Assignment 2
Mobile Robotics

Jivitesh Jain, Kanish Anand

1 Point Cloud Registration

The aim of this task was to register 77 point clouds, each in their own local frame into a global frame of reference, given the ground truth transformations between these frames.

The first task to achieve this was to convert the points from the local LiDAR frame to the local camera frame, because the global transformations are given with respect to the camera frame.

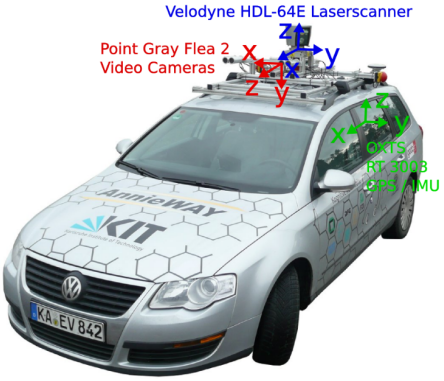

Figure 1: The local LiDAR and Camera Frames.

A ZYX Euler angle rotation of $(\frac{\pi}{2}, -\frac{\pi}{2}, 0)$ would convert the camera frame to the LiDAR frame, and hence send points the other way.

Thus,

$$T_{LiDAR}^{camera} = R_{Z'Y'X'}\left(\frac{\pi}{2}, -\frac{\pi}{2}, 0\right) \tag{1}$$

$$= \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}_{\alpha=\frac{\pi}{2},\beta=-\frac{\pi}{2},\gamma=0} \tag{2}$$

$$= \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3}$$

In our code, this step is achieved by the function `lidar_to_world` defined in `utils.py`.

After applying this rotation, the next step is to transform the point clouds into the global frame, using the ground truth transformations. This was done by converting the 3×1 points into 4×1 homogeneous coordinates, converting the 3×4 transformations into 4×4 homogeneous ones and then applying them to all the points of each point cloud. This is done by the function `make_homogenous_and_transform` defined in `utils.py`.

After this, all the points, now in a common global frame, were concatenated together, converted into an `open3D.utility.Vector3dVector`, and set as the `points` property of an instance of `open3d.geometry.PointCloud`, which was visualised.

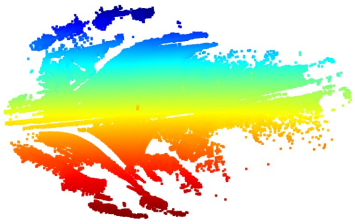

Figure 2: The registered point cloud.

2 Occupancy Map Creation

A 2-D or 3-D occupancy map presents information pertaining to whether a particular location in the environment is occupied or free, and is usually probabilistic.

As per the assignment requirements, we assumed all the given point clouds to be priors, and used the number of distinct y values occurring at a particular

x and z as an indicator of the probability of presence of an obstacle at that x and z. This was done because we were building a 2-D map for a 3-D imaged-environment.

We used a threshold fraction of 0.0002 to convert this into a binary map.

To go about this question, the first step was to down-sample the point clouds and get them into the world frame, as done in the first task. The `open3d.geometry.PointCloud` class was used to remove duplicate points. The points were rounded off into integers (because occupancy maps are discrete) and an image array of dimensions $Range\ of\ x\ values \times Range\ of\ z\ values$ was initialised and filled with the counts of the number of points with those x and z coordinates (after remapping into the range 0 to the size of the array). This was later converted into a fraction of the total number of points, and then compared with the threshold to output a binary occupancy map.

This was repeated for all the 77 point clouds, as well as a combination of 5, 10 and 15 of them.

Figure 3: The occupancy map.

The result files can be found here.