3D Localisation using NDT algorithm task procedure

Day 1:

1. Installed ROS melodic and ROS2 dashing on Ubuntu 18.04 (5h)
   (a) Since ROS2 dashing is supported on Ubuntu 18.04 and ROS melodic is mainly targeted at Ubunt 18.04 as well, I had to upgrade my Linux distribution from Ubuntu 16.04 to Ubuntu 18.04. While upgrading the system I had to solve numerous issues related to software packages compability and GUI problems
   (b) Forked given ndt_matching skeleton repository (which is available here - https://github.com/woookey/ndt_matching)
2. Visualised both the map available from map.pcd and lidar_data.bag using RVIZ to get the feeling about the environment and published messages (2h)
   (a) Found out that the map.pcd can be transformed to /PointCloud2 using PCL package, specifically using the following command, *rosrun pcl_ros pcd_to_pointcloud map.pcd* on ROS Melodic. Be default the node would publish all data in one message on topic name /cloud_pcd with frame id /base_link. The visualised map is shown in Figure 1. To make it work correctly on RVIZ, I had to set the global fixed frame to /base_link.

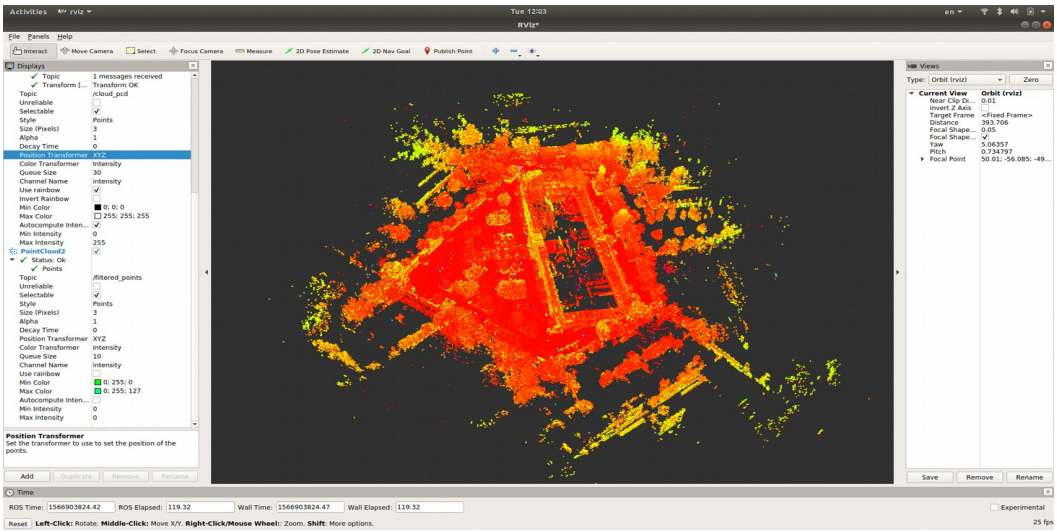

Figure 1: Map visualised in RVIZ

   (b) By subscribing RVIZ to /filtered_points topic I could visualise the LIDAR data recorded in lidar_data.bag using *rosbag play lidar_data.bag* command on ROS Melodic. The visualised one step of time is shown in Figure 2. The message type was also /PointCloud2 and the bag record consisted of 1275 messages.

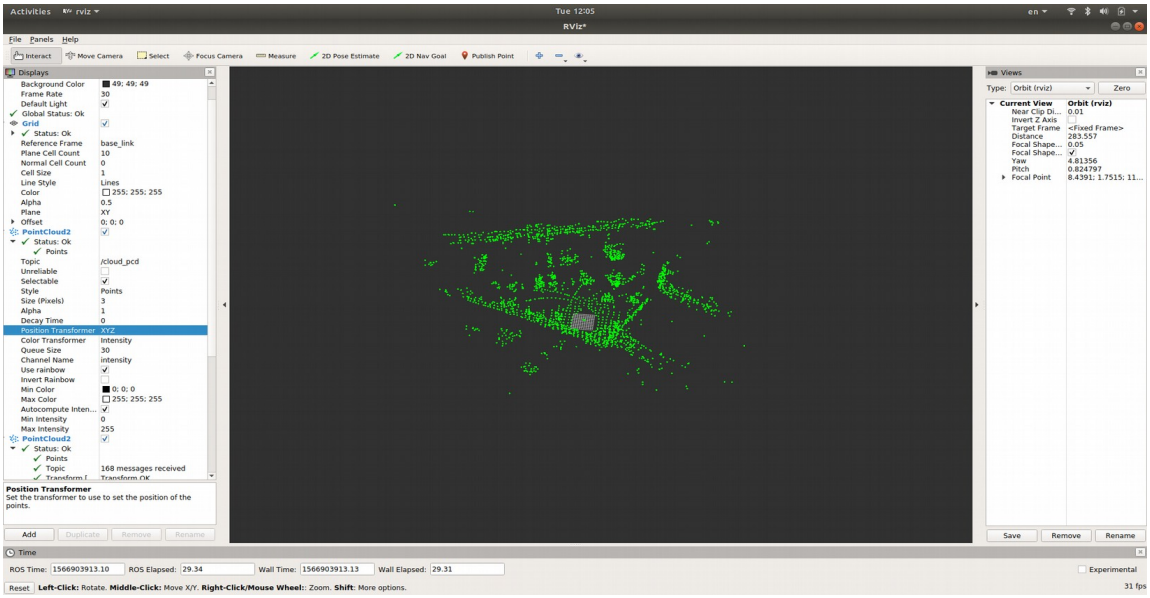

Figure 2: Visualised LIDAR measurements in RVIZ

3. Compiling initial ndt_node with working subscribers and a publisher (4h)
   (a) Had to figure out how to build and run nodes on ROS2. In addition, it turned out that ROS2 installation instructions do not mention catkin_pkg which is required by colcon package. This is a known issue already reported on ROS forum, and I solved it by installing catkin_pkg for python3 separately with following command, *python3 -m pip instal catkin_pkg –user*
   (b) Added subscription to PointCloud2 created from /cloud_pcd and /filtered_points, initial positions from /current_pose and publisher transmitting PoseStampled on /estimated_pose topic
   (c) Verified that messages published from ROS1 are received through ROS1/ROS2 bridge to ROS2 by streaming debugging information.

Day 2:

1. Setting Eclipse project for easier development (0.5h)
2. Understanding the 3D-NDT algorithm (1h)
3. Writing software (8h)
   (a) Implementing ndt_node.cpp – it took me some time to become familiar with PointCloud2 and PCL library
   (b) Started implementing the actual ndt_lib. I left the logical steps to implement NDT in comments but I think I would be able to complete if I had one day more. It would probably take some more time to tune the algorithm for performance. The output from running the program shown in Figure 3

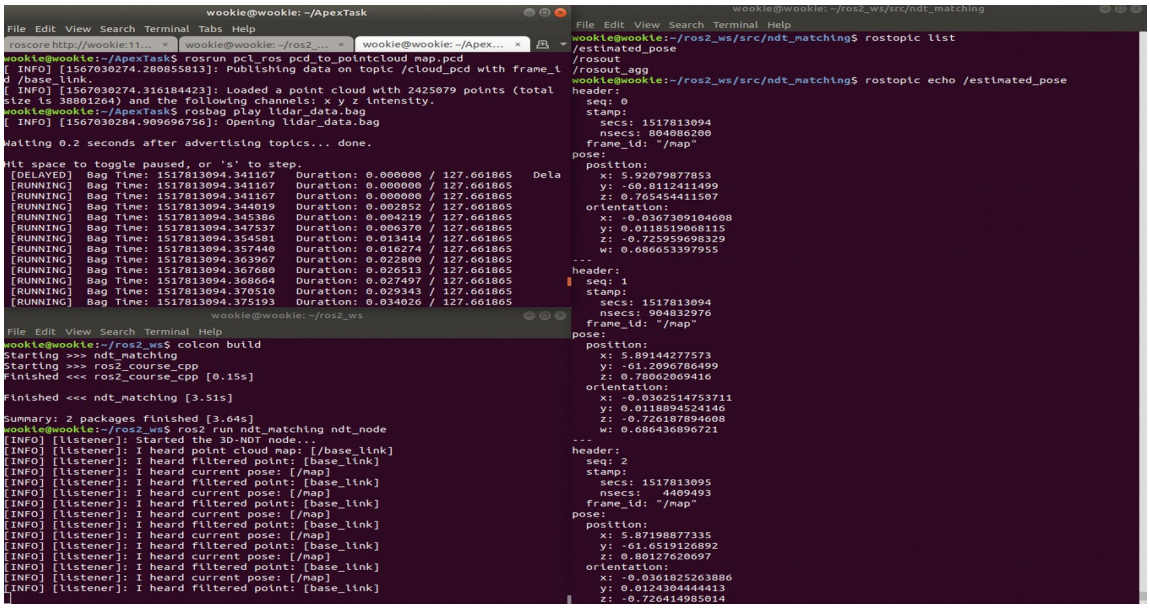

Figure 3 Output from running ndt_node with inputs

4. Documentation (0.5h)
   (a) Writing task procedure with the steps I had taken and Readme file with instructions on how to compile and run the program