HW2: Sexy Semantic Mapping
RBE549
(Using 1 late day)

Karter Krueger
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: kkrueger2@wpi.edu

I. AutoCalibration

Nearly all self-driving cars use LiDAR for mapping and localization. However, when running SLAM or localization methods, there are some points in the LiDAR point-cloud that are less-desirable to consider in the localization, such as moving vehicles and people. To reduce localization noise, we ideally want to use fixed points such as trees and other landmarks along the sides of the road. This is one big motivation to fuse semantics with the point-cloud data in this homework. Another motivation is that self-driving cars must pay special attention to moving objects such as cars and people. But adding semantics to point-cloud data, we can spot these classes of objects and perform differently around them.

For background, ICP (Iterative Closest Point) is a method used for mapping with LiDAR points and for SLAM (Simultaneous Localization And Mapping) on LiDAR point cloud data to localize the self-driving car while it drives along the road, while also building a map of the surroundings. In this homework, we will use an existing ICP method to match the LiDAR points between frames to build a map.

For additional background, semantics can be determined using RGB images and a semantic-segmentation neural-network method. In this homework, we will use an existing pre-trained semantic segmentation neural-network to extract semantic class labels from RGB images. We will then write the math and functions to project the pixel classes from (u, v) space to the 3D XYZ space of the LiDAR. We can then find the nearest color/class for each LiDAR point and color it on a 3D map.

A. Step 1: ICP Mapping

In this homework, we use a standard ICP method [1], from this pip library [2]. This library can take in two 3D point clouds and find the correspondences between them with the nearest matches for each point, and the transformation between the two frames. We can then use these transformations to shift the point cloud from each frame and merge them all into one large map. Below is an example of a map that was displayed in the homework outline. (my code isn't at the point of running ICP yet)

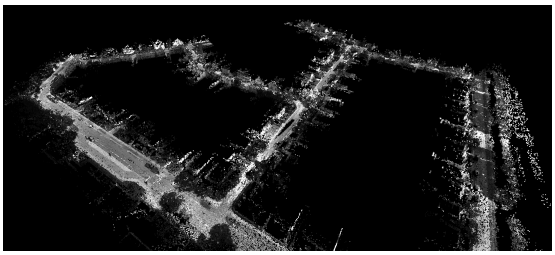

Fig. 1. Example map from the HW

B. Step 2: Semantic Segmentation

We use DeepLabV3 (already available in Pytorch) with weights that have been trained on the KITTI dataset that we are using for LiDAR and RGB data in this homework. For each RGB image, we pass the frame into the DeepLabV3 semantic segmentation neural-network model in pytorch and get a semantically-segmented image of the same size out. Each output frame has a class of highest likelyhood of object assigned to each pixel. We represent the classes with colors such as green, purple, blue, etc for classes such as tree, road, car, etc. An example of a semantic segmentation output 3 of an RGB image 2 is shown below.

Fig. 2. RGB Input Image from KITTI Dataset

C. Step 3: Project 2D Image Values to 3D LiDAR Points

We can project the semantic labels from the image space to their 3D ray space using the intrinsic camera matrix K. We can then use the extrinsic relation between the camera and LiDAR to shift the rays to be aligned with the LiDAR center. We can then take the rays and find the nearest ray to each

Fig. 3. Semantic Result of the RGB Image

LiDAR point. This semantically-labeled point cloud can then be visualized like the original point cloud.

D. Results

Note: the provided code does not run with the Kitti dataset due to challenges with downloading the large files and getting things to load properly. However, the code does demonstrate the full flow of how things should work if the data was working correctly.

REFERENCES

[1] https://en.wikipedia.org/wiki/Iterative_closest_point
[2] https://pypi.org/project/simpleicp/