonus (if your own local icp algorithm can do as good as open3d icp) 10 %