Assignment 2

Team 18: French Biriyani
Sudhansh Yelishetty (2018102029)
Dipanwita Guhathakurta (2018112004)

September 16, 2020

1 Assignment 2.1 - Point Cloud Registration

1.1 Objective

The task for this part of the Assignment is to register 77 point clouds given the global poses for each of them. You have to rotate and translate each point cloud according to the pose given, append them and finally plot them.

1.2 Approach

Since the poses in the file 01.txt are from the ith camera(and not LiDAR) frame(at the ith timestamp) to the 0th camera frame, and the points in the given LiDAR bin files are observed in the LiDAR frame, it is first necessary to apply the transformation on the points so that we go from the LiDAR frame into the ith camera frame. This transformation brings us to the ith camera frame from the LiDAR frame and aid us in the next part of the assignment as well. Assume distance between the origins of the LiDAR and the camera frame to be 0. The transformation matrix for the same is:

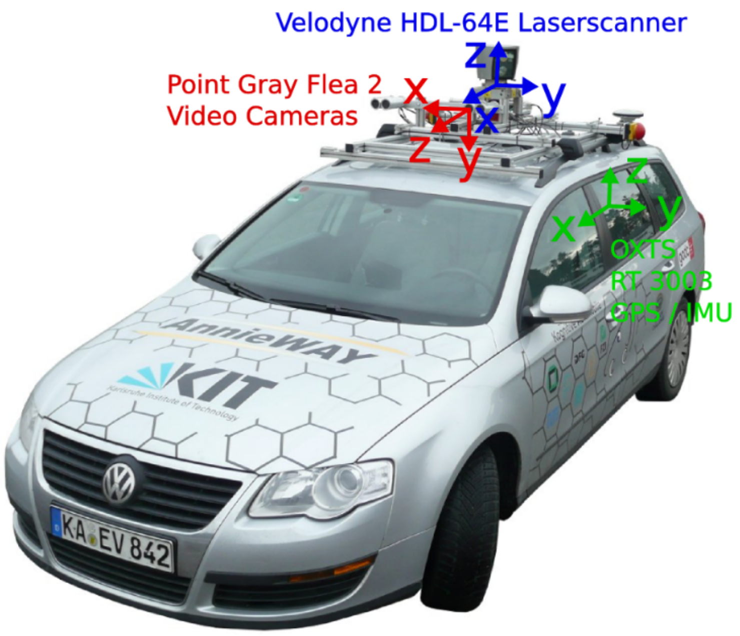

Figure 1: Frames Orientations

$$Q^{cam_i} = T^{cam_i}_{LiDAR} Q^{LiDAR} \tag{1}$$

$$T^{cam_i}_{LiDAR} = \begin{bmatrix} R^{cam_i}_{LiDAR} & t^{cam_i}_{LiDAR} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2}$$

$$R^{cam_i}_{LiDAR} = \begin{bmatrix} \hat{x}_{LiDAR}\hat{x}_{cam_i} & \hat{y}_{LiDAR}\hat{x}_{cam_i} & \hat{z}_{LiDAR}\hat{x}_{cam_i} \\ \hat{x}_{LiDAR}\hat{y}_{cam_i} & \hat{y}_{LiDAR}\hat{y}_{cam_i} & \hat{z}_{LiDAR}\hat{y}_{cam_i} \\ \hat{x}_{LiDAR}\hat{z}_{cam_i} & \hat{y}_{LiDAR}\hat{z}_{cam_i} & \hat{z}_{LiDAR}\hat{z}_{cam_i} \end{bmatrix} \tag{3}$$

$$R^{cam_i}_{LiDAR} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{4}$$

$$t^{cam_i}_{LiDAR} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

$$\therefore T^{cam_i}_{LiDAR} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Since the above transformation matrix is a 4x4 matrix, we convert each LiDAR point coordinates to homogenous coordinates, i.e. from 3x1 vectors to 4x1 vectors as follows:

$$Q_{3\times1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \implies Q_{hc} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{7}$$

Next, we used the pose matrix details given in the file 01.txt, reshaped them to form matrices of the following type:

$$T^{cam_0}_{cam_i} = [R|t]_{3\times4} \tag{8}$$

Since our points are in 3D space (Q), our transformed point cloud (P) will look like this:

$$P = RQ + t \tag{9}$$

Instead of having to perform matrix multiplication and vector addition at every iteration in the process, we vectorized the transformation and the pose matrix from camera_i frame to camera_0 was changed to look like this homogeneous SE(3) representation by appending the vector $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ at the end of each pose matrix. These computations are handled by our function $computeTransformation()$ in A21.ipynb:

$$T^{cam_0}_{cam_i} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix}_{4\times4} \tag{10}$$

Coming to the transformation between the 0th camera frame and the world frame, we follow a similar approach as while finding the transformation between LiDAR and camera_i frame.

$$T_{cam_0}^{world} = \begin{bmatrix} R_{cam_0}^{world} & t_{cam_0}^{world} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{11}$$

$$R_{cam_0}^{world} = \begin{bmatrix} \hat{x}_{cam_0}\hat{x}_{world} & \hat{y}_{cam_0}\hat{x}_{world} & \hat{z}_{cam_0}\hat{x}_{world} \\ \hat{x}_{cam_0}\hat{y}_{world} & \hat{y}_{cam_0}\hat{y}_{world} & \hat{z}_{cam_0}\hat{y}_{world} \\ \hat{x}_{cam_0}\hat{z}_{world} & \hat{y}_{cam_0}\hat{z}_{world} & \hat{z}_{cam_0}\hat{z}_{world} \end{bmatrix} \tag{12}$$

$$R_{cam_0}^{world} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{13}$$

$$t_{cam_0}^{world} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

$$\therefore T_{cam_0}^{world} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Finally let us consider the overall transformation. The overall transformation is computed in the function $computePoseCameraFrame()$ in A21.ipynb:

$$T_{LiDAR}^{world} = T_{cam_0}^{world} T_{cam_i}^{cam_0} T_{LiDAR}^{cam_i} \tag{16}$$

Because of this, our new point cloud representation looks like this:

$$P_{world} = T_{LiDAR}^{world} Q_{LiDAR} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \implies P = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{17}$$

1.3 Results

The resultant net point cloud from 77 LiDAR bins is as follows:

We export the notebook *A21.ipynb* into a Python script A21.py for ease of access of its functions in the later half of the assignment.

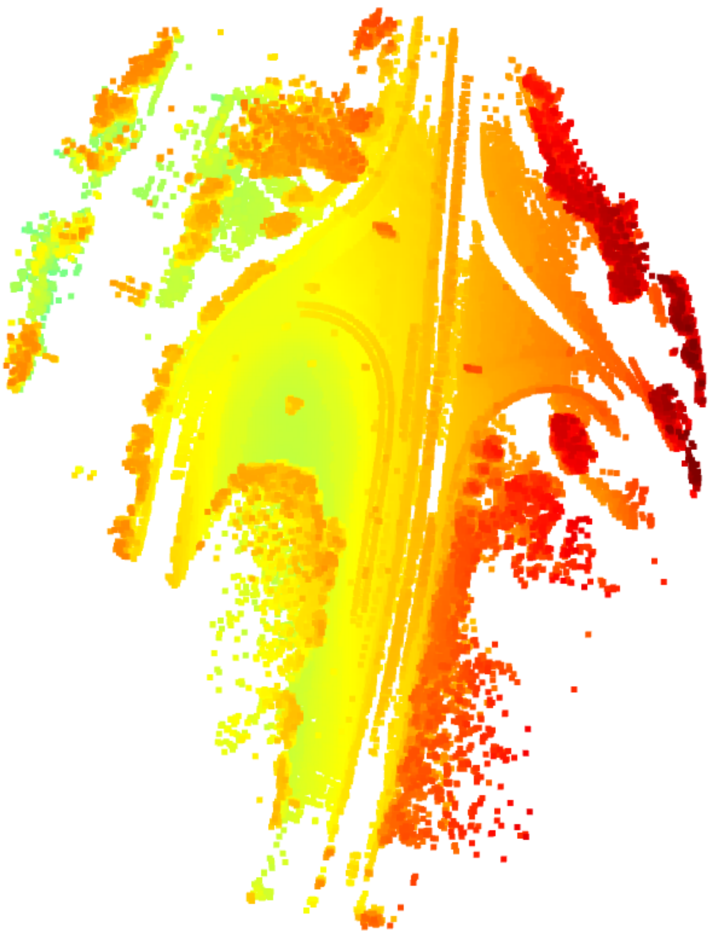

Figure 2: Net point cloud over all 77 bins

2 Assignment 2.2 - Occupancy Grid Construction

2.1 Objective

1. Create an occupancy grid map for each LiDAR scan. For the purpose of this assignment, you do not need to apply bayesian update rules and each individual scan can be assumed as a prior. Save each scan as a binary png.

2. Using odometry data, concat multiple scans and create a occupancy grid of 5, 10 and 15 scans

2.2 Approach

Occupancy grid maps are probabilistic in nature due to noisy measurements. Each cell can have three states: Occupied, unoccupied, and unknown. For the purpose of this assignment, we have ignored the unknown and worked in a binary setting where 1 is occupied and 0 is unoccupied.

The trick we employ in this part of the assignment is that we can mark a cell as occupied based on a threshold of how many different y values are there for a particular (x,y) cell. For each LiDAR bin file, we compute the following to generate the occupancy grid:

- Read *.bin using functions from 2.1, store the point cloud and round off the scan's x,y,z readings for discretization using np.round().
- Convert the scan data into a Pandas data frame, dropping duplicate rows and grouping by x,y to create all pairs of (x,y).
- Count number of distinct z readings for every (x,y) pair.
- Compute the statistics(mean, standard dev) for the relevant columns: the column containing the count of distinct z to help in deciding the threshold for the occupancy grid generation. In Pandas this is done easily using *df.describe*().
- Create a 400*400 numpy array of zeros to denote our occupancy grid image and mark points (basically the (x,y) pairs) where the count of z exceeds a certain threshold (in this case>1)
- Display the grid using matplotlib *imshow*(), scale it to (700,700) size and save it using OpenCV *imwrite*().

For part b) we concatenate the first 5,10,15 point clouds from the LiDAR bins using the *computePoseCameraFrame* from A21. We subject the net point cloud to the same process as described above to obtain the overall occupancy grids. Note: We have used a higher threshold in case of the combined point clouds (mark white if count>2).

2.3 Organisation of the notebook - Important functions

The first part of the notebook A22.ipynb guides step-by-step how the occupancy grid is generated for a given point cloud. Please refer to the notebook for detailed explanation of each cell of code.
The notebook contains two standalone functions: *computeGridindividualpcd* and *computeGrid*.
computeGridindividualpcd: Function to compute occupancy grid for an individual pcd.
  Input: pcd (an individual point cloud (or a netpcd resulting from addition of multiple pcds)
  Output: An 400*400 array representing the image of the occupancy grid
  Usage: img = computeGridindividualpcd(individualpcd)
computeGrid: Function to compute occupancy grid of a LiDAR bin file indicated by a certain index
  Input: the index of the LiDAR bin file to read and process for creation of occupancy grid
  Output: An 400*400 array representing the image of the occupancy grid
  Usage: img = computeGrid(binindex)

2.4 Outputs and Results

The individual occupancy grids for the LiDAR bin files are stored in assignment-2-18_french-biriyani/data/imgs/cv. The individual occupancy grids have also been uploaded to OneDrive at the following link: Link to individual occupancy grids

For link to the occupancy grids for 5,10 and 15 bins, refer to the following:

- For 5 bin files
- For 10 bin files
- For 15 bin files

2.4.1 Output format:

In the directory assignment-2-18_french-biriyani/data/imgs/cv or in the above OneDrive link you will find images of generated occupancy grids numbered as grid00.png, grid01.png till grid76.png. They correspond to each of the individual 77 LiDAR bin files that can be found at the location mentioned above, namely 000000.bin, 000001.bin respectively and so on.

2.4.2 Sample output:

For 2.2a)

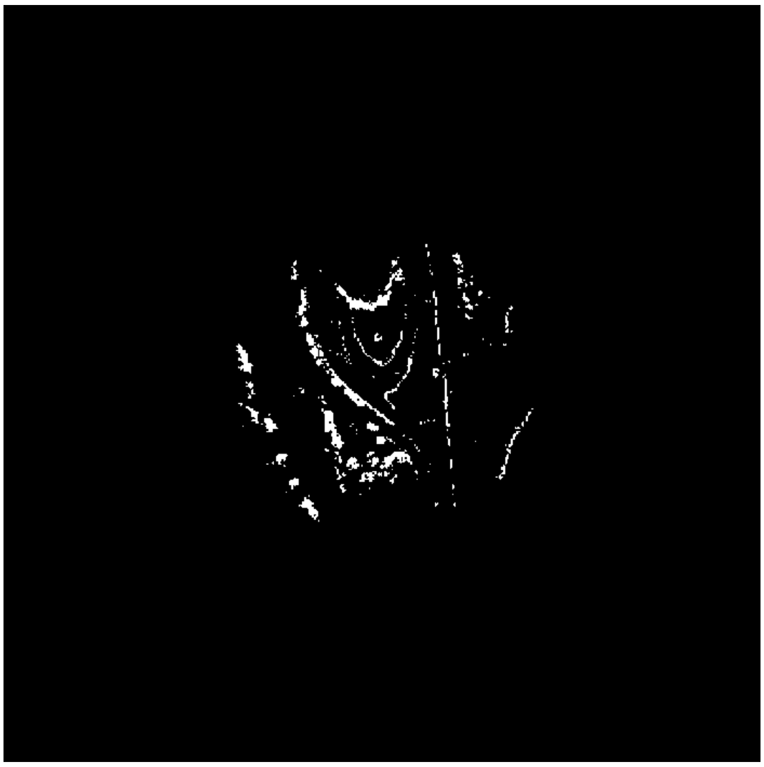

Figure 3: Occupancy Grid for 0000000.bin

For 2.2b) The occupancy grids for the first 5, 10, and 15 bins are as shown in the next page. Note that a higher threshold has been used to generate the point clouds for the combined point clouds of 5,10,15 bins.

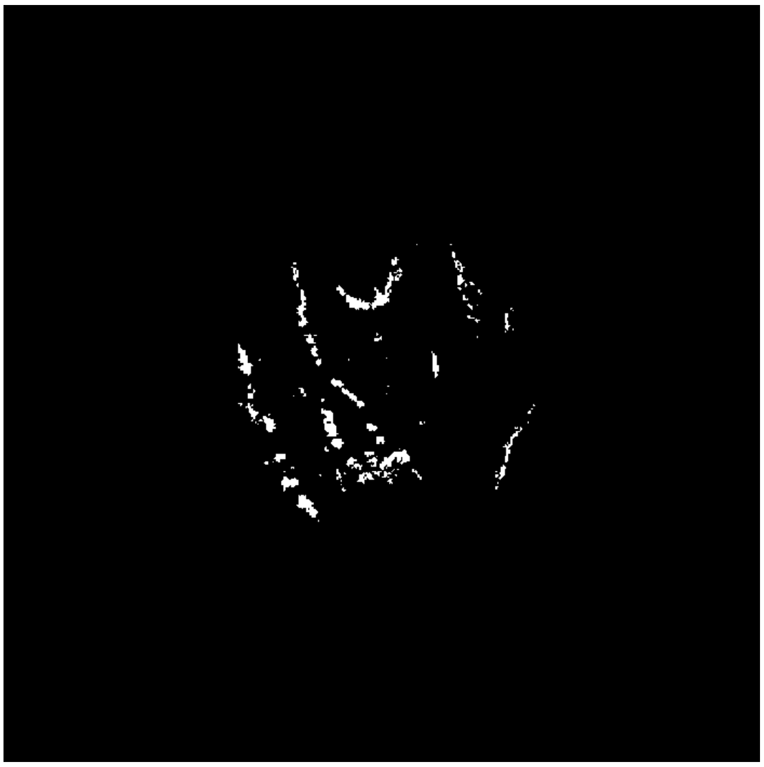

Figure 4: Occupancy Grid for the first 5 combined bins

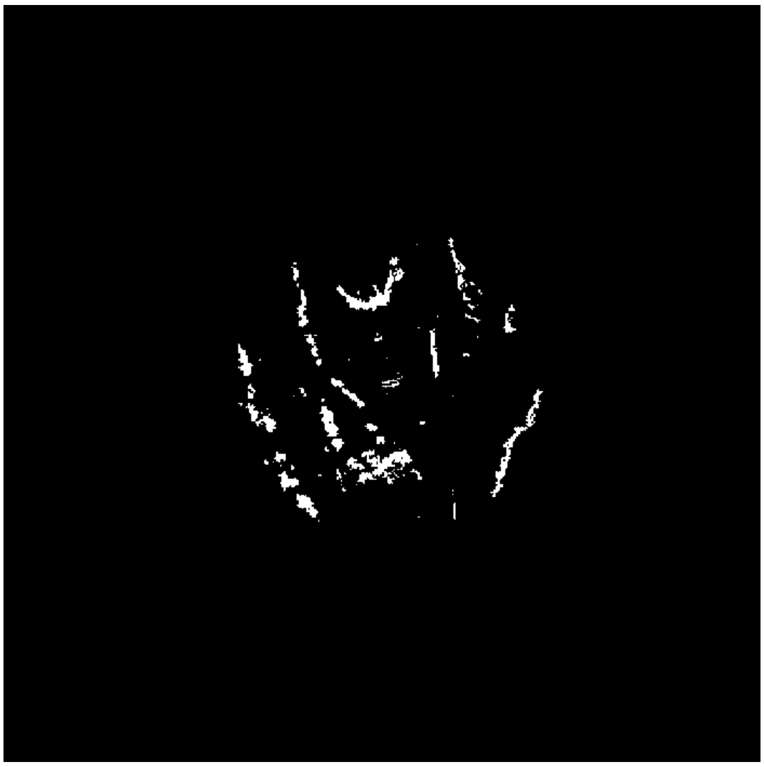
Figure 5: Occupancy Grid for the first 10 combined bins

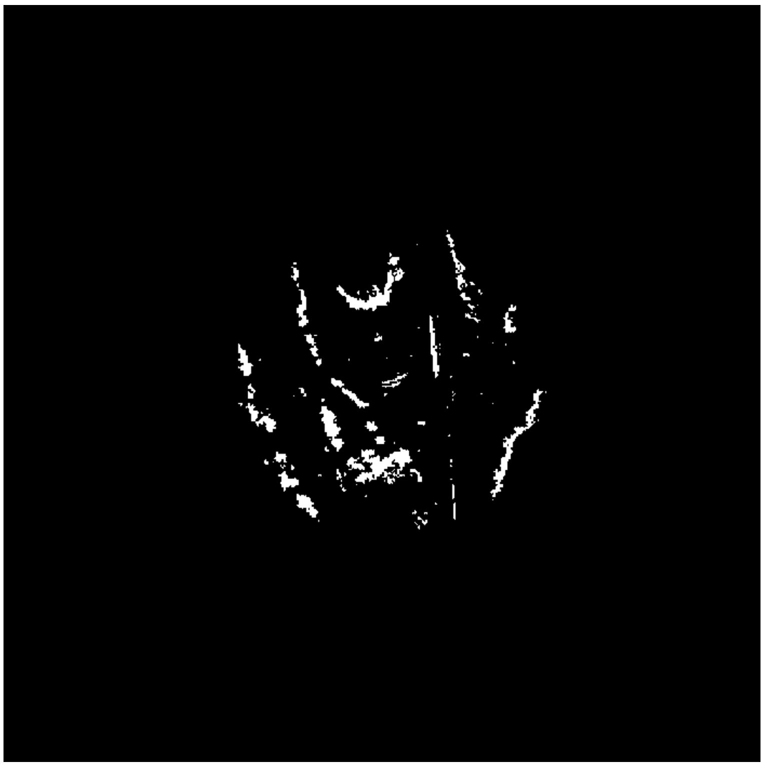
Figure 6: Occupancy Grid for the first 15 combined bins

2.4.3 Result:

We observe that the white patches in the generated occupancy grids become more distinct and prominent as we incorporate a greater no. of LiDAR bins. An analogy of this to the probabilistic Bayesian update would be that the confidence of a grid cell being occupied increases as we consider more LiDAR scans.

3 Individual Contributions

Sudhansh: Setting up transformations in homogeneous system, data restructuring, producing binary images from point clouds, report
Dipanwita: Pose computations, plotting, occupancy grid generation (in 2.2) and concatenation of point clouds (in 2.1), report for 2.2