Perception and Decision Making in Intelligent Systems
Homework 1:
BEV projection and 3D Scene Reconstruction
Announce: 9/20, Deadline: 10/11 23:55

1. Introduction

In this homework, you are required to use **Habitat** and **Replica** to complete this homework. You are required to implement BEV projection and 3D reconstruction using your own collected data in the apartment_0, one of the scenes from replica datasets (https://github.com/facebookresearch/Replica-Dataset).

There are two tasks: **BEV projection and ICP alignment and Reconstruction**

Task1: BEV projection

1. Data Collection:
   Save a pair of front view and BEV image at the same time.
2. BEV Projection:
   Select points in the BEV (top view) image on the ground, mark an area enclosed by these points, and project the marked area from the BEV image to the perspective (front view) image.

Task2: ICP Alignment and Reconstruction

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

2. Point Cloud Alignment and Reconstruction:
   Since the agent is moving, positions of the camera are changing, so we need to align their coordinate system by using the ICP algorithm.

(There will be **two versions** of alignment, The first one is your own implementation, the second one uses open3d implementation. For the further description please check **Tasks** part 2.)

3. Camera Pose Estimation and Visualization:
   Visualize the trajectory and show the difference between the trajectory you estimate by ICP algorithm and ground truth trajectory.

2. Requirements

The requirement of the development environments:

- OS : ubuntu 18.04 , 20.04
- Python 3.6, 3.7 (You can use conda to create new environment)
- opencv-contrib-python
- Open3d
- Habitat-lab and Habitat-sim
  following the link https://github.com/facebookresearch/habitat-lab to install it.

3. Detail implementation

Task1: **BEV projection**

Part 1: Data Collection

1. Please download the replica dataset 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. Collect pairs of BEV view and front view image in the same time step. You can modify this file to finish your homework
https://drive.google.com/file/d/1Epjszg90BHyQm3ITRSQSw0LsvfrFvLzk/view?usp=sharing

Part 2: BEV Projection

1. Please read a BEV image and top view image
2. Please use the mouse to click points on a BEV image.
3. Please mark the area enclosed by these points
4. Project the marked area onto a perspective (front view) image
5. Save the projectected image to the .png file.

- **For intrinsic parameters, an agent camera utilizes a pinhole camera model (the resolution of an image is 512 x 512)**
- **Horizontal and vertical FOV are 90 degrees.**
- **We have implemented a click_event() method in the bev.py. You can modify bev.py as you wish.**

Example results

Note: the results below are the screenshots.

BEV 1.png

front 1.png

BEV 2.png

front 2.png

Task2: **ICP alignment**

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. You need to modify this file to save observation)
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 first which is used as the initialization of the local methods. (you may need to do **voxelization to your point cloud for reducing the number of points for reducing memory usage and speedup.**)

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 the same process to the whole trajectory(Hint: matrix multiplication).

   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 or point to plane)**
   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

After finishing these functions, we can simply reconstruct a 3D scene 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 L2 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 scene 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:

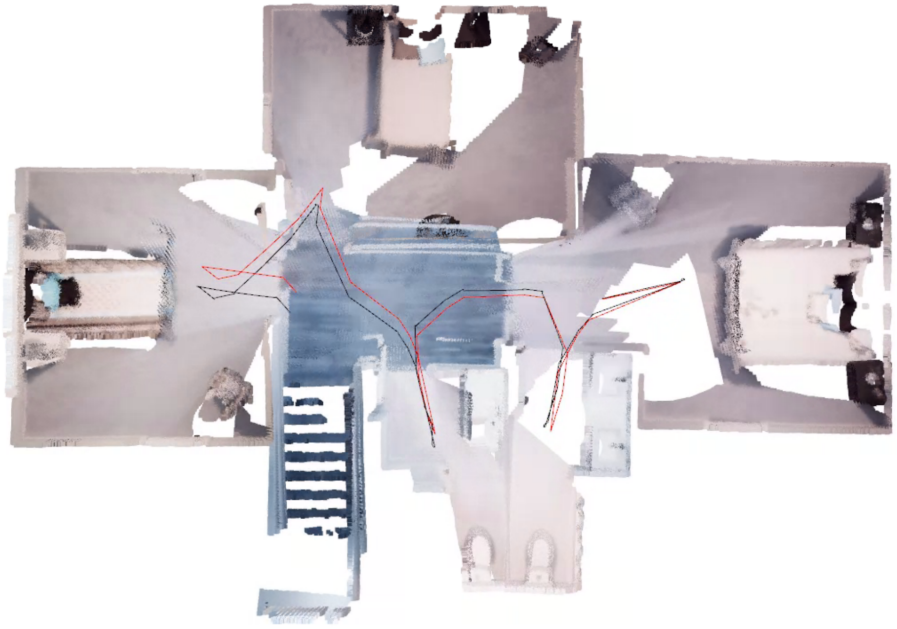
Data size = 152

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:**

studentID_Name_hw1
| --- load.py # **only for exploring the scene and save the data**
| --- reconstruct.py # **put your implementation code here**
| --- bev.py
| --- report.pdf
| --- README

README : Please describe how to run your program.
Please put all the files into folder **studentID_name_hw1** and compress your folder into **one "zip" file**. Submit it to New E3 System with the format **studentID_name_hw1.zip.**

2. The **deadline** for this homework is **10/11 23:55.**
3. Late submission leads to **-20 points per day**.
4. Online Demo: You will be required to reconstruct the scene on your local and show the result to TA during demo.
5. **Wrong submission format leads to -10 point**

6. Grading

1. Online Questions (explain BEV projection, depth unprojection, ICP and so on) 20%
2. Online Demo (3D reconstruction and pose overlay) 50%
    a. Present the BEV projection result on your computer.
    b. Show the reconstruction result on your computer.

3. Report of you implementation 30%

Task1:

i. Code

1. Detailed explanation of your implementation.
2. For example, how you do the projection

ii. Result and Discussion

1. Result of your projection (2 different pairs). Like the example result above.
2. Anything you want to discuss
3. Any reference you take

Task2:

i. Code

1. Detailed explanation of your implementation.
2. For example, how you implement ICP, depth_projection and other functions.

ii. Result and Discussion

1. Result of your reconstruction(Floor 1 and Floor2, Both open3d implementation and your own implementation)
2. Mean L2 distance between ground truth and estimated trajectory.
3. Anything you want to discuss, such as comparing the performance of two implementations.
4. Any reference you take