RBE-549: Homework 2- Sexy Semantic Mapping

Chinmay Kate
M.S. Robotics Engineering
Worcester Polytechnic Institute (WPI)
Worcester, MA 01609
Email: cskate@wpi.edu

I. Overview

Point-Painting is the method used to fuse the semantic segmentation results based on RBG images to raw Lidar point cloud. Point-Painting architecture consist of two main stages. First step is to build the Map using Doppler ICP. This is Lidar based pointcloud mapping. Secondly, We use RGB images to segment semantic information and transfer it to point cloud map using extrinsics. Thus we use extrinsic between Lidar and cameras (Here we use left camera of stereo) to transfer the semantics data to lidar ICP.

II. Dataset

We use KITTI-360 as our dataset which contains rich sensory information, full annotations and information of road scenes. This is recording of suburbs in Karlsruhe, Germany. We used Perspective 2D images from the dataset to Train and test for our semantic segmentation. Raw Velodyne scans data is used to map point clouds using Doppler ICP. Here we also use Camera calibration and pose data like Intrinsic and extrinsic information. Each sensors and relative poses information is available at Kitti-360 dataset officaial page.

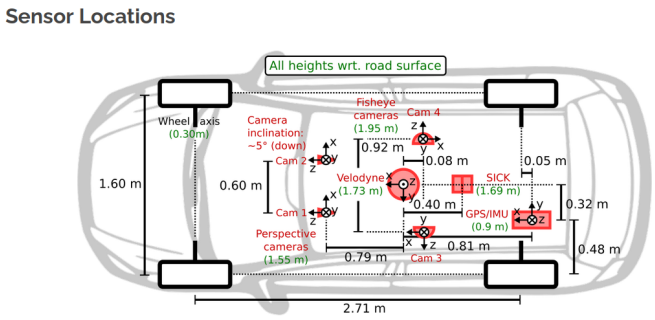

Fig. 1: Sensors Information for KITII360 Dataset.

III. Doppler-ICP

Doppler ICP is a novel algorithm for point cloud registration for Lidar sensor. It exploits the compatibility of each point's Doppler measurement and the sensor's current motion estimate. Joint optimization is done for Dopplers velocity objective function and the geometric objective function.
It fixes the point cloud alignment problem where there is less no of features in the environment. In this algorithm we prune the points of the dynamic target, which gives more robust ICP solution.
These FMCW Lidars are able to measure the relative velocity between each measured point to the sensor along the radial direction (Doppler velocity), Thus providing additional motion information about the target.
Following are the highlights of Doppler ICP Algorithm.

- First, relationship is established between the Doppler velocity of the observed point and the transformation to be estimated.
- A Doppler velocity residual is then defined and included in overall objective function, that needs to be minimized along with the geometric objective function.
- Defining and including Doppler error distance metrics and their gradients in the objective term. Including gradients makes it independent of the environment which make optimization easy task.
- We optimize the cost functions iteratively with great convergence.
- Correspondence matches used in optimization step are improved by exploiting the compatibility of each point's Doppler measurement and current sensor's rigid motion estimate. This improves overall algorithm's performance by eliminating most moving objects in a scene which generally degrade ICP performance.

IV. Semantic Segmentation using RGB Images

Semantic segmentation is performed using Deep Neural Networks. Here **DeepLabV3-Plus-Mobilenet** is used which is encoder decoder architecture. This is used to perform multiclass semantic segmentation. This network is already trained on Cityscapes and VOC datasets. We can download the weights and fine tune on the KITII-360 dataset, which gives very sharp results. This can be done by training after 6th layer with KITII-360 dataset. Predicted results are shown in figure.

V. Transfer sematics to Point cloud

Now, utilizing semantic segmantation information we transfer the semantic color labels to the generated map. You need to use the extrinsics between the sensors to transfer the labels from RGB image to LIDAR point cloud.

Algorithm 1: Doppler Iterative Closest Point

Input:

$\mathcal{P}$ Source point cloud containing Doppler velocity measurements

$\mathcal{Q}$ Target point cloud (optionally containing normals)

Δ_d Maximum correspondence distance

Δ_v Maximum velocity error

$\mathbf{u}_0$ Initial state vector

Output:

$\mathbf{u}$ Transform that aligns $\mathcal{P}$ with $\mathcal{Q}$

$\mathbf{u} \leftarrow \mathbf{u}_0$
while *not converged* **do**
  $\mathcal{P}' \leftarrow \varnothing$
  $\mathcal{Q}' \leftarrow \varnothing$
  for $j \leftarrow 1$ **to** $|\mathcal{P}|$ **do**
    $p'_j \leftarrow \texttt{TransformSourcePoint}(\mathbf{u}, p_j)$
    $q_j \leftarrow \texttt{FindClosestTargetPoint}(\mathcal{Q}, p'_j)$
    if $|r_{v_j}| < \Delta_v \wedge \texttt{Dist}(p'_j, q_j) < \Delta_d$ **then**
      $\mathcal{P}' \leftarrow \mathcal{P}' \cup p_j$
      $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup q_j$
    end
  end
  $\mathbf{u} \leftarrow \arg\min_{\mathbf{u}} E(\mathbf{u}, \mathcal{P}', \mathcal{Q}')$
end

Fig. 2: Doppler ICP Algorithm

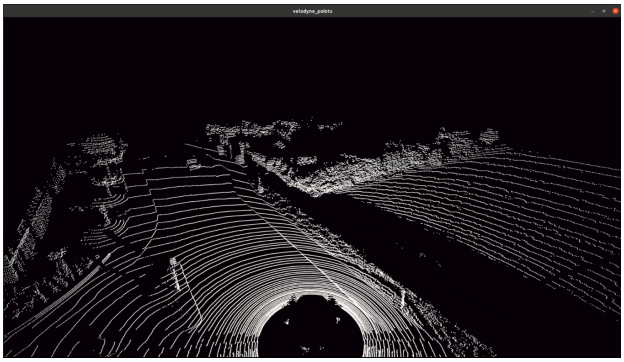

Fig. 3: Raw point cloud map Doppler ICP result's

Fig. 4: RGB image

Fig. 5: Semantic Segmentation

With the given information of poses and rotation between the sensors. We define Transformation matrix/extrinsic matrix. We can use this information to map the points onto the pixels.

Given data from Calibration and Pose in KITII360 dataset :

$$T_C^L = \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

r_{ij} = Rectified Rotation from left stereo camera.

t = Pose of the Lidar.

The transformation from LiDAR to camera will be given by:

$$T_l^c = \begin{bmatrix} R^{-1} & -R^{-1}t \\ 0 & 1 \end{bmatrix}$$

And the Perspective transform from LiDAR to Camera frame (pixel co-ordinates) will be given by:

$$P_l^c = P.R_{rect}.T_l^c$$

Using the above equation we can transform the 3d points in LiDAR frame to camera image plane directly.

Point painting

In Point painting we have camera frame which has smaller FOV angle compare to LIDAR angle. We take away the points that are not required or outside of the Camera frame, making computational more easy.

We find the projection of all points as points of sparse compared to each pixels . Projection of the LiDAR point cloud in image plane is done, Then correspondences is computed of each class label of the projected pixel. We finally color code the points with the pixel colors and to do so there are different iterative ways known one is K-means. We repeat this for all such 3d LiDAR points available.

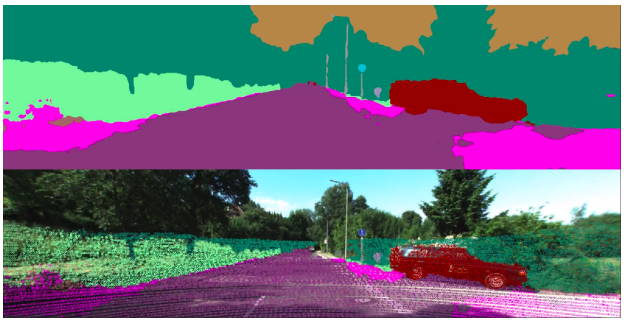

Fig. 6: Example 1: Painted Point cloud.

VI. References

- https://github.com/aevainc/Doppler-ICP
- https://github.com/VainF/DeepLabV3Plus-Pytorch
- https://www.cvlibs.net/datasets/kitti-360/documentation.php

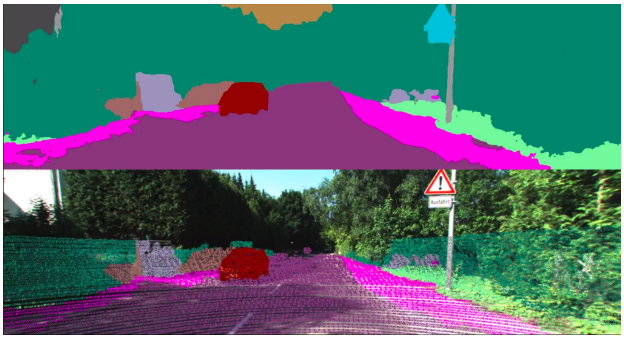

Fig. 7: Example 2: Painted Point cloud.

- https://github.com/naitri/PointPainting
- https://github.com/hanzheteng/color_icp