Report

Team ID - 28
Team Name - lilly
Raja Vechalapu - 2018102032
Mohit Gadamsetty - 2018102016

Part I - Point Cloud Registration : (Done by Mohit)

Two Rotation Matrices are used to convert the Lidar scans into world frame.

1. Lidar to Camera
2. Camera to World

1. Lidar to Camera :

- The rotation angles used are zyx = [90 -90 0] , resulting in the following Rotation matrix.

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

- The transformation matrix is achieved by appending a column of 0's to the rotation matrix as there is no translation.

$$T_{L\_to\_C} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

2. Camera to World :

- As per the given poses in the file 01.txt , rotation matrices are generated respectively , given to take the values in row major.

  Eg : let line 1 in the txt be : $r_1\ r_2\ r_3\ r_4\ r_5\ r_6\ r_7\ r_8\ r_9\ r_{10}\ r_{11}\ r_{12}$ then

$$T_{C\_to\_W} = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_5 & r_6 & r_7 & r_8 \\ r_9 & r_{10} & r_{11} & r_{12} \end{bmatrix}$$

Then , the input from bin files is read as numpy arrays of size m*4 ,

Step -1 :

Result_1 $_{(3*N)}$ = $T_{L\_to\_C}$ $_{(3*4)}$ * Array^{T} $_{(4*N)}$, dimensions are mentioned

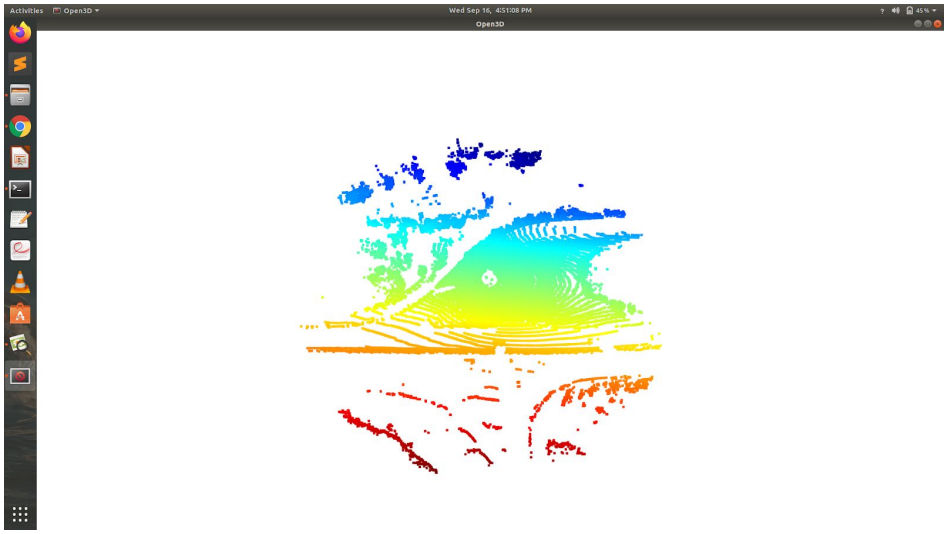
Step -2 :

A row of 1's are appended to Result_1 , since the last column in the transformation matrix corresponds to translation which is to be added. Let it be Result_2 $_{(4*N)}$

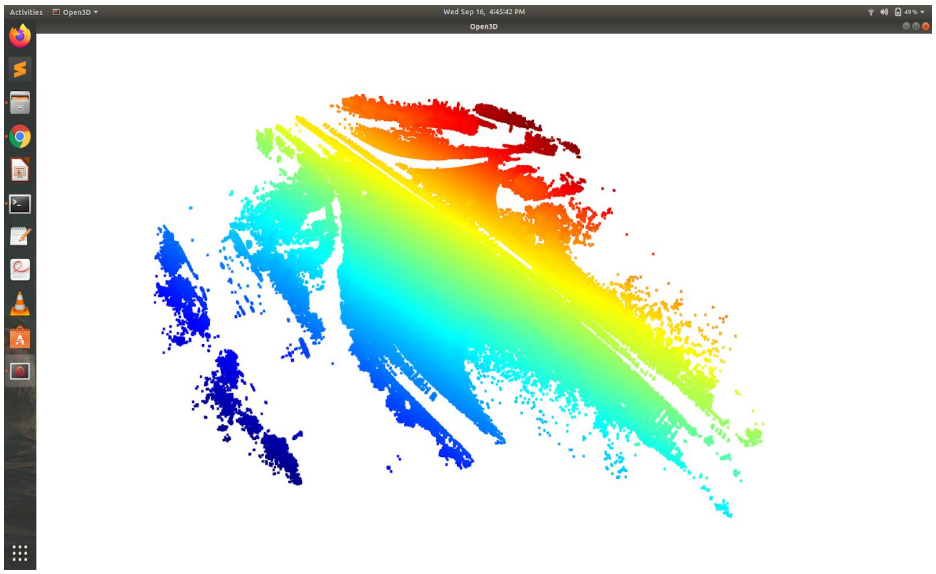
Step -3 : Final = $T_{C\_to\_W}$ * Result_2

The final array is converted into point cloud by open3d **"Vector3dVector()"** All the pcds are successively added and the final pcd is downsampled by voxel downsample for better display

Single bin pcd (1st bin) :-



All bins pcd :-



Part II - Occupancy Grid Mapping :- (Done by Raja)

Step -1 :-

A Grid with dynamic size is initialised as Range of the columns (1st*3rd).

Step -2 :-

The “Final” array obtained from step-3 above is rounded off and the Duplicate rows are removed. Using **numpy.unique()**

Step -3 :-

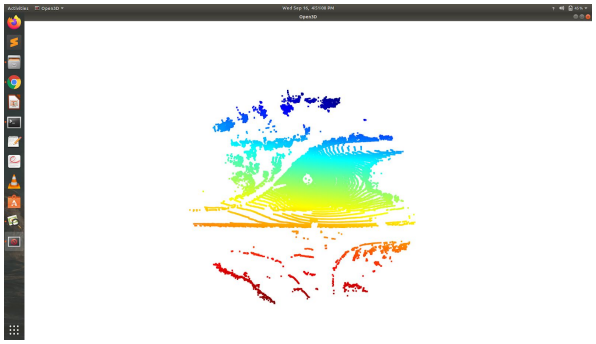
The result of the above contains the unique rows, from this result array, the count of all points (x_i , z_i) is calculated. From this the grid is filled ‘1’ (occupied) if the count of the point is greater than threshold. Else it’s filled ‘0’ (unoccupied).

Step -4 :-

The Grid then is saved as image using **matplotlib.image.imsave()**

Bin - 1 (threshold = 2) :

Pcd



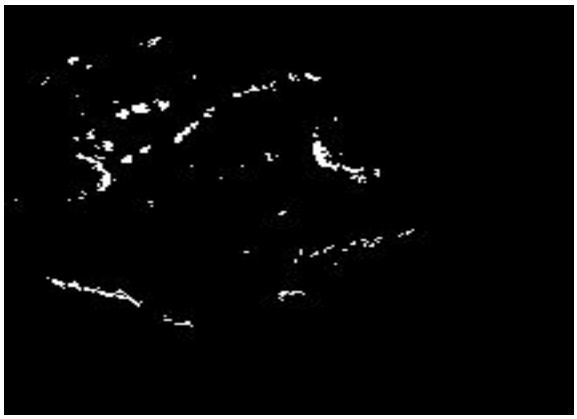
Occupancy grid

For the second case (5 , 10 , 15 scans concatenated):

1. From step -3 of Part I , the final arrays for each of the scans will the calculated and they are concatenated using **numpy.concatenate()**
2. After the specified concatenations are done the steps in Part II are done for the resulted array resulting the following grid maps.

“Threshold = 3”

5



10

15

Link to Outputs : Outputs