Sexy Semantic Mapping RBE549 (Using 1 Late Day)

Tript Sharma
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: tsharma@wpi.edu

I. Introduction

Self driving cars often use LIDARs as a central depth sensing modality. But often during mapping or localization, higher level semantic information is desired. in this project I aim to build a map from raw LIDAR point cloud and then transfer the predicted semantic labels from the camera image onto the LIDAR point cloud. We will talk about these steps next.

However translating semantic info from RGB frames at certain timestamp to the fusedd point will require pose estimation between each LiDAR point cloud. It is easier to perform semantic segmentation on images and translate them to the corresponding LiDAR point cloud at that timestamp. Then we can fuse these point clouds easily using ICP.

Hence the approach I followed is as follows:

1) Parsing Dataseet
2) Semantic Segmentation of RGB Images
3) Point Painting
4) ICP

II. Dataset

In this project, I used the 'CITY' sub-dataset KITTI dataset [1]. I used the raw LIDAR point clouds (or scans), rectified RGB Images and sensor intrinsics and extrinsics.

A. Parsing Dataset

The dataseet contains LiDAR points in '.bin' files. I used Open3D to parse them and visualize the point clouds as shown in Figure 2 Example of RGB and Velodyne pointcloud are shown in Figures 1.

Fig. 1. Rectified RGB Image

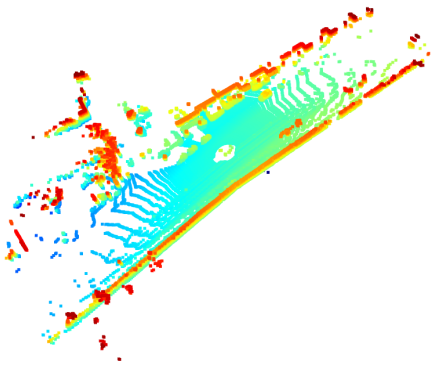

Fig. 2. velodyne point cloud

III. Semantic Segmentation

I used the FCN model with ResNet 101 backbone for semantic segmentation available pre-trained on MS-COCO model containing same classes as PASCAL VOC. The model is available in torchvision [2]. The architecture for the model has been presented in Figure 3

All the images were first read into the memory to perform semantic segmentation.

The model will give each pixel in the image a class label. Based on this class label, there is a colour associated with it. The output of this semantic segmentation can be seen in Figure 4

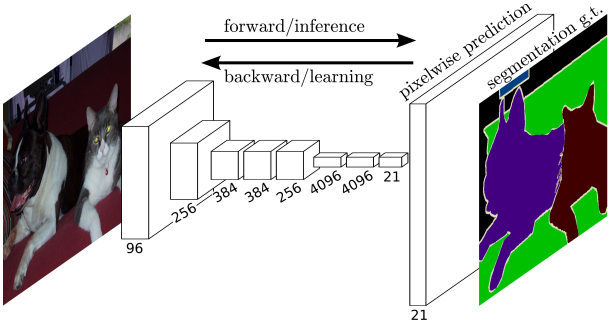

Fig. 3. FCN architecture for Semantic segmentation

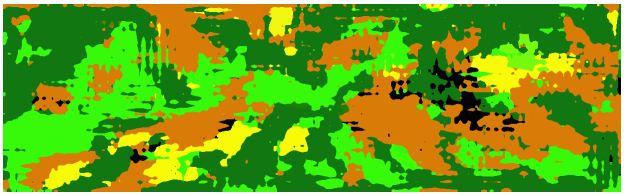

Fig. 4. Segmented image

IV. Point Painting

The goal is to project 3D points from the LiDAR to image plane. The projected 2D points can be semantically segmented using the semantic information obtained from the previous section. Thus, enabling us to 'paint' the points in the point cloud.

LiDAR and camera are physically at in different frames and have different intrinsic parameters. Thus the points cannot be projected simply by taking it's projection in the 2D space.

But since we know the geometry between these 2 sensors, we know the Rotation Matrix R from the Camera to the Lidar and the Translation vector T from the LiDAR to the Camera. Thus using Equation 1, I transformed the LiDAR 3D points into the frame of reference of the camera.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where P is the projection matrix of dimension $(3, 4)$

After projection the third element of each point gives us the projected depth in the image. Some points might have the depth value less tha zero and we neglect those points. The remaining points are homogenized and projected onto the image to obtain the corresponding semantic labels.

After obtaining the semantic labels we color the 3D points using the colors for the corresponding semantic class.

V. Iterative Closest Point

The painted point clouds are then merged using Iterative Closest Points algorithm.

ICP is a point cloud registration algorithm that merges two point clouds. It returns a transformation matrix. We can use this matrix to align the two point clouds. Dependent upon the initial transformation, it optimizes the transformation by iteratively reducing the distance between corresponding points or a point and plane or other associations depending upon the ICP method used.

I used Point-to-point ICP method who's objective function is to reduce the distance between corresponding points in two point clouds.

To find correspondences between these points clouds, we do a simple nearest neighbour approach to each point to obtain the correspondences.

We then find the best transformation matrix using Equation

$$E(\mathbf{T}) = \sum_{(p,q) \epsilon K)} ||p - \mathbf{T}q||^2 \tag{2}$$

where p is the point in target point cloud while q is the point in source point cloud. Furthermore, $\mathbf{T}$ defines the transformation we want to optimize while K defines the number of points pairs.

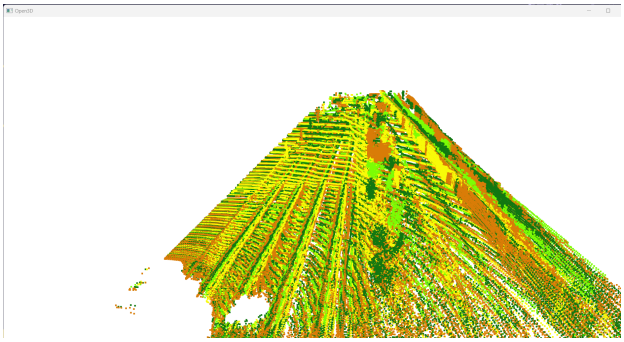

Fig. 5. P2P ICP Result

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," The Int. J. Robot. Res., vol. 32, no. 11, pp. 1231–1237, 2013.

[2] https://pytorch.org/vision/0.12/generated/torchvision.models.segmentation.fcn_resnet101.html