Perception and Decision Making in Intelligent Systems
Homework 1:
BEV projection and 3D Scene Reconstruction
Announce: 9/19, Deadline: 10/17 23:59

Introduction

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

Task1: BEV projection

a. **Data Collection:**
   Save a pair of front view and BEV image at the same time.
b. **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: 3D Reconstruction

a. **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 (GT_pose.npy).
b. **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**.
   (Open3D library provides a function to do it, and in addition, you need to implement your own version. There will be **two versions** of alignment. For further description, please refer to Task2.)
c. **Camera Pose Estimation and Visualization:**
   Visualize the trajectory and show the difference between the trajectory you estimate by the ICP algorithm and the ground truth trajectory.

Implementation

Task1: BEV projection

We gave two pairs of images (first and second floor) of front view and top view in ***bev_data***. The position and orientation of these two cameras are different.

You need to calculate the relative pose of two cameras to complete Task 1.

- **Camera1 (front): position (0, 1, 0) orientation (0, 0, 0)**
- **Camera2 (bev): position (0, 2.5, 0) orientation (-(np.pi/2), 0, 0)**

You can also modify load.py to collect your own pair data.

You can modify bev.py as you wish to finish your homework

1. Read a BEV image and top view image.
2. Use the mouse to click points on a BEV image.
3. Mark the area enclosed by these points
4. Project the marked area onto a perspective (front view) image
5. Save the projected image to the .png file

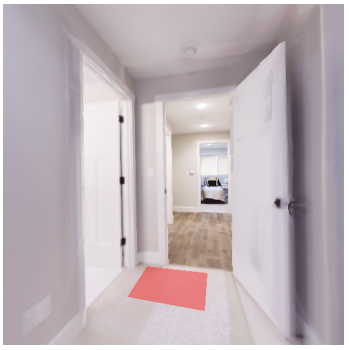
- **For intrinsic parameters, an agent camera utilizes a pinhole camera model (the resolution of an image is 512 x 512)**
- **Horizontal and vertical FOV are 90 degrees**
- **We have implemented a click_event() method in the bev.py**

Example results


bev1.png


front1.png


bev2.png


front2.png

Task2: 3D Reconstruction

a. Point Cloud Alignment and Reconstruction

We provide a work pipeline to do reconstruction:

1. Unproject depth images t_i and t_{i+1} to reconstruct two point clouds.
2. Do voxelization to your point cloud for reducing the number of points for less memory usage and speedup.
3. Apply global registration first which is used as the initialization of the local methods.
4. Apply local registration, using ICP to obtain the transformation matrix.
5. Align point cloud at t_{i+1} to point cloud at t_i, and apply the same process to the whole trajectory(Hint: matrix multiplication).

The code template provides some functions for you to reference. You can modify the template as you wish.

- **depth_image_to_point_cloud**: get point cloud from rgb and depth image
- **local_icp_algorithm:** Use Open3D library to implement it. i.e. point to point or point to plane
- **my_local_icp_algorithm:** Implement your own icp registration.

- **For intrinsic parameters, the agent camera uses a pinhole camera model (resolution 512 x 512)**
- **Horizontal and vertical FOV are 90 degrees**
- **depth_scale = 1000**

- Here is the reference about ICP algorithm 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

b. 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)

Example results

Red line: The estimated camera pose
Black line: The ground truth camera pose (GT_pose.npy)

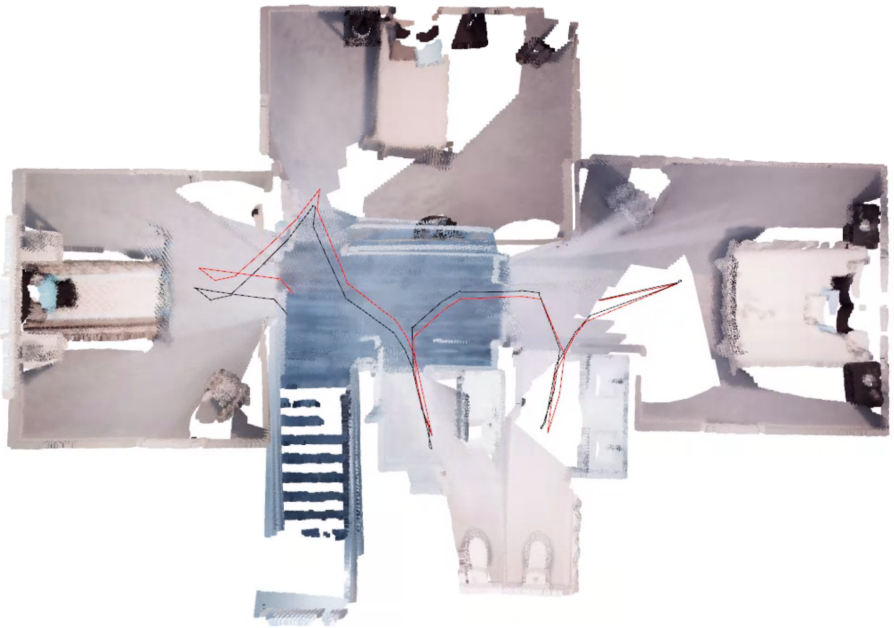
Note: the below results are the screenshots.

- **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



Grading

Online Demo 30%

1. Present the BEV projection result on your computer.
2. Show the reconstruction result on your computer.

Report 70%

Your report should include the following content:

1. **Implementation (40%)**

   Task1:

   a. Code
      Detailed explanation of your implementation. For example, how you do the projection
   b. Result and Discussion
      i. Result of your projection (2 different pairs). Like the example result above.
      ii. Anything you want to discuss
      iii. Any reference you take

   Task2:

   a. Code
      Detailed explanation of your implementation. For example, how you implement ICP, depth_projection and other functions.
   b. Result and Discussion
      i. Result of your reconstruction (Floor 1 and Floor2, Both open3d implementation and your own implementation)
      Make sure your execution time for each reconstruction less than 5 mins, or you will get 0 point in this part.
      ii. Mean L2 distance between ground truth and estimated trajectory.
      iii. Anything you want to discuss, such as comparing the performance of two implementations.
      iv. Any reference you take

2. **Questions (30%)**
    a. What's the meaning of extrinsic matrix and intrinsic matrix?
    b. Have you ever tried to do ICP alignment without global registration, i.e. RANSAC? How's the performance? Explain the reason. (Hint: The limitation of ICP alignment)
    c. Describe the tricks you apply to improve your ICP alignment.

Submission

Due Date: 2023/4/21 23:59

Please directory compress all your code files and report (.pdf) into {STUDENT ID}_hw1.zip and submit it to the New E3 System.

The file structure should look like:

```
📂{student_id}_hw1.zip
 ├ 📜 load.py          # only for exploring the scene and save the data
 ├ 📜 reconstruct.py   # put your implementation code here
 ├ 📜 bev.py
 ├ 📜 report.pdf
 └ 📜 README.md        #  describe how to run your program
```

Wrong submission format leads to -10 point

Late submission leads to -20 points per day