RBE549 Computer Vision : Homework-2

USING 1 LATE DAY

Ajith Kumar Jayamoorthy
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, U.S.A.
ajayamoorthy@wpi.edu

***Abstract*—In this paper, we implement two concepts. First, we use semantically segmented images to paint the corresponding point clouds. Second, we use the painted point clouds from various consecutive timestamps and combine them together using the Iterative Closest Point Algorithm. The results and observations of the above implementations have been recorded.**

I. Introduction

In case of self-driving cars, LiDARs are used to sense the depth objects in the environment in the form of point cloud, for the ego vehicle. However, during localization and mapping using LiDAR information, the static vehicles can end up being considered as building artifacts. Therefore, higher level of semantic information is required for LiDAR based SLAM. In addition to this, LiDAR don't have sufficient vertical resolution to detect objects and their labels. Moreover, the sparsity and the huge number of unorganized points in a 3D point cloud data cannot be used for semantic segmentation. To resolve this issues, experts in the field combined LiDAR and camera information to create semantic painted point clouds. [1]

II. DATA

The data we have used is the part of the KITTI-360 data set [2]. It consist of 31 consecutive scans of the LiDAR point cloud and the corresponding 2D images. We also have calibration information of all the cameras and the velodyne LiDAR. An example of the image and the corresponding 3D point cloud data are shown by figures 1 and 2, respectively.

Fig. 1. The original rgb image part of the sampel the data.

III. METHODOLOGY

The painting point cloud method has been implemented in the following steps:

- Image semantic segmentation
- Semantic mapping of point clouds
- Building entire Map

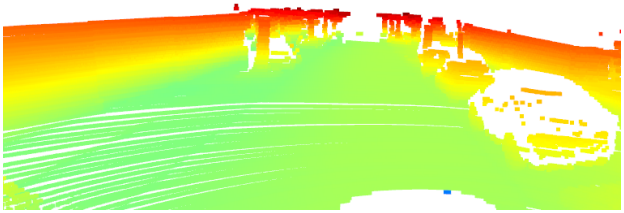

Fig. 2. The 3d point cloud data corresponding to figure 1 in the image perspective.

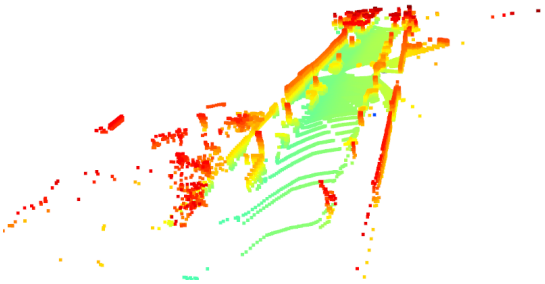

Fig. 3. The full 3d point cloud data corresponding to figure 1.

A. *Image Semantic Segmentation*

Semantic segmentation is an image analysis procedure in which we classify each pixel in the image into a class. In simple terms, segmentation helps detect objects in a image and label them accordingly. In autonomous driving, the computer driving the car needs to have a good understanding of the road scene in front of it. It is important to segment out objects such as cars, pedestrians, lanes and traffic signs [3].

We have used pre-trained **Pyramid Scene Parsing ResNet101** model to implement semantic segmentation on the source rgb images in our data. The model we used have weights trained on the **cityscapes** dataset. The segmented output images are stored in a separate folder to be used for painting the point clouds. The following figure 4 shows the actual rgb image on the top and the corresponding segmented output in the bottom.

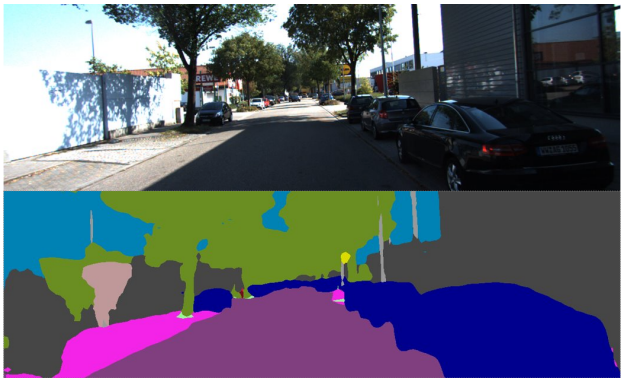

Fig. 4. RGB image (TOP) and corresponding segmented output(BOTTOM).

B. *Semantic mapping of point clouds*

In this step, we are going to project the points in the point cloud to the image plane and based on the segmented values in the image, we are going to paint the points and then project the point cloud back into 3D space. This process is repeated for all the 31 images and their corresponding point clouds. The following are the steps in the semantic mapping process.

1) ***Project of Point Clouds to Image****:* The calibration file contains the values of 7 matrices:

- P0 - projection matrix for a point in the rectified referenced camera coordinate to the camera0 image
- P1 - projection matrix for a point in the rectified referenced camera coordinate to the camera1 image
- P2 - projection matrix for a point in the rectified referenced camera coordinate to the camera2 image
- P3 - projection matrix for a point in the rectified referenced camera coordinate to the camera3 image
- R0_rect - rectifying rotation for reference coordinate
- Tr_velo_to_cam - maps a point in point cloud coordinate to reference co-ordinate

The the corresponding (x,y) image co-ordinate of the point cloud from the velodyne frame can be calculated as follows [5]:

$$y\_image = P2 \cdot R0\_rect \cdot Tr\_velo\_to\_cam \cdot x\_velo\_coord$$

2) ***Painting the point clouds****:* The colours in the semantically segmented images are scaled down from 0-255 to 0-1. After this step, we then create a list of all the colour-labels for each pixel in the image. Using the open3d library, we will map each point with the corresponding colour values from the segmented pixel and stored in to a new variable. This new variable has information of the point cloud data along with the colour code. After this the semantically mappe point cloud data is stored in a separate file. The figure 5 shows the painting of point cloud image using RGB images in the camera perspective. The figure 6 shows the entire rgb image painted point cloud for verification.

Later, the semantic images were used for painting the point clouds and the results has been shown in figure 7 and 8.

Fig. 5. Point cloud data painted using the source camera RGB images (In camera perspective).

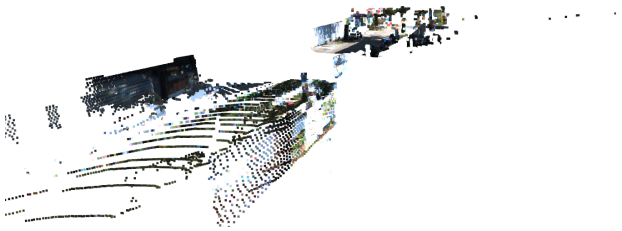

Fig. 6. Point cloud data painted using the source camera RGB images.

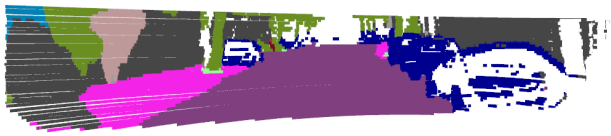

Fig. 7. Point cloud data painted using the source segmented images (In camera perspective).

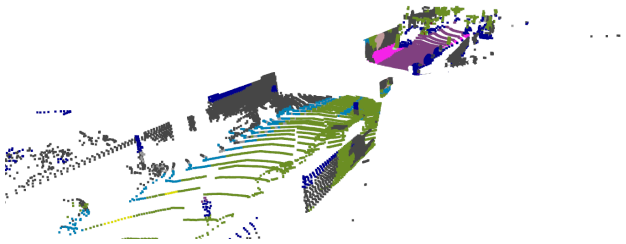

Fig. 8. Point cloud data painted using the source segmented images.

C. *Building entire Map*

In order to build the entire map using the segmented point clouds we use the point-to-point Iterative Closest Point Algorithm in the Open3D library. The objective of the ICP algorithm is to do Point cloud alignment. Thus, the algorithm estimates the transformation to move the first point cloud data to the second point cloud data to be aligned with each other.

Iterative Closest Point Algorithm

The point cloud alignment is performed in two steps by the ICP algorithm:

- Data Association
- Estimating Transformation

1) ***Data Association****:* Let us assume there are two consecutive points clouds, namely, PCD-I and PCD-II. In this step, for a given point in PCD-I, we calculate the Nearest Neighbour from the PCD-II point cloud. Similarly, for the other points in

the PCD-I point cloud we calculate the corresponding Nearest Neighbours in the PCD-II point cloud data.

*2) **Estimating Transformation**:* In this step, we are trying to minimize the distance between the point cloud pairs and try to estimate the transformation from PCD-I to PCD-II. There are two steps to achieve this process:

- First, we compute the centre of mass the point clouds PCD-I and PCD-II. Then we evaluate the translation vector required to move the centre of mass on top of each other.
- Second, we compute the optimal rotation between the two super-imposed point clouds to match the corresponding point pairs. This process is optimized using SVD method.

We have to recompute the data association and transformation after each iteration to achieve the most optimal solution. To fine-tune the point-to-point ICP for better results, the transformation matrix is initialized as

$$T = \begin{bmatrix} 1 & 0 & 0 & point\_cloud\_index \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, point_cloud_index is the i^{th} point cloud to register. The idea behind using point cloud index is that, the point clouds are captured when the vehicle is in motion, and there will be a displacement (along X-axis in this case) in the corresponding point of two point clouds. So, initializing the transformation matrix by point cloud index will first translate the i^{th} point cloud by '**i**' distance. This brings the source point cloud near the target point cloud. Then, on performing ICP we get better results.

IV. RESULTS

The result of the ICP algorithm has been as shown in the following figure 9.

Fig. 9. Map built using point-to-point ICP algorithm on semantic painted point clouds.

It can be observed from the above output that the right part of map is properly segmented, whereas the left side of the map is not. This is because the images used to paint the point clouds are from the from view of the ego vehicle. To improve the accuracy of the segmentation, the scene must be captured in both directions and it should be used when segmenting the point cloud. Furthermore, from the above map we can observe that the result of the ICP algorithm is decent but not that efficient and there are others methods to better generate the map using painted point clouds.

REFERENCES

[1] https://rbe549.github.io/fall2022/hw/hw2/
[2] https://github.com/HengLan/Visualize-KITTI-Objects-in-Videos/tree/main/data/KITTI
[3] https://learnopencv.com/pytorch-for-beginners-semantic-segmentation-using-torchvision/
[4] https://cv.gluon.ai/build/examples_segmentation/demo_psp.html
[5] https://medium.com/test-ttile/kitti-3d-object-detection-dataset-d78a762b5a4
[6] https://www.youtube.com/watch?v=QWDM4cFdKrE