Mobile Robotics

Assignment-2

Team number: 16

Avani Gupta and Sreeharsha Paruchuri

Libraries used – Open3d, NumPy and Matplotlib.

Contributions – Both of us discussed approaches for both subtasks and maintained constant communication regarding progress/ strategies/ solutions/ problems. We'd help each other out if any of us encountered bugs and we'd give each other feedback regarding our respective outputs. This documentation was prepared by the both of us.

Avani: Coded the point cloud registration.

Sreeharsha: Coded the occupancy map.

Task 1 – Point Cloud Registration

Initially, a python script was coded up to change the bin files numbers for the sake of the main code being concise. The number range was taken from 0 – 77 to 10 – 87 in order to enable easy understanding of the code flow in the main "for" loop.

Concepts/Formulas used

Transformation done by multiplying matrix

With the points

While looping through each bin file and reading points :-

- The coordinate frame was converted from the LiDAR frame to the camera frame by multiplying with matrix

0	-1	0
0	0	-1
1	0	0

This we obtained from observing lidar and camera frame coordinate system. For example, the x of lidar frame is equivalent to z of camera frame. Hence, we get first column. Similarly substituting y, z equivalents we obtain the above matrix.

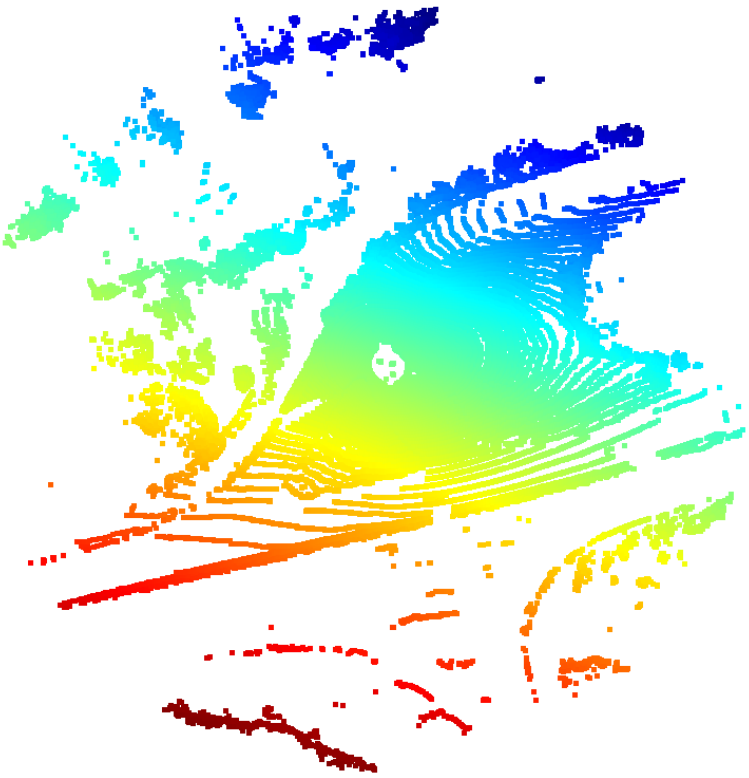
Now we simply multiply the matrix to do the transform from lidar to camera frame.

- The points were then transformed according to the ground truth poses given via the odometry. The objective of this step is to represent all the bin files in the coordinate frame of the first LiDAR bin. These poses were obtained from the text file "01.txt". Each line had twelve entries, this was reshaped into a 3*4 transformation matrix which when pre-multiplied to the points of the corresponding frame, brought the coordinate frame to that of the first LiDAR bin.
- Point cloud registration was done using functions from the open3d library. We do not perform down sampling as we're going use thresholding to remove noisy readings.
- Each individual point cloud was then appended to the global point cloud which has all the points from the 77 bin files.

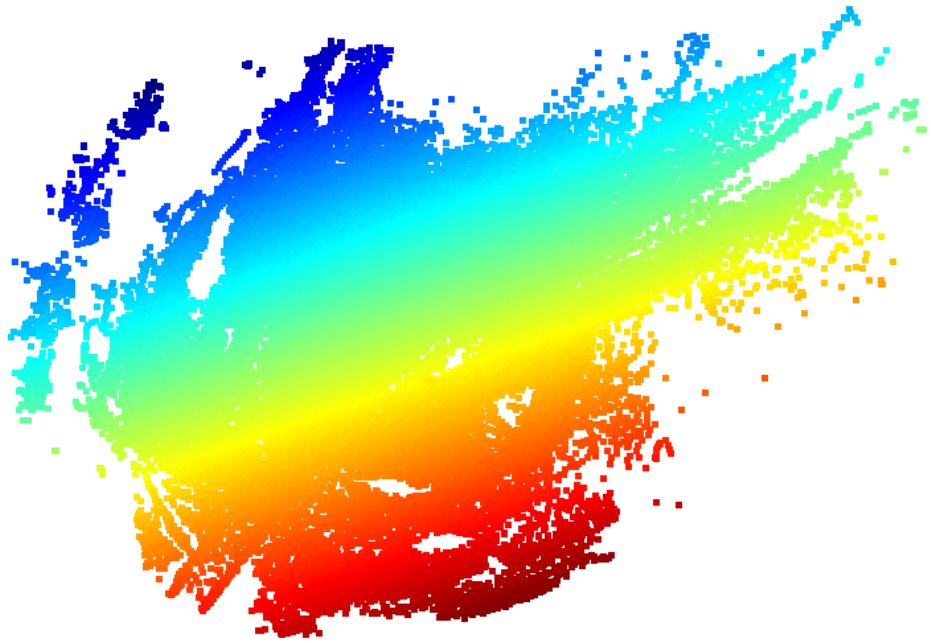
The final point cloud is then saved and displayed with the help of the open3d library.

Results

Point cloud 0 (without down-sampling)



Global Point cloud(combination of all 77 point clouds with down-sampling)



Task 2 – Occupancy Map Generation

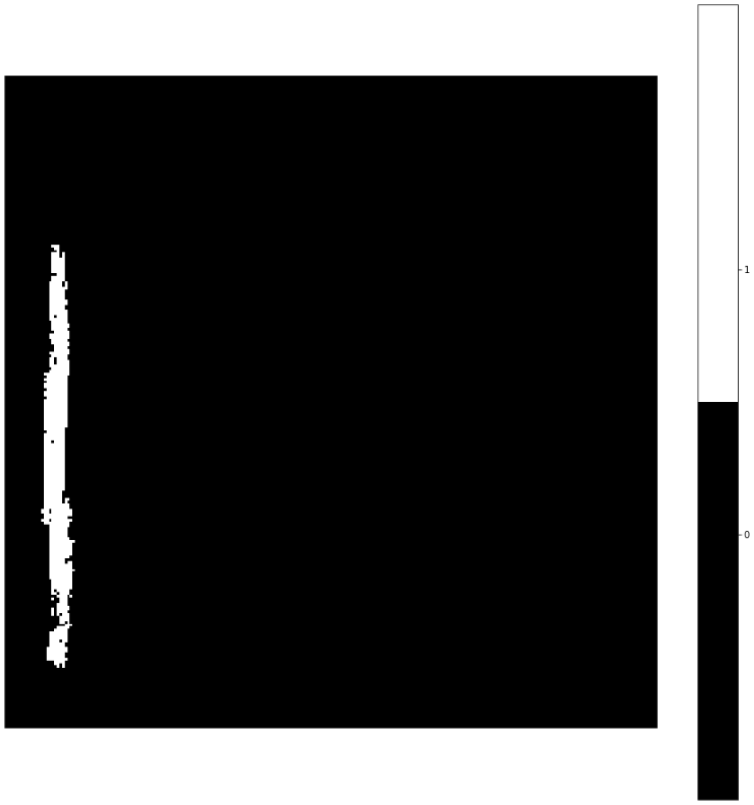
Once again, initially some preprocessing of the data was done. The ranges of the x, y and z coordinates were found out, the range of the Z data is redundant as the occupancy map has its x-axis as the x-axis of the transformed data and its y-axis as the y-axis of the transformed data but was done for the sake of completeness. The dimensions of the occupancy map were determined from this computation.

The below steps were performed four times, once for 5 bins, once for 10 bins, once for 15 bins and finally once for all 77 bins.

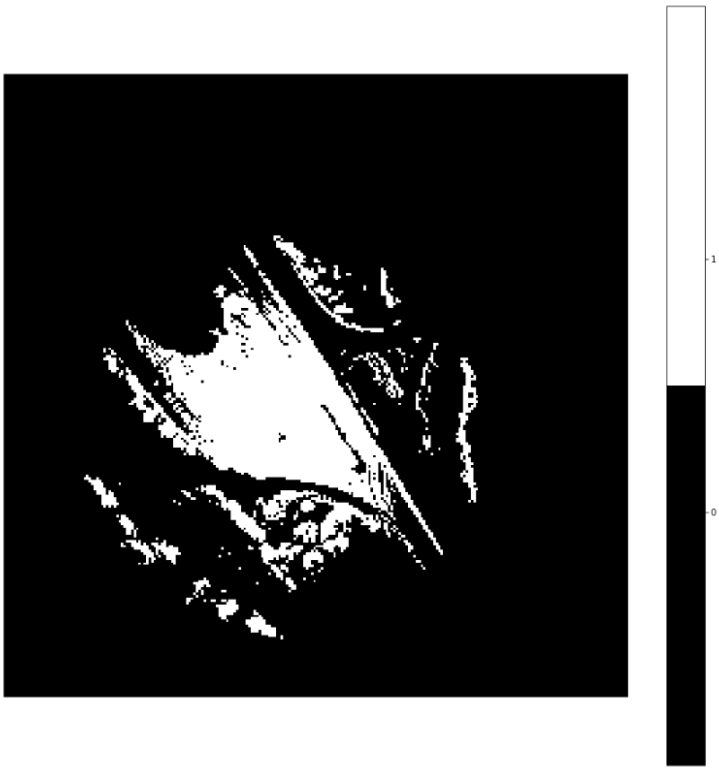
- Firstly, a loop runs through the number of bin files to be concatenated, this loop converts all the points into the coordinate frame of the initial bin on pre-multiplication with the poses matrices.

- The floor of the x, y values are taken and then added to a constant value to ensure that all the coordinates have positive values, that is, they lie in the first quadrant of the cartesian coordinate system.
- For each instance of a point having the same x, y values, the value of the corresponding cell in the occupancy map is incremented by 1. All the cells are initially initialized to 0. This is done for all the points.
- We now have a matrix whose cells have a value greater than or equal to zero. A cell value of zero indicates that the LiDAR didn't detect anything at that coordinate in the coordinate frame of the initial LiDAR bin. The higher the cell value, the more LiDAR beams struck a point at that coordinate in coordinate frame of the initial LiDAR bin.
- We iterate through the matrix and rewrite the value of each cell to be a zero if the value of the cell falls below the chosen threshold value. The threshold is essential as there will be noise in the LiDAR readings as well as the poses dictated by the odometry, these factors may throw up some erroneous coordinates where an object may not lie. Thus, thresholding is important as it reflects our confidence in that an object lies at the given x, y coordinate. The threshold was taken to be zero for one LiDAR bin, for 5,10,15 LiDAR bins the threshold was taken to be five.
- The final matrix with binary values is then displayed with the help of the library: Matplotlib. A cell is displayed as black if its value is zero, else it's displayed as white.

The occupancy maps project from XYZ to the XZ plane as when projecting to XY, the result was heavily skewed along one axis as shown below:



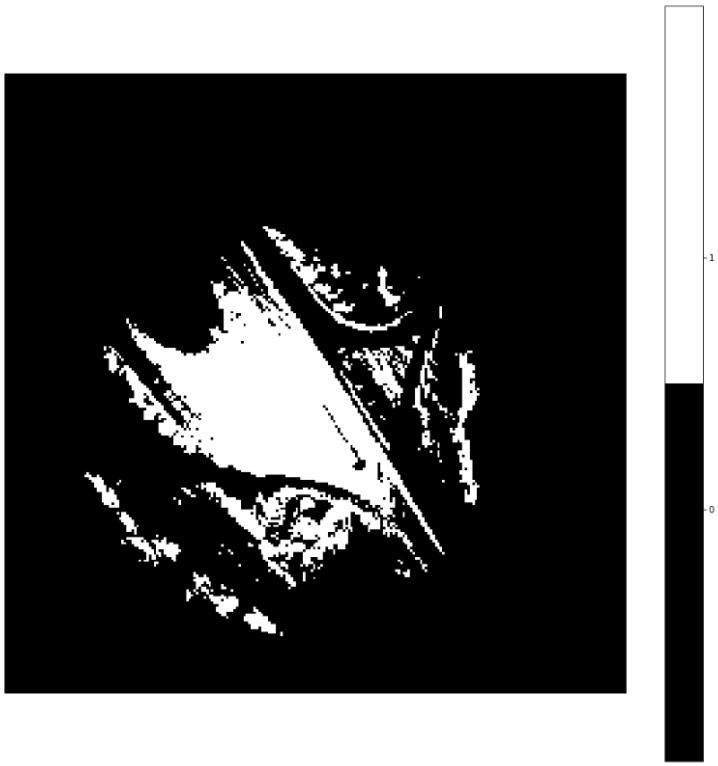
Compare this with the same down sampled point cloud but projected onto the XZ Plane, we can see much more detail.



5 consecutive bins:

The above XZ image is for the first 5 LiDAR bins.

10 consecutive bins:



15 consecutive bins:

Clearly, we can see that with more LiDAR readings, the occupancy map becomes denser.