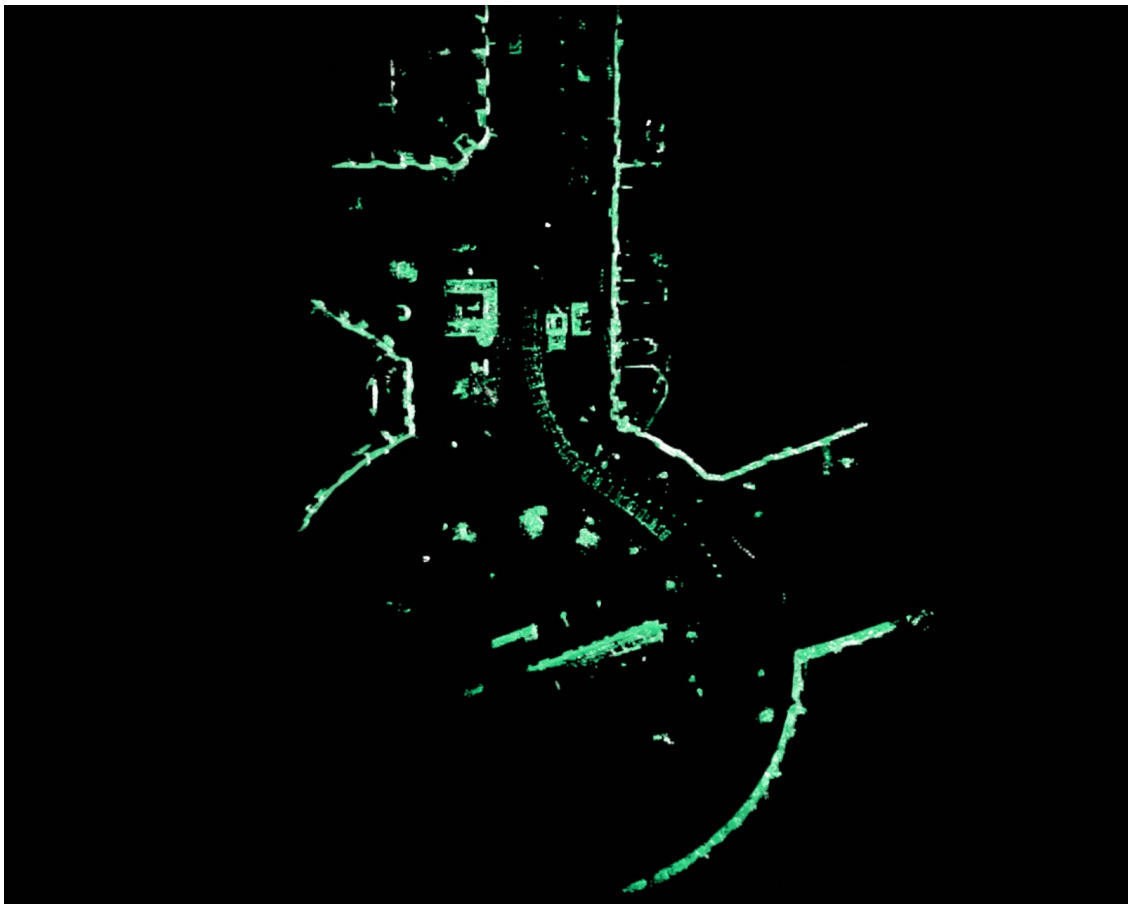

Mapping and 3D reconstruction based on lidar

Master's thesis in Master Programme Systems Control and Mechatronics

Liangyu Wang
Varun Ganapati Hegde

DEPARTMENT OF Mechanics and Maritime Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Mapping and 3D reconstruction based on lidar

Liangyu Wang
Varun Ganapati Hegde

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and maritime sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2021

Mapping and 3D reconstruction based on lidar
Liangyu Wang
Varun Ganapati Hegde

Supervisor: Ola Benderius, Department of Mechanics and Maritime Sciences, Chalmers
Examiner: Ola Benderius, Department of Mechanics and Maritime Sciences, Chalmers

Master's Thesis 2021:58
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Mapping and 3D reconstruction obtained for a *kitti* dataset.

Typeset in LaTeX
Gothenburg, Sweden 2021

Mapping and 3D reconstruction based on lidar
Liangyu Wang
Varun Ganapati Hegde

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

A 3D reconstruction and mapping framework aided by nonlinear filter optimization is introduced in this thesis project. A pipeline consisting of point cloud prepossesing, followed by a robust and rapid non-feature-based *normal distributions transform* (NDT) registration algorithm was developed for accurate mapping. An *unscented kalman filter* (UKF) solution to fuse *inertial measurement unit* (IMU) and *global navigation satellite system* (GNSS) measurements with augmented *continuous turn rate velocity* (CTRV) magnitude model is considered to obtain continuous *six degree of freedom* (6-DOF) pose estimation for accurate localization of the agent. The primary motivation for developing a high-quality mapping and localization system is because they play a key role in advancing towards an autonomous vehicle. Due to the lack of synchronized public datasets with IMU, GNSS, and *light detection and ranging* (lidar) measurements, the initial implementation of the mapping solution was tested over outdoor dataset from cars. The final phase of the project development is aimed at testing and tuning the program for the custom maritime dataset.

Keywords: mapping, 3D reconstruction, lidar, Kalman filter.

Acknowledgements

It is our pleasure to express our sincerest gratitude to our examiner and supervisor, Ola Bendirus, as he inspired our enthusiasm by providing us with guidance and resources for the project. We also thank him for providing the gateway to connect with academic research experts to understand the core concepts in developing effective solutions for the project.

We would like to deeply thank Anna for her input and collaboration for the software development for the sensor on-board the platform, which was common to both our projects. The insightful conversations and ideas exchanged with other thesis groups from CFSD and Revere during the weekly meetings also inspired us to improve the project.

Krister Blanch, Ted Sjöblom, Christian Berger, and other researchers who are part of the reeds project have our immense gratitude for collecting and sharing vast custom datasets. We would also like to thank Fredrik von Corswant and Arpit Karsolia for all their support.

Liangyu Wang, Varun Ganapati Hegde, Gothenburg, May 2021

Preface

This project was done as a collaboration between the two authors, including writing the report, testing and evaluating the software system. Liangyu Wang was the main responsible for implementing the C++ algorithm, as well as the construction of the software framework and the point cloud processing. Varun Hegde took the responsibility for the algorithm development and verification of UKF and lidar odometry.

Contents

List of Figures

List of Tables

1
Introduction

The surge of computational capacity in recent years has provided a solid foundation for the development of autonomous driving. The perception, planning and control capabilities of *autonomous vehicles* (AVs) have all been greatly developed in the past decade. Current assessments of the AVs suggest an annual growth rate of 63.1% within the next ten years [1]. This progress also affects the development of autonomous solutions in many other areas, such as maritime, indoor, industrial, and aerial, to mention a few. Decisions of AVs are based on the accurate understanding of the surrounding environment. Therefore, accurate perception and localization algorithms are the cornerstones of autonomous driving technology. With resources from Revere who is at the forefront of developing self-driving vehicles, this project aimed to implement a mapping algorithm. Compatibility and performance tests in maritime environment are conducted using data collected by the Revere Seastar research platform.

The lidar, *global navigation satellite system* (GNSS), and *inertial measurement unit* (IMU) are used as primary sensors for the perception and localization system. Lidar is essential for the 3D mapping of the environment as they produce point cloud data, which describes the depth information of the surroundings. A continuous process of point cloud registration enables 3D reconstruction of the surroundings. The GNSS provides 3D position estimations for the vehicle localization, which requires uninterrupted communication with at least four satellites. Inevitably, interruption of communication with GNNS satellite does occur in scenarios like in tunnels, under bridges, and nearby dense vegetation or urban infrastructure. The GNSS measurement frequency is lower than that of the IMU, while the IMU provides uninterrupted measurements. The sensor fusion of GNSS and IMU utilizes the complementary characteristics of sensors, which enables continuous accurate pose estimation at a high frequency. In this project, the mapping algorithms were tested on the standard datasets, such as the *kitti* dataset. Its performance on maritime environment will also be tested after the data collection.

1.1 Objective

This thesis project aimed to develop a reliable mapping and 3D reconstruction system for vehicles equipped with 3D lidar in a complex outdoor environment. The goal was also to use limited computing resources to accurately detect surrounding objects. Ultimately, the algorithm will be used on the racing car in *2021 Chalmers Formula Student Driverless* (CFSD21) and participate in the *2021 Formula Student Czech Republic* competition.

1.2 Research questions

The research questions chosen to guide the thesis project are:

- How robust and accurate can a lidar-based SLAM be on a 6-DOF moving vehicle?
- Can the maps generated by the SLAM solution be used for detecting objects moving relative to the world frame?

1.3 Outline

Sects. 2 and 3 present the background and theory leading to the chosen method. The fundamental methods and the state-of-the-art methods are introduced in Sect. 2 based on an extensive literature study in the field. This is followed by a description of the interaction and the knowledge exchange between peer groups. Sects. 4 and 5 describe the implementation and results of the algorithm. The analysis and discussion of the test results, as well as the follow-up summary and expectations for the future work, are shown in Sects. 6 and 7.

2
Background

Mapping and localization are the essential elements in the field of AVs, as they form the basis of *simultaneous localization and mapping* (SLAM), path planning and vehicle control functions. 3D reconstruction can express spatial environment information to enhance the AVs' perception ability. An accurate 3D reconstructed map serves in the object detection in the surrounding environment of the vehicles. It also enables the vehicle to integrate different sensors to accurately estimate its own pose. The reconstruction system also enables more accurate object tracking and collision avoidance. Novel studies on behavioural predictions of pedestrians and cyclists enables optimal decision making based on the situation.

There are various existing state-of-the-art methods of 3D mapping, which are LOAM [2], LIO mapping [3] and HDL SLAM [4]. The difference between these methods lies in the selection of the effective registration method and the extent of sensor fusion. The 3D mapping methods can be broadly classified into feature-based and non-feature-based methods. The prominent feature-based methods are *lidar odometry and mapping* (LOAM) and *Lidar inertial odometry and mapping* (LIO mapping). It is mainly composed of two algorithms, one is used for lidar odometry and the other is used for mapping. An overview of the LOAM method is shown in Fig. 2.1.

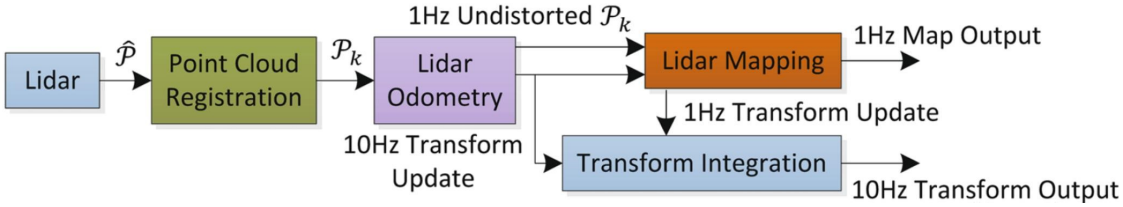

Figure 2.1: Overview of the LOAM method.

The registration step extracts sets of edge and planar feature points meeting certain criteria. The lidar odometry utilizes these feature points form lidar scans to obtain a geometric relation between them and to estimate the motion of the vehicle. The lidar mapping then uses the odoemtry information to register the point cloud to a submap obtained from the previous iteration. LIO mapping is an improved version of LOAM by using IMU measurements for pose estimation, which also improves the lidar odometry on the original LOAM. The feature based methods are computationally expensive and thus have lower mapping frequency. Using GNSS measurements to improve the mapping is also not considered in these algorithms. Therefore, this project considers the robust estimation of dynamics in all *six degrees of freedom* (6-DOF) and is suitable for the accurate mapping. An accurate pose estimation

system with the sensor fusion of GNSS and IMU is considered to enhance the speed and reliability of the 3D reconstruction system.

The point cloud containing noise and dynamic objects hinders the accurate transformation matrix estimation during the point cloud registration. So the preprocessing and object detection are the necessary steps in the mapping and 3D reconstruction process. Preprocessing is divided into two main directions, one is to reduce the point cloud density, and the other is to reduce the interference of the noise points. The point cloud features need to be preserved as much as possible in the preprocessing, so as to make the object recognition and 3D reconstruction more accurate. The detection and segmentation of the surrounding objects are essential for constructing consistent maps, obstacle detection, and path planning in the autonomous driving scenarios [5]. Object detection relies on the accurate point cloud clustering and object classification algorithms. The key to the segmentation is to extract the smallest bounding boxes from the clustered point clouds. The modern object recognition and segmentation methods are usually based on machine learning. The typical method is the classification method using deep neural network developed by Zdzislaw Kowalczuk and Karol Szymanski. This is a model-based deep learning method, which trains a model based on the PointNet architecture to detect and classify objects in the point cloud data [6]. It requires a lot of preliminary data preparation and model training, but it has high accuracy and fast speed under the guarantee of computing performance. Since this project does not have a very powerful computer platform, the method based on deep learning is not suitable.

Furthermore, since multiple functions need to work at different frequencies at the same time, containerizing the functions is a necessary operation. Docker is the most commonly used containerization technology. The Docker containers are isolated from each other and have their own software, libraries and configuration files. They can communicate with each other through well-defined channels [7]. The use of Docker also allows the system to achieve continuous integration and deployment, which makes the system to be able to be test on different platforms.

3
Theory

The theory used in this project is described in this section. Sects. 3.1, 3.2 and 3.3 introduce the sensor setup. Then, the coordinate system and transformation are explained in Sect. 3.4. Lastly, the theory behind localization and point cloud processing is shown in Sects. 3.5 to 3.7.

3.1 Inertial measurement units

Inertial measurement units work on different principles depending on the margin of accuracy of the intended application. Small size, cheap, and accurate inertial measurements are obtained through *micro-electro-mechanical systems* (MEMS). Therefore, MEMS IMU is the most common choice in autonomous driving applications. For the applications requiring higher accuracy, such as boats, helicopters, and safety-critical systems, high-precision IMU sensors are required. The *Revere Seastar research platform* has an advanced IMU, which is KVH P-1775.

Figure 3.1: KVH P-1775 high-precision IMU.

3.1.1 Gyroscope

Gyroscopes based on MEMES estimate the Coriolis force generated by the resonating mass. The Coriolis force is obtained by measuring the voltage across the capacitive elements around the mass. The Coriolis forces are measured along the three axes to produce rotational rates. The KVH P-1775 IMU with *fibre optic gyroscope* (FOG) technology is used on the maritime platform to get high precision measurements. The working principle of FOG is to measure the rotational rate based on the phase shift difference of two laser beams emitted in opposite directions. This occurs due

to the *Sagnac effect*, where three FOG sensors produce rotational rates on their respective axes.

3.1.2 Accelerometer

The accelerometer measures the change in capacitance to estimate the acceleration caused by the reaction force on the mechanical slider. The slider is a mass with extended electrodes attached to a spring and moves between the electrodes on a static frame in a specific direction. The difference in the distance between the electrodes affects the capacitance change. The accelerometers on the three axes measure the acceleration in their respective directions.

3.1.3 Magnetometer

Magnetometers are used to measure the magnetic forces based on various working principles. *Hall effect* and *magnetoresistance* are the most commonly used working principles. The KVH P-1775 IMU uses the *fluxgate* magnetometer principle. Two ferromagnetic elements generate equal and opposite magnetic fields through currents in opposite directions to cancel each other. The external magnetic fields are measured when it interferes with the balance between magnetic field, which is captured by the secondary coils around the two ferromagnetic materials. The calibrated magnetometer measurements are used to find the true north and estimate the heading of the vehicle [8]. However, due to the interference of the external magnetic field caused by the circuit current, permanent magnets, or other ferrous distortions, these sensors requires calibration to achieve accurate inertial navigation. The advantages of magnetometers in inertial navigation are limited, and the magnetic interference from vehicle components may bring uncertainty. Therefore, the magnetometer measurements are not used in this project [9].

3.2 Global navigation satellite system

The GNSS provides the geo-spatial position measurements of the receiver. It uses the *world geodetic system 1984* (WGS84) which depicts the oblate geometry of the Earth [10]. The distance information of at least four satellites are needed to accurately estimate the positions. Thus, GNSS cannot produce output in areas with limited satellite signals, such as caves and tunnels. In the dense urban environment, receiving the reflected GNSS signal will also cause multi-path errors and lead to erroneous outputs.

3.3 Light detection and ranging

Lidar is a sensor that obtains the distance from the sensor to the object based on the characteristics of light. The two widely used principles are the *time of flight* (ToF) approach and the *coherent detection* principle. The main idea of ToF principle is to calculate the time difference between emitting and receiving a lidar pulse. The

distance from the object to the sensor is obtained by calculating the half cycle of the time difference. The frequency, working principle, power, and sensing elements of the lidar system depend on the application scenarios. Lidar systems are integrated on the platforms such as *unmanned aerial vehicle* (UAV) and airplanes to generate the topographic maps, underwater topography or bathymetric surveys of the area.

3.3.1 3D laser lidar

3D laser lidar emits and receives eye-safe laser pulses with the wavelength of 905 nannometers to generate depth information of the surroundings, which is known as the point cloud. The point cloud contains direct centimeter-level measurements on the order of hundreds of thousand points. Lidars are robust to fluctuations or absence of ambient lighting and weather conditions such as rain or fog. However, the extreme fog condition still affects the range of the lidar. When comparing with *radar*, the detailed depth map enables the collection of a higher amount of distinct features and information of the surrounding objects.

Figure 3.2: Velodyne VLP-16 3D lidar.

3.4 Coordinate systems and transformation

The pose of the vehicle details its position and orientation and they differ based on the coordinate system chosen. The pose in reference to another coordinate system can be obtained from the transformation between the coordinate systems. 3D reconstruction of the map relies on accurate estimation of the pose through sensor fusion of measurements from sensors placed at different locations. Thus transformation between sensors enables the representation of pose in a common coordinate system and accurately estimate the pose of the vehicle.

3.4.1 ECEF and ENU coordinate system

The *Earth centered Earth fixed* (ECEF) is the most commonly used coordinate system when converting WGS84 into a planimetric coordinate. It considers the origin at the Earth's centre. The intersection of the Greenwich line and the equator as the

X-axis. The intersection of the Greenwich line and the Earth's rotational axis is the Z-axis, and Y-axis is orthogonal to the X, Y planes at the origin. The ECEF is then converted to the planimetric *east north up* (ENU) coordinate system to capture the local motion of the vehicle.

The ENU system considers the movement in the north, east, and up directions as positive direction. The first global position estimation is considered as the origin. The up direction is the line perpendicular to the tangent of the Earth's surface intersecting the Earth rotational axis. The oblate sphere is converted to a plane using the standard Mercator projection system to represent the location in the widely known *Cartesian coordinate system* [11].

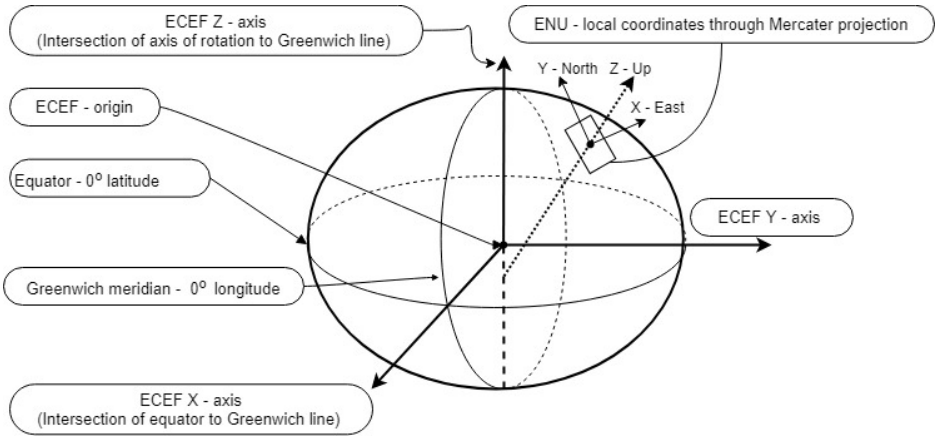

Figure 3.3: ECEF and ENU coordinate systems.

3.4.2 Body coordinate system

The body coordinate system enables us to capture the vehicle motion in all 6-DOF. Transformation and inverse transformation matrices from the body coordinates to the lidar, GNSS, and IMU coordinates are considered to use a common coordinate system. The IMU is usually placed close to the *centre of gravity* (COG) of the vehicle, so the IMU coordinates is usually considered as the body coordinate. In a general implementation, the origin of the global coordinate is the first GNSS readings obtained from integrated GNSS and IMU system. For a non integrated GNSS and IMU system, a transformation from the GNSS coordinate to the IMU coordinate T_{G-I} has to be carried out to capture vehicle motion accurately, as shown in Fig. 3.4.

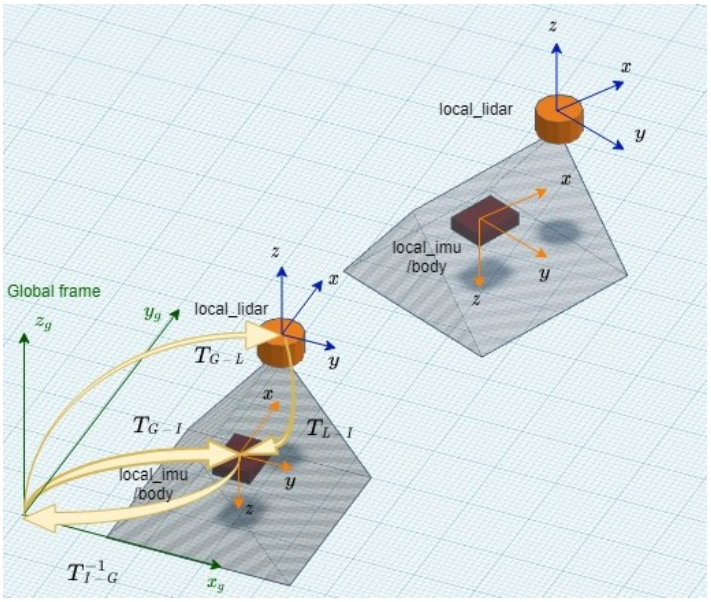

Figure 3.4: Transformation between global and sensor coordinates.

3.4.3 Lidar coordinate system

The point cloud data obtained from lidar considers the sensor position as its origin, so the orientation of the sensor is necessary to coherently align the point cloud to the global coordinate system. The orientation and position of the sensors on the vehicle are essential for determining the transformation between different coordinate systems. This transformation converts the map in the lidar coordinate into the body coordinate to compare the lidar odometry with ground truth. As shown in Fig. 3.4, the global coordinate is the ENU coordinate. The transformation from the global coordinate to the lidar coordinate is obtained by the transformation T_{G-L} and the transformation from lidar to global is given by T_{G-L}^{-1}.

3.4.4 Rigid body transformation

Moving the point cloud to different coordinates is usually done by applying an estimated 4×4 rigid body transformation matrix, which represents the geometric translation and rotation transformation in a Euclidean space. In the rigid body transformation, the length and included angle of the same vector in each coordinate system remain unchanged. The coordinates of the same object P in the world coordinate W and lidar coordinate C are different. This transformation is represented by a rigid body transformation matrix $T = [R\ t]$.

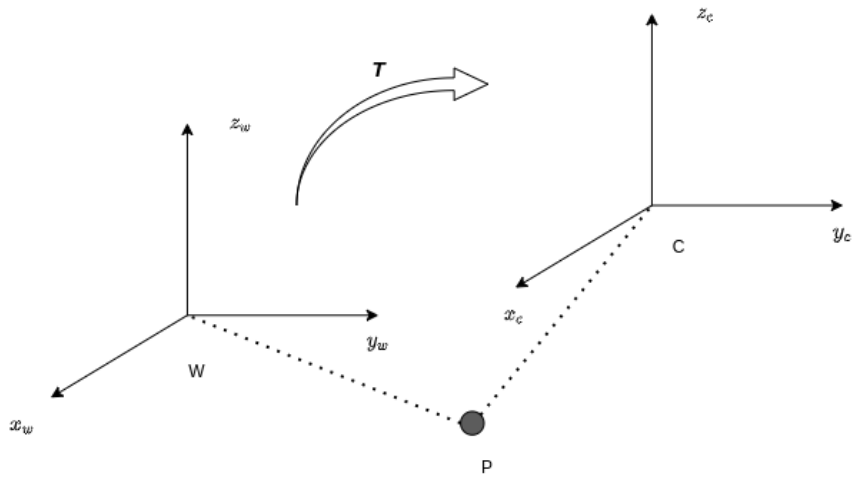

Figure 3.5: Rigid body transformation between two coordinates.

When considering the rotation transformation, a unit orthogonal basis $[e_1\ e_2\ e_3]$ becomes $[e_1^{'}\ e_2^{'}\ e_3^{'}]$ after one rotation. For the same vector a, its coordinates in the two coordinate systems are $[a_1\ a_2\ a_3]$ and $[a_1^{'}\ a_2^{'}\ a_3^{'}]$ respectively [12].

$$[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [\boldsymbol{e}'_1, \boldsymbol{e}'_2, \boldsymbol{e}'_3] \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} \tag{3.1}$$

The rotation matrix R is a 3×3 orthogonal matrix composed of the inner product between two sets of basis. The coordinate transformation relationship of the same vector before and after rotation can be described as:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^T\boldsymbol{e}'_1 & \boldsymbol{e}_1^T\boldsymbol{e}'_2 & \boldsymbol{e}_1^T\boldsymbol{e}'_3 \\ \boldsymbol{e}_2^T\boldsymbol{e}'_1 & \boldsymbol{e}_2^T\boldsymbol{e}'_2 & \boldsymbol{e}_2^T\boldsymbol{e}'_3 \\ \boldsymbol{e}_3^T\boldsymbol{e}'_1 & \boldsymbol{e}_3^T\boldsymbol{e}'_2 & \boldsymbol{e}_3^T\boldsymbol{e}'_3 \end{bmatrix} \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} \triangleq \boldsymbol{R}\boldsymbol{a}' \tag{3.2}$$

The translation matrix t is a three-dimensional vector, which is added on the rotated coordinates. So the complete rigid body transformation can be described as:

$$a' = Ra + t \tag{3.3}$$

3.4.5 Euler angle

The rotation matrix in 3D space has nine elements. However, one rotation only has three degrees of freedom, so that any rotation can be represented by a rotation axis and a rotation angle. Euler angle is introduced to represent the rotation around the XYZ axes. Roll represents the rotation around the X axis of the object and is represented by ϕ. Pitch represents the rotation around the Y axis of the object and is represented by θ. Yaw represents the rotation around the Z axis of the object and is represented by ψ.

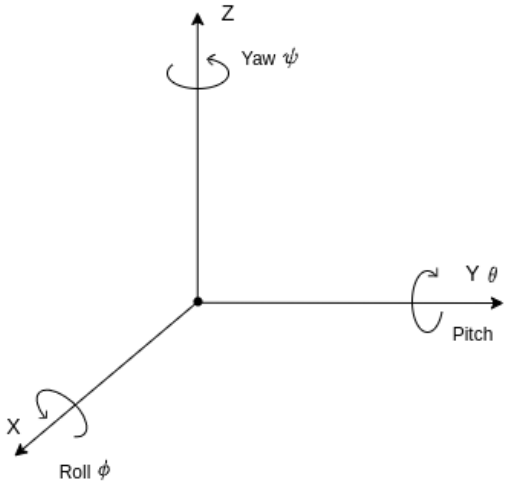

Figure 3.6: X-Y-Z Euler angle.

3.4.6 Quaternion

Quaternion is a compact and non-singular extended complex number, which can be used to describe 3D rotation with higher accuracy and solve the 'gimbal lock' problem of the Euler angle rotation. When the vehicle transforms its attitude and position, the transformation can be equivalent to rotations and translation transformations around its three axes. Therefore, the transformation matrix can use the quaternion to find the relationship between the two states. Quaternion q is composed of one real part and three imaginary parts, which is represented as:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{3.4}$$

where i, j, k are the imaginary parts and should satisfy:

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases} \tag{3.5}$$

The rotation matrix can be constructed by a given unit quaternion, which is used to calculate the posture of the vehicle:

$$R(q) = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 - 2q_3q_0 & 2q_1q_3 + 2q_2q_0 \\ 2q_1q_2 + 2q_3q_0 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 - 2q_1q_0 \\ 2q_1q_3 - 2q_2q_0 & 2q_2q_3 + 2q_1q_0 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{3.6}$$

Similarly, the quaternion can be also obtained by Euler angles [13]:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\varphi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\varphi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\varphi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\varphi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\varphi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\varphi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\varphi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\varphi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{3.7}$$

3.5 GNSS-aided inertial navigation system

GNSS-aided inertial navigation system (GNSS + INS) plays a vital role in the vehicle localization. The characteristics of IMU and GNSS are complementary and the integration of the two sensors is suitable for dead reckoning in GNSS denied areas. However, the integration errors from the navigation equation result in the drift of estimated orientation from the IMU. Whereas, GNSS works at a lower frequency of 10Hz but can provide highly accurate position measurements.

An integrated GNSS + INS system is based on either the open-loop architecture or the closed-loop architecture. The closed-loop GNSS + INS rectify the errors through the feedback of state estimations and GNSS clock drift estimation [14]. Nevertheless, it has the drawback of accumulating errors during the linearization step of a Kalman filter due to the elimination of higher-order terms of the navigation equations. Since an advanced FOG IMU is used in this project, an open-loop system with *unscented Kalman filter* (UKF) can provide continuous and accurate state estimation [15] [16].

3.5.1 Extended Kalman filter

Extended Kalman filter is the most traditional nonlinear filtering method, which assumes that the noise in the system is Gaussian noise. It uses Gaussian priors to predict the states by linearizing the motion model. Thus, it involves computation of the complicated Jacobians for the system dynamics to predict the state uncertainties. Kalman gains and the cross covariances are computed based on the uncertainties of the motion model and measurements. Predictions are updated periodically on the availability of new measurements, and the new estimations are considered as prior for the next iteration. A brief architecture of the EKF is shown in Fig. 3.7.

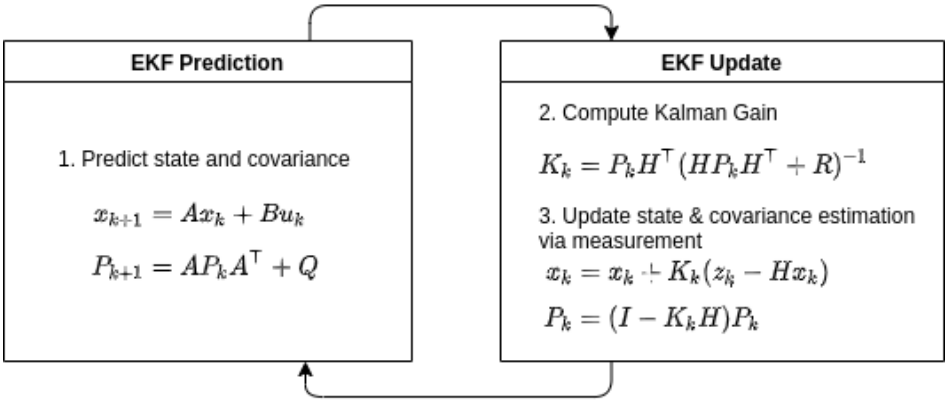

Figure 3.7: Brief architecture of EKF.

3.5.2 Unscented Kalman filter

Unscented Kalman filter proposed by Julier and Uhlmann is the most popular variant of sigma point methods in nonlinear filtering [17] [18]. The basic framework of

the UKF algorithm is shown in Fig. 3.8

The EKF linearizes the prediction model, so it's not as robust as the UKF in approximating nonlinear models. Although UKF is slightly more computationally demanding, it is not prone to errors due to the linearization. Therefore, the UKF is more suitable for nonlinear motion models such as the *error state total transition model* [19], *constant velocity* (CV) model [20] or *constant turn rate velocity* (CTRV) model [21] [22].

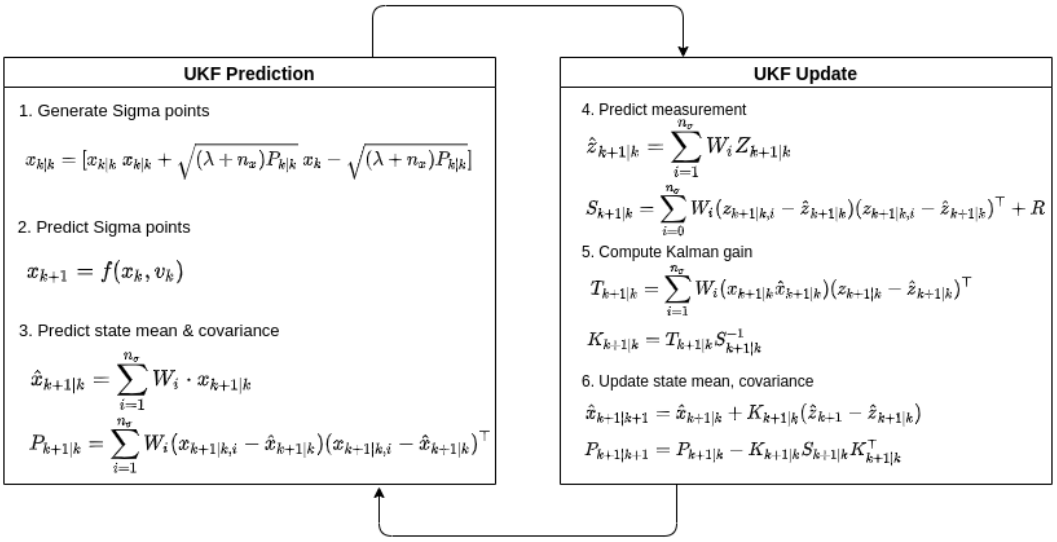

Figure 3.8: Brief architecture of the UKF.

3.6 Object detection and segmentation

It is necessary for the AVs to detect all the surrounding moving and fixed obstacles, so as to avoid collisions and dangers. At the same time, the effective segmentation of the drivable area can also significantly improve the accuracy of autonomous driving route planning and reduce the amount of point cloud registration calculations.

3.6.1 K–d tree

K–d tree is a binary search tree with other constraints, which is very useful for interval and nearest neighbour searches. When processing the point clouds, the k–d tree is three-dimensional. The construction of the k–d tree is a step-by-step recursive process. The tree has multiple nodes to store elements, and there are certain connections between the nodes. The upper node is called root node, the lower node is called child node, and the node without child nodes is called leaf node, which is shown in Fig. 3.9.

When each level is expanded, a hyperplane perpendicular to the corresponding axis is used to divide the remaining datasets along a specific dimension. The subtree on

the left of the node represents the point on the left of the hyperplane, and the same on the right side [23].

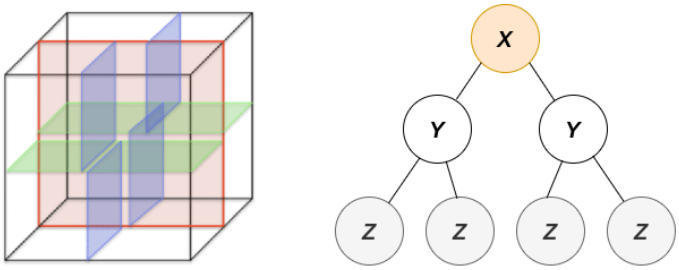

Figure 3.9: K–d tree space division.

Data searching in the k–d tree is an important part of feature matching, which is to find the point closest to the query point. The following formula determines the distance between two points in a $N-dimensional$ Cartesian space [24]:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{3.8}$$

A leaf node needs to be found which contains the target point in the k–d tree. Starting from the root node, the algorithm recursively searches the k–d tree downwards to find the closest point. If the current coordinates of the target point are less than the coordinates of the split point, move to the left child node, otherwise move to the right child node until the child node is a leaf node. With this leaf node as the current closest point, the search process is repeated upwards and backwards until the root node is returned.

3.6.2 Plane segmentation

In order to achieve speed and accuracy improvements on the limited hardware, the removal of ground point clouds is particularly important. The point cloud on the ground will also easily affect the clustering of obstacles causing an inability to identify objects accurately. In the space coordinate system, a three-dimensional linear equation with x, y, z can be used to express any plane:

$$ax + by + cz + d = 0 \tag{3.9}$$

Random sample consensus (RANSAC) is an iterative algorithm that can estimate the parameters a, b, c, d of a plane mathematical model from the input point cloud. The RANSAC method scores the sample set according to the interior points within the threshold. The process is repeated N iterations and the guess with the highest interior point score is retained.

However, due to the difference in angle and height on the ground, there is no perfectly flat plane that a simple model can represent. Therefore, sorting and segmenting the input point cloud by height and area to optimize the input point cloud will result in better plane segmentation.

3.7 Point cloud registration

Point cloud registration can be understood as integrating point cloud from different perspectives into a unified coordinate system through rigid body coordinate transformation. The transformation matrix is obtained by minimising the translation and rotation error between transformed and target point cloud. The registered point cloud can be used for the 3D reconstruction and object detection. Point cloud registration can be divided into rough registration and refine registration according to the initial conditions and accuracy.

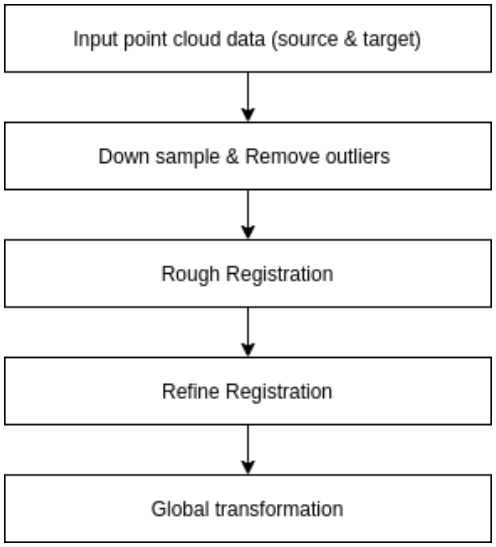

Figure 3.10: Point cloud registration process.

Rough registration is a fast estimation process to obtain the transformation matrix when the relative positions are unknown between the source point cloud and the target point cloud. Both the accuracy and reliability of the output results can be balanced. Common methods for rough registration include methods based on local feature descriptions and methods using statistical probabilities. Refine registration uses the known initial transformation matrix obtained from rough registration to calculate a more accurate transformation matrix. Therefore, the spatial position difference between the point clouds can be minimized.

Fast point feature histograms (FPFH) registration is a method based on the feature description. It extracts the neighbourhood geometric features of source and target and quickly determines the corresponding relationship between two point sets through geometric features. Finally, the transformation matrix is obtained through this relationship.

Normal distribution transform (NDT) registration is a method based on probability distributions. The algorithm will grid the target and reference point cloud sets, and count the normal distribution of each grid. Based on the probability density in the transformed grid, the registration accuracy can be estimated to obtain the

transformation matrix quickly.

Iterative closest point (ICP) is an optimal registration method based on the least square method, which pays more attention to the relationship between points. By selecting the corresponding point pairs, the optimal rigid body transformation is calculated until the convergence accuracy requirement is met.

In order to select a better registration method, different point cloud registration methods and method combinations are used to test the same pair of point cloud sets. The accuracy and speed comparison is obtained, which is shown in Tab. 3.1. The NDT registration shows its speed advantage but is less precise. The feature based method FPFH achieves higher accuracy. However, the feature acquisition process involves a lot of calculation, so the speed is slower than other registration methods. It is clear from Tab. 3.1 that the combination of NDT and ICP registration achieved a better balance of speed and precision. The ICP registration was not able to converge during the test, which proved the importance of rough registration to reduce the probability of mapping failure.

Registration type	Registration accuracy (Convergence accuracy)	Registration speed [s]
NDT	0.368704	0.661719
FPFH	0.185613	6.5235
ICP	177.007	0.178124
NDT + ICP	0.0528088	1.096469
FPFH + ICP	0.0528029	6.958273

Table 3.1: Registration speed and accuracy comparison with 3,897 input points.

4
Methods

This section describes the methods used in this project. Sect. 4.1 explains the UKF working principle. The detailed preprocessing is described in Sects. 4.2 and 4.3. Sects. 4.4 to 4.6 describe the critical methods used to investigate the research questions. The complete software framework is shown in Sect. 4.7.

4.1 Unscented Kalman filter

Considering the advantages of the UKF, it is implemented to achieve accurate tracking of all 6-DOF of the vehicle at a constant frequency. The performance of the filter primarily depends on the prediction and the update step. An accurate motion model capturing all aspects of the vehicle dynamics is necessary for the prediction step. In the update step, the UKF used in this project accommodates two different motion models and two measurement models.

4.1.1 Initialization

The UKF is initialized where the means of states (X, Y, ϕ, v) are set to the first measurement from the GNSS, and the other states are initialized to zero as priors. In contrast, the state covariance matrix P_x represents the uncertainty, which is initialized to a large value. The prediction step is then carried out on the prior followed by the update step to provide the filtered output.

4.1.2 Prediction step

The key elements of the prediction step are the state transition based on the motion model $f(x)$ and the uncertainty associated with the predicted state Q. Regular CTRV or CV model captures the dynamics for only three degrees of motion (X, Y, ψ). Therefore, the augmented models are considered in this project to capture all 6-DOF and process noise for the implementation. The prediction is carried out through an augmented CTRV model when the turn rate is above the threshold. Otherwise, an augmented CV model is used. The models in detail are presented in the Appx .1.

$$\vec{x}_{t+1} = \underbrace{\vec{x}_t + \int_t^{t+dt} \begin{bmatrix} \dot{p}_X(t) \\ \dot{p}_Y(t) \\ \dot{v}(t) \\ \dot{\psi}(t) \\ \ddot{\psi}(t) \end{bmatrix} dt + \vec{\nu}}_{f(\vec{x}_k, \vec{\nu}_k)} \tag{4.1}$$

The augmented system consists of three position states, one velocity, three Euler angles and its rates. The remaining augmented states are the noise of acceleration in the longitudinal direction ν_{aL}, angular rate of pitch, roll and yaw $\nu_{\dot{\theta}}$ $\nu_{\dot{\phi}}$ $\nu_{\dot{\psi}}$, and acceleration ν_{az} in Z axis. Thus, the number of states of the augmented system is fifteen ($N_S = 15$). The system noise covariance matrix is given by Q and the augmented system noise covariance matrix is given by P_S.

$$\vec{x} = (X, Y, Z, v, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}, \nu_{aL}, \nu_{\dot{\theta}}, \nu_{\ddot{\phi}}, \nu_{\ddot{\psi}}, \nu_{\mathbf{az}}) \tag{4.2}$$

$$Q = diag(\nu_{aL}, \nu_{\dot{\theta}}, \nu_{\ddot{\phi}}, \nu_{\ddot{\psi}}, \nu_{\mathbf{az}}) \tag{4.3}$$

$$P_S = diag(P_x, Q) \tag{4.4}$$

The magnitude of the elements in process noise covariance is determined by uncertainties associated with the extent of propagation of the state.

$$P_x = \text{diag}(\sigma_{aL}^2, \sigma_{aL}^2, \sigma_{az}^2, \sigma_v^2, \sigma_{\dot{\theta}}^2, \sigma_{\dot{\phi}}^2, \sigma_{\dot{\psi}}^2 \sigma_{\dot{\theta}}^2, \sigma_{\dot{\phi}}^2, \sigma_{\dot{\psi}}^2) \tag{4.5}$$

The prediction step starts with the generation of sigma points and the number of sigma points is decided by the size of the state vector. The scaling factor λ is a tuning parameter that decides how far away the sigma points are with respect to each other. The sigma points are then passed through the motion model. The mean and the covariance are recomputed to obtain the predicted mean $\hat{x}_{t+1|t}$ and the uncertainty $P_{t+1|t,}$. The predicted mean is computed based on the weighted sum of the sigma points. The predicted covariance is computed from the weighted square difference of the predicted mean and sigma points with the same logic.

$$\hat{x}_{t+1|t} = \sum_{i=1}^{n_\sigma} W_i \cdot x_{t+1|t} \tag{4.6}$$

$$P_{t+1|t} = \sum_{i=1}^{n_\sigma} W_i \left(x_{t+1|t,i} - \hat{x}_{t+1|t}\right) \left(x_{t+1|t,i} - \hat{x}_{t+1|t}\right)^\top \tag{4.7}$$

4.1.3 Update step

The key elements of the update step are the measurement models $h(x)$, Kalman Gain K, and the uncertainty associated with the measurement given by covariance matrix R. The method is initialized by generating sigma points $X_{\sigma,t+1|t}$ from the predicted mean and covariance. The sigma points are then used to obtain the mean of expected measurement $\hat{z}_{t+1|t}$ by computing the weighted sum of transformed sigma points $Z_{t+1|t}$ through the measurement model. The covariance of expected measurements $S_{t+1|t}$ is computed as weighted square difference of the predicted mean $\hat{z}_{t+1|t}$ and sigma points $Z_{t+1|t}$. Since the IMU and GNSS output data at different frequencies, the measurement model varies for different sensors.

The IMU measurement model updates the rotation rate obtained from the gyroscope. The covariance matrix for the model R_G is determined by the random angle walk specification obtained from the datasheet for the sensor.

$$h(x)_I = [\dot{\theta}, \dot{\phi}, \dot{\psi}] \tag{4.8}$$

$$R_G = diag[\sigma_{\dot{\theta}}, \sigma_{\dot{\phi}}, \sigma_{\dot{\psi}}] \tag{4.9}$$

While the GNSS and IMU measurement model uses GNSS measurements that directly provide the position, velocity and heading. It is also appended by the IMU model to capture the turn rates of pitch, roll and yaw. Velocity and heading are used without transformation. The covariance matrix for the model R_{IG} is determined by the standard deviation of the positional error and the *root mean squared* (RMS) error of heading obtained from the sensor datasheet.

$$h(x)_{IG} = [X_{GNSS}, Y_{GNSS}, v, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}] \tag{4.10}$$

$$R_{IG} = diag[\sigma_{GNSS}, \sigma_{GNSS}, \sigma_v, \sigma_\psi, \sigma_{\dot{\theta}}, \sigma_{\dot{\phi}}, \sigma_{\dot{\psi}}] \tag{4.11}$$

The Kalman gain K is computed through the cross-correlation matrix $T_{k+1|k}$ and expected measurement covariance $S_{t+1|t}$, which is dependent on R. The updated state vector mean $\hat{x}_{t+1|t+1}$ is computed by taking the difference in expected and actual measurements weighted by the K for the measurement model, while the updated covariance $P_{t+1|t+1}$ is obtained from $S_{t+1|t}$ and K.

$$\hat{x}_{t+1|t+1} = \hat{}_{t+1|t} + K_{t+1|t}\left(\hat{z}_{t+1} - \hat{z}_{t+1|t}\right) \tag{4.12}$$

$$P_{t+1|t+1} = P_{t+1|t} - K_{t+1|t} S_{t+1|t} K_{t+1|t}^{\top} \tag{4.13}$$

4.2 Point cloud preprocessing

3D laser lidar produces around 1.2 million points per second, which brings tremendous pressure on the limited computing resources. Therefore, the preprocessing step to reduce the point cloud density is essential for the point cloud registration and object detection.

4.2.1 Voxelgrid downsampling

Voxelgrid downsampling is performed by dividing the point cloud into uniform grids, which is called voxels. The points inside the voxels are averaged by computing their centroid to generate a point for each voxel. The grid size of the voxel is controlled to obtain a uniformly sampled cloud with the required density for the subsequent processes. The downsampled point cloud preserves the original features, so it can reduce the amount of calculation for the registration and object detection while ensuring the accuracy. The comparison of down sampled point cloud to the original point cloud scan is shown in the Fig. 4.1

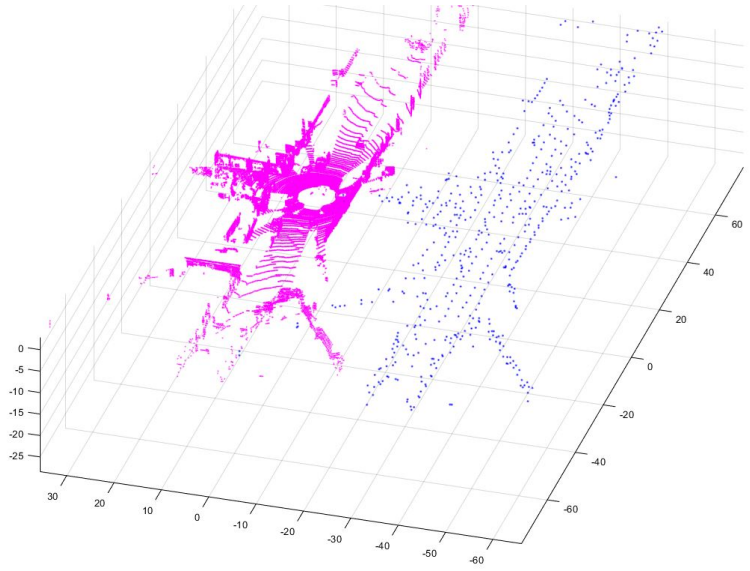

Figure 4.1: Original point cloud (left) and Voxelgrid downsampled point cloud (right).

4.2.2 Box cropping filter

The box cropping filter is used to retain the points that fall inside the *region of interest* (ROI). The points outside of the ROI are filtered out by defining the minimum and maximum boundaries for all points from the input point cloud.

This filter is primarily used to filter out points that are far away from the vehicle. These points are usually much noisier due to the long-distance environmental interference and may lead to poor registration. At the same time, points far higher than the vehicle are also not needed for the further process. These points will bring a lot of unnecessary calculations, thereby reducing the performance of the mapping algorithm. Box crop filter is also used to filter the redundant points caused by the sensors installed around the lidar. The removal of fixed noise points can significantly reduce the amount of data for the subsequent filtering process.

4.2.3 Pass-through filter

The pass-through filter is important for the selection of points within a certain range in a particular dimension. It is performed by sorting the indices of the points along a particular axis and consider the points exceeding a set threshold as outliers.

Pass-through filter is used to remove the points below the ground plane due to the measurement errors, so these fixed points will not interfere with the calculation, especially during the plane segmentation process. By reducing the redundant points, this filter can also help to increase the processing speed.

4.2.4 Statistical outlier filter

Noise points caused by the lidar errors, environmental interference, or object reflection characteristics will cause much interference in the subsequent processing. So the removal of outlier is imperative in the preprocessing. The main principle of this filter is to assume that the average distance between all points in the point cloud and its nearest k neighbour points meets the Gaussian distribution. The threshold of the filter is determined based on the mean and variance. When the average distance between a point and its nearest k points is greater than this threshold, the point is judged to be an outlier. This filter eliminates points that are not closely associated with objects resulting from measurement errors. The detailed algorithm is explained in App. E.

4.3 Ground plane segmentation

The accuracy of ground plane segmentation determines the accuracy and speed of point cloud registration. The two-level plane segmentation method composed of rough ground extraction and ground plane fitting ensures the segmentation and removal of the uneven ground.

4.3.1 Rough ground extraction

Since all the points on the plane have similar upward normal vectors, points with messy normal vectors can be found and removed. The normal vector of the point cloud surface is obtained by solving the normal of the approximate plane tangent. The eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbours of the query point are used to solve the normal estimation problem. The covariance matrix is represented as:

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T, \mathcal{C} \cdot \overline{\mathrm{v}_j} = \lambda_j \cdot \overline{\mathrm{v}_j}, j \in \{0, 1, 2\} \tag{4.14}$$

where k is the number of the neighbour points of p_i, $\overline{p}$ is the 3D centroid of the nearest neighbors, and v are eigenvalues and eigenvectors respectively [25].

Denoising is achieved by traversing all points in a fixed range and selecting points based on the requirements. This point set is used to calculate the plane model from the input point cloud.

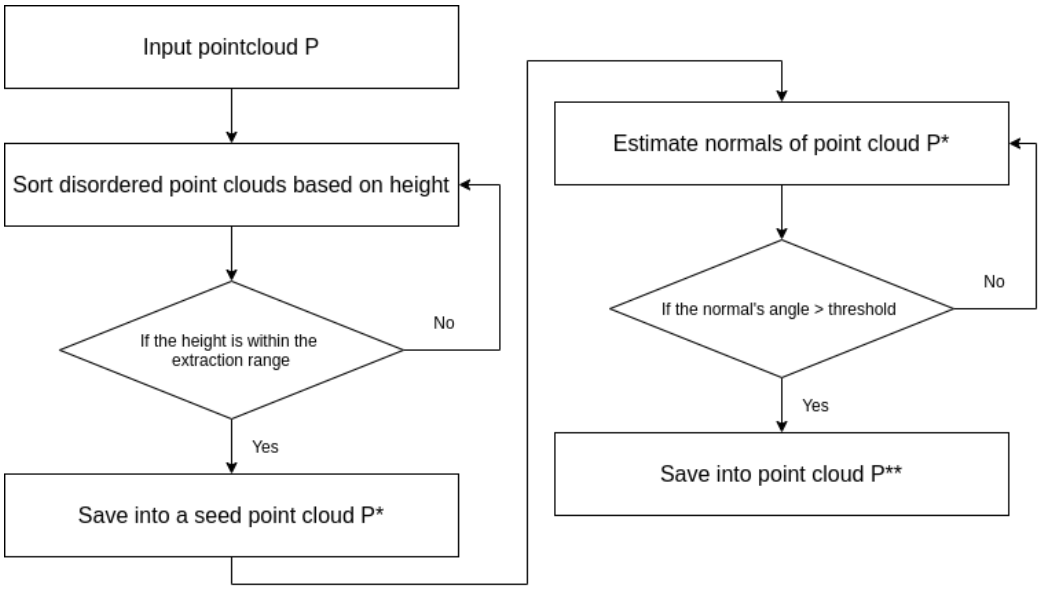

Figure 4.2: Rough ground point extraction process.

4.3.2 Ground plane fitting

Since the rough plane extraction has pre-extracted the point cloud, the accurate plane extraction can be achieved by finding the smallest feature value. To achieve the plane fitting equation, it is necessary to calculate the covariance matrix of the input point cloud and perform SVD decomposition to obtain the eigenvalues and eigenvectors. The two points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ are assumed to belong to the same plane, which is $ax + by + cz = d$. Then, the relation between them can be expressed as:

$$\begin{aligned} ax_1 + by_1 + cz_1 &= d \\ ax_2 + by_2 + cz_2 &= d \end{aligned} \Rightarrow \begin{bmatrix} (x_1 - x_2) & (y_1 - y_2) & (z_1 - z_2) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{4.15}$$

where $\mathcal{N} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ is the normal vector of the plane. By traversing all the points of the input point cloud to find the mean point $\bar{p} = (\bar{x}, \bar{y}, \bar{z})$, the plane model turns out to be:

$$A\mathcal{N} = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ x_2 - \bar{x} & y_2 - \bar{y} & z_2 - \bar{z} \\ \cdots & \cdots & \cdots \\ x_n - \bar{x} & y_n - \bar{y} & z_n - \bar{z} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{4.16}$$

The normal vector $\mathcal{N}$ can be solved through the covariance matrix $\mathcal{C}$:

$$\mathcal{C} = \sum_{i=1:|p|} (p_i - \bar{p}) (p_i - \bar{p})^T, \tag{4.17}$$

By calculating this covariance matrix $\mathcal{C}$ and performing *singular value decomposition* (SVD) on it, each component of the normal vector is obtained [26]. The vector

corresponding to the smallest singular value is the ground normal vector $\mathcal{N} = \begin{bmatrix} A \\ B \\ C \end{bmatrix}$.

Points belonging to the ground plane are described as:

$$\begin{aligned} Ax_i + By_i + Cz_i + D \pm \delta \approx 0 \\ \Rightarrow Ax_i + By_i + Cz_i \in (-D - \delta, -D + \delta) \end{aligned} \tag{4.18}$$

where δ is the error threshold.

4.4 Euclidean clustering and extraction

In order to identify the objects, the ground-removed point cloud needs to be clustered and segmented before doing object classification. Euclidean clustering is a clustering algorithm based on the Euclidean distance metric, which is used for clustering and segmentation of point clouds.

Euclidean distance $d(x, y)$ is the straight-line distance between two points x and y in the Euclidean space, which is calculated as [27]:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{4.19}$$

For a certain point x in the space, the k–d tree nearest neighbor searching is performed to find the closest point p which also meets the threshold range, and store it in the point set Q. The algorithm terminates when the number of points in Q doesn't increase. Otherwise, points other than point p in the set Q must be selected to repeat the current process.

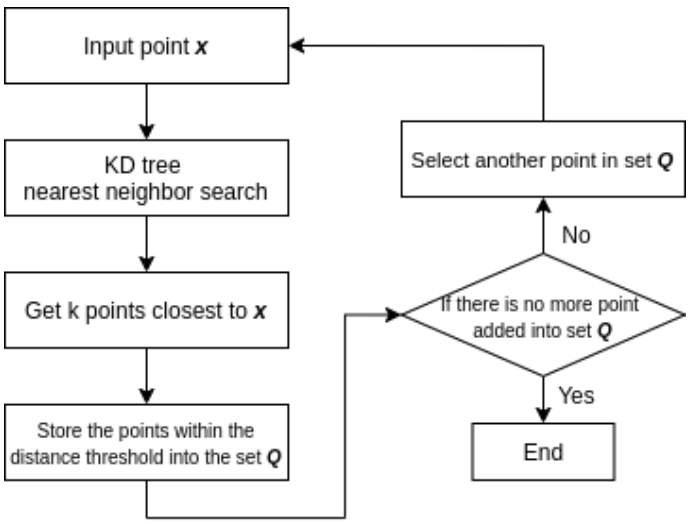

Figure 4.3: Process of Euclidean clustering.

The small clustered point clouds are filtered out by constraining the number of points. This can provide a better search range for the subsequent target detection

and improve the detection speed.

After point cloud clustering, by solving the optimal enclosing space of the discrete point set, the smallest bounding box can be obtained to replace the complex and irregular point cloud. The three main directions of the point cloud, centroid and covariance can be obtained by *principal component analysis* (PCA) [28]. After calculating the eigenvalues and specialty vectors of the covariance matrix, the main direction of the input point cloud is obtained and subsequently the width, length and height can be obtained.

4.5 Point cloud registration

In order to achieve fast and accurate point cloud registration, the NDT method is chosen as the rough registration, and the ICP method is selected as the refine registration.

4.5.1 Rough registration

NDT registration pays more attention to the probability density distribution of the target point cloud and the transformed source point cloud rather than the slight difference between point pairs. It is more suitable as a rough registration method especially in the situation without a guess of the initial transformation.

NDT registration process can be decomposed into:

1) Grid the source point cloud P_s and calculate the normal distribution parameters of each grid.
2) Use the transformation matrix T to transform the source point cloud P_s and calculate the probability density distribution of the transformed point in the grids of the target point cloud P_t.
3) Update g and H according to the result, and calculate the new step length.
4) Determine whether to end or not by converging or reaching the number of iterations. Otherwise, proceed to steps 2-4.

NDT registration expresses the reference point cloud as a set of local normal distributions through gridding. The points in the cells are represented as $\{x_i\}_{i=1,\ldots,n}$ and the probability of a point in the grid is modelled by the normal distribution $N(q, \sum)$. The probability p is described as:

$$p(x) = \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\mathbf{\Sigma}|}} e^{-\frac{(x-\mathbf{q})^T \mathbf{\Sigma}^{-1}(x-\mathbf{q})}{2}} \tag{4.20}$$

where $\mathbf{q} = \frac{1}{n}\sum_i \mathbf{x}_i$ is the mean and $\Sigma = \frac{1}{n}\sum_i (\mathbf{x}_i - \mathbf{q})(\mathbf{x}_i - \mathbf{q})^T$ is the covariance matrix for the cell [29].

The goal of NDT registration is to find the pose of the current source point to maximize the possibility that the current source point is located on the surface of

the target point cloud. The sum of the scores of all points in the scan is called the NDT point-to-distribution cost [30], which is expressed as:

$$\Psi = \sum_{i=1}^{n} p\left(T\left(p, x_i\right)\right) \tag{4.21}$$

In order to solve for the minimum cost, Newton's method is used to solve the optimal search step size, which is used to calculate the parameters of the new transformation matrix. The key to the Newton's method is to solve the step size Δp through the gradient matrix g and the Hessian matrix H:

$$\begin{aligned} H\Delta\vec{p} &= -\vec{g} \\ \vec{p} &\leftarrow \vec{p} + \Delta\vec{p} \end{aligned} \tag{4.22}$$

When the NDT cost reaches the minimum value, the optimal transformation matrix T_i of the current iteration is obtained.

4.5.2 Refine registration

The transformation matrix $T_{NDT} = (R_{NDT}, t_{NDT})$ obtained from rough registration is used as the initial transformation of the refine registration. A more accurate transformation matrix can be obtained with this optimization. The most widely used refine registration algorithm is the ICP registration, which can achieve a good balance between speed and accuracy. The working principle of the ICP registration is to find an optimal transformation matrix $T^* = (R^*, t^*)$ between source point cloud P_s and target point cloud P_t to minimize the error function, which is shown in Eq. 4.23 [31]:

$$R^*, t^* = \arg\min_{R,t} \frac{1}{|P_s|} \sum_{i=1}^{|P_s|} \left\| p_t^i - \left(R \cdot p_s^i + t\right) \right\|^2 \tag{4.23}$$

As shown in Fig. 4.4, this process can also be considered as reducing the distance d between the point P_t and the transformed point P_s as much as possible.

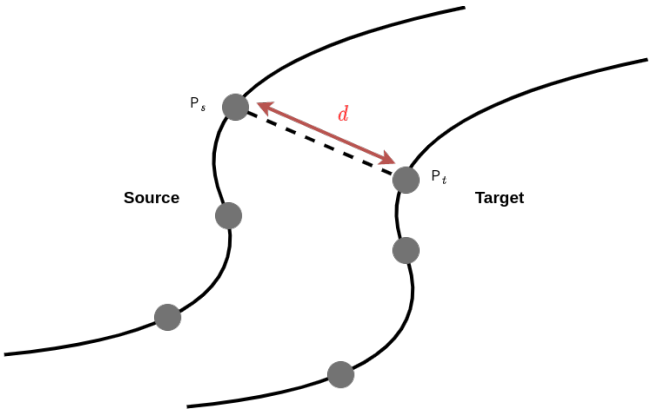

Figure 4.4: Point cloud registration distance metrics.

The initial transformation R_{NDT}, t_{NDT} from NDT registration are used to transform the source point cloud to obtain a temporary transformed point cloud. This point

cloud is compared with the target point cloud to find the nearest neighbor of each point in the source point cloud. When the point-to-point distance is less than a certain threshold, it is considered that the corresponding point is found, and there is no need to traverse the entire point set. The process of ICP point cloud registration is shown in the Fig. 4.5.

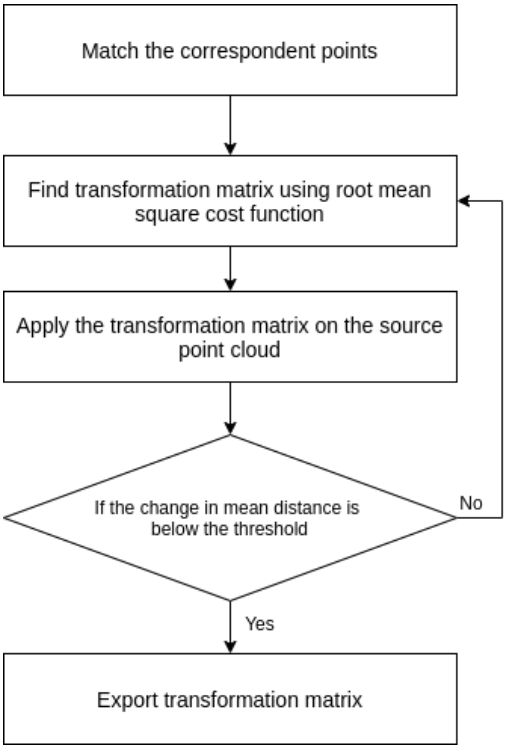

Figure 4.5: ICP registration process.

4.6 Lidar odometry evaluation

Since the IMU and lidar are located at different positions on the vehicle, coordinate system conversion is a necessary step to compare the lidar odometry with the integrated GNSS + IMU filtered odometry.
The initial heading of the vehicle is the orientation of the vehicle with respect to the ENU coordinates, which is considered as the ground truth coordinate system for the project. In order to obtain the lidar odometry in the ENU coordinate, the global transformation has to be initialized with the origin $GT_{L-[0]}$. Subsequently, it is transformed by the initial heading of the vehicle TV_I along with the transformation from GNSS + IMU to lidar T_{I-L}.

$$GT_{L-[0]} = T_{I-L} \cdot TV_I \cdot I \tag{4.24}$$

The global transformation of subsequent frames is obtained by multiplying the current global transformation with the local transformation obtained from the point cloud registration.

$$GT_{L-[i]} = T_{L-[i-1,i]} \cdot GT_{L-[i-1]} \tag{4.25}$$

In order to evaluate the accuracy of 3D reconstruction, the transformation from the global lidar coordinate system to the integrated GNSS + IMU coordinate system $GTC_{L-[i]}$ is carried out on each frame.

$$GTC_{L-[i]} = T_{I-L}^{-1} \cdot GT_{L-[i]} \tag{4.26}$$

The rigid transformation matrix is converted to their state vector form for comparision. The sensor placement in the *kitti* dataset is shown in Fig. 4.6.

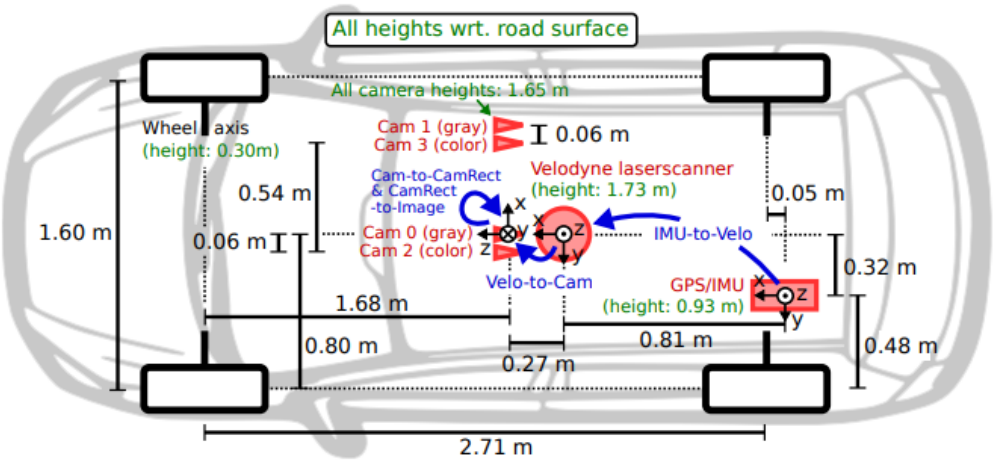

Figure 4.6: Kitti sensor setup.

4.7 Software framework

A containerized software framework is introduced to realize the mapping and 3D reconstruction with the assistance of UKF. The whole framework is divided into six different microservices based on the function requirements. The communication between microservices is based on the OpenDLV, which is a modern open source software environment to support the development and testing of autonomous vehicles. Different OpenDLV-based microservices are grouped in the *user datagram protocol* (UDP) multicast sessions, so that they can communicate with each other. Shared memory is used to store and disperse preprocessed point cloud data, since the size of these data is much larger than the size that UDP can share. Fig. 4.7 shows the structure of the entire system.

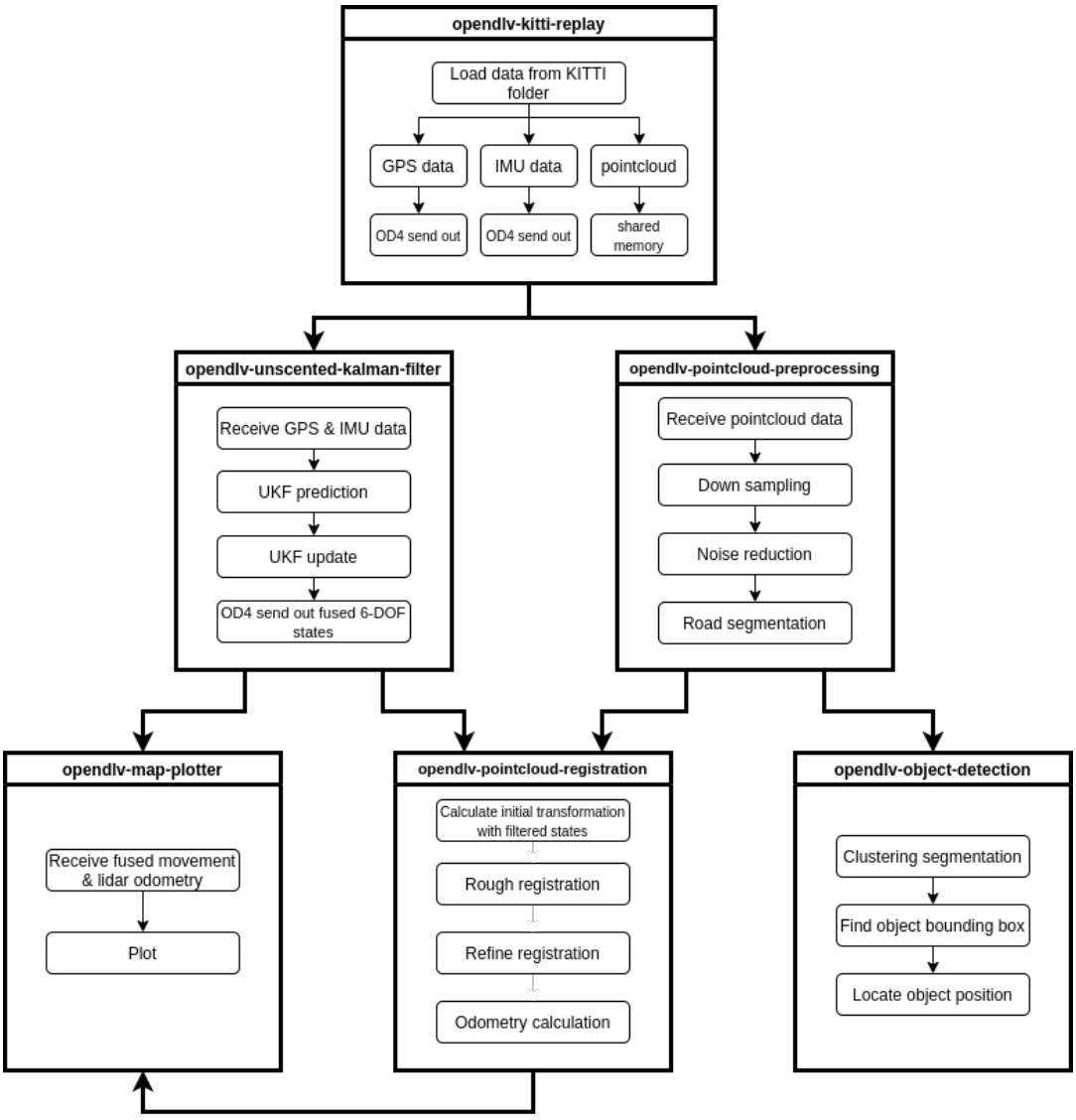

Figure 4.7: Implemented software architecture.

5
Results

The test results based on the *kitti* dataset are shown in this chapter. Preprocessing, localization, mapping and 3D reconstruction, and object detection are described in Sects. 5.1, 5.2, 5.3 and 5.4 separately.

5.1 Preprocessing

The output point cloud of the Velodyne VLP-16 has up to 130,000 points per frame. After a series of preprocessing including downsampling, outlier removal and ground plane removal, the point cloud density was reduced to around 6,000 points. The original and preprocessed point clouds are shown in Fig. 5.1. The ground plane indicated by the yellow points is effectively removed. The features of the point cloud were preserved, but the density was greatly reduced.

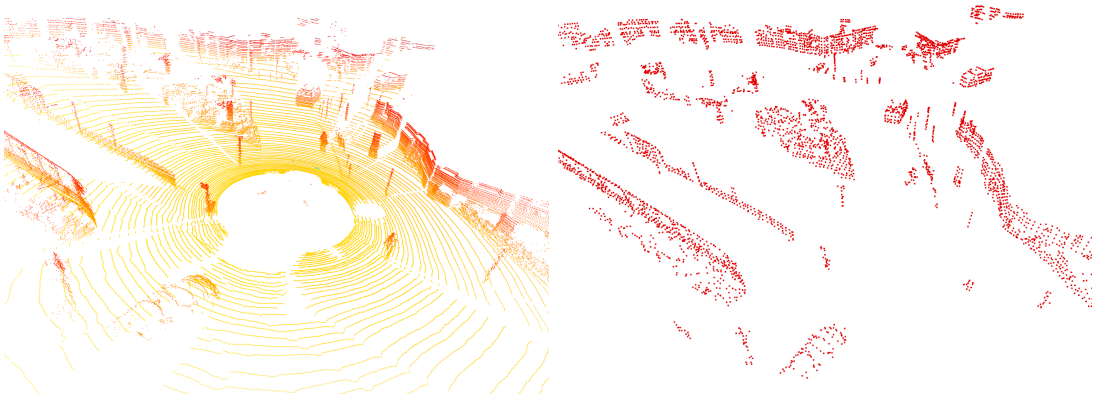

Figure 5.1: Original and preprocessed point cloud.

Registration of the preprocessed point cloud was faster in comparison to the original point cloud. However, fewer points also resulted in a decrease in the registration accuracy score. The influence of preprocessing on the point cloud registration in different scenarios is shown in the table below.

Influence of preprocessing on point cloud registration			
Dataset	Preprocessed	Convergence accuracy	Registration time(s)
0005		0.5259	1.330095
0005	✓	0.953334	0.197751
0093		0.677781	1.513680
0093	✓	1.493130	0.252758
0096		0.723640	0.960391
0096	✓	1.901210	0.122831
00117		0.682991	1.846660
00117	✓	0.887143	0.180213

Table 5.1: Influence of preprocessing on point cloud registration.

5.2 UKF performance

The UKF was used to obtain the localization estimations on the IMU and GNSS measurements from the raw *kitti* dataset. The following table lists its performance on the four sample datasets. The *root mean squared error* (RMSE) describes the extent of deviation of the filtered output from the ground truth. The *minimum error* (min) and *maximum error* (max) are used to describe the best case and worst case for the dataset. The units of the min, max errors in the following table are in meters.

Performance of UKF odometry						
Dataset	X RMSE	Y RMSE	X min	Y min	X max	Y max
0005	0.1916	0.5064	0.0017	0.0216	0.3543	1.1275
0093	46.007	16.112	0.0203	0.1800	88.3591	30.703
0117	0.5413	0.5862	0.0026	0.0143	1.9940	1.1935
0096	0.9270	0.7435	0.019	0.0017	2.8424	2.2651

Table 5.2: UKF performance test in different scenarios.

The *kitti* dataset 0005 was chosen to represent the performance of tracking the vehicle's yaw state due to the roundabout with a large arc. The comparison of estimation and ground truth is shown in Fig. 5.2.

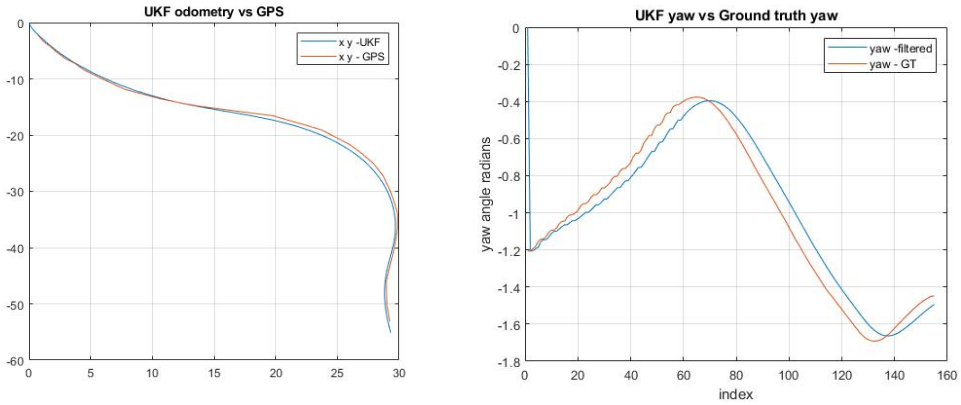

Figure 5.2: Position estimation against ground truth on dataset 0005.

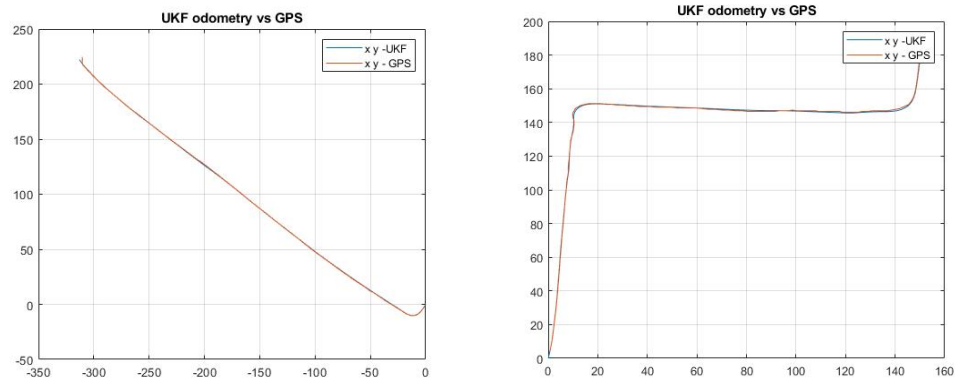

Figure 5.3: Position estimation against ground truth on dataset 0096 and 0117.

Noticeably, the filter performed poorly on the dataset 0093, but this was attributed to the inaccurate GNSS measurements. The Fig. 5.4 shows the performance of the UKF filter and lidar odometry in the case of inaccurate GNSS measurements. The UKF filter couldn't output accurate position and orientation estimation.

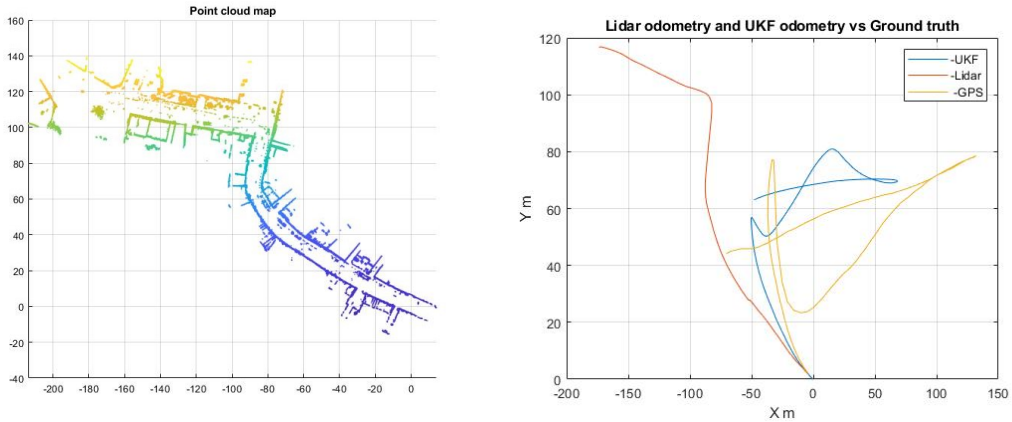

Figure 5.4: UKF performance on dataset 0093.

The residual plot shows the extent of bias due to the motion and measurement model and tune the UKF. The residuals of the states are obtained by the difference between the actual measurement values and the predicted measurements. An ideal residual plot should contain characteristics of a Gaussian distribution with zero mean for zero bias which can be only achieved with the ideal tuning settings and an ideal prediction model for the UKF. The dataset 0005 was chosen to describe the filter's performance through analysis of residuals, since it has distinct dynamics in all X, Y, and yaw states.

$$r = Z_{m(t)} - h(X_{p(t+1|t)}) \tag{5.1}$$

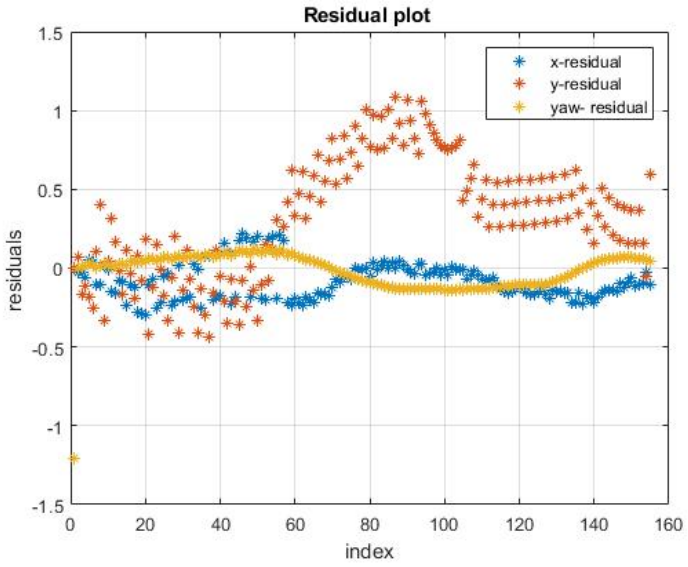

Figure 5.5: Residual plot of the X, Y and yaw estimation on dataset 0005.

5.3 Mapping and 3D reconstruction

Various tests have been performed to determine the optimal parameters of the NDT registration, which play a crucial role in achieving accurate 3D reconstruction. The quality of the map and the accuracy of the lidar odometry against filtered estimations for the datasets by using the guess from UKF for the NDT are shown in Fig. 5.6, Fig. 5.7 and Fig. 5.8. As dataset 0093 is compared with GNSS for ground-truth the deviation is expected due to factors mentioned previously. When GNSS and IMU can provide accurate measurements, the reconstruction system performs well and can generate accurate dense point clouds of the surroundings.

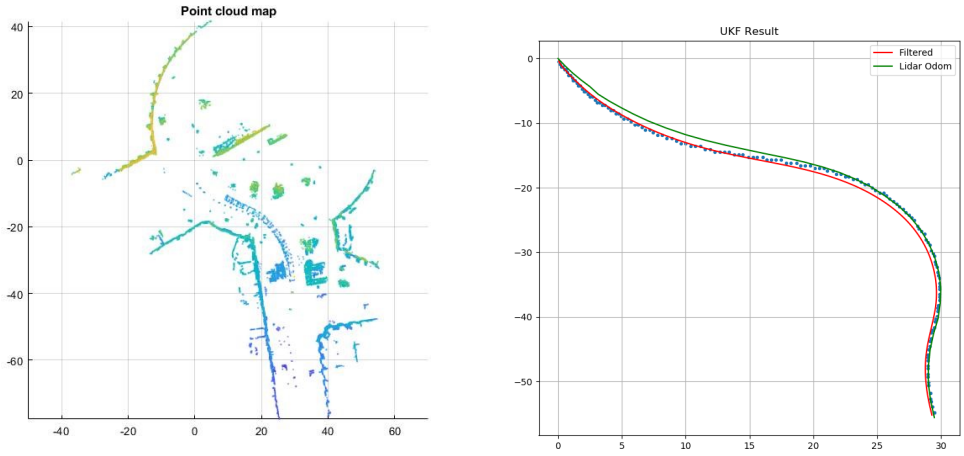

Figure 5.6: Point cloud map and odometry evaluation on dataset 0005.

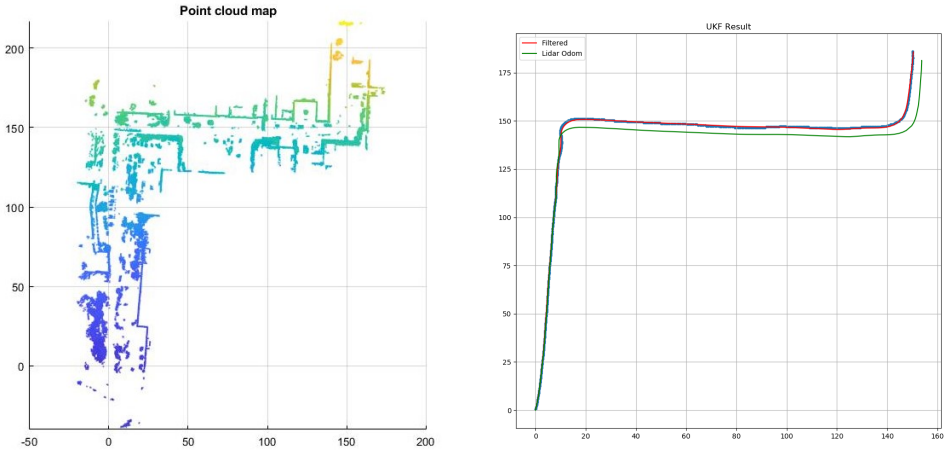

Figure 5.7: Point cloud map and odometry evaluation on dataset 0117.

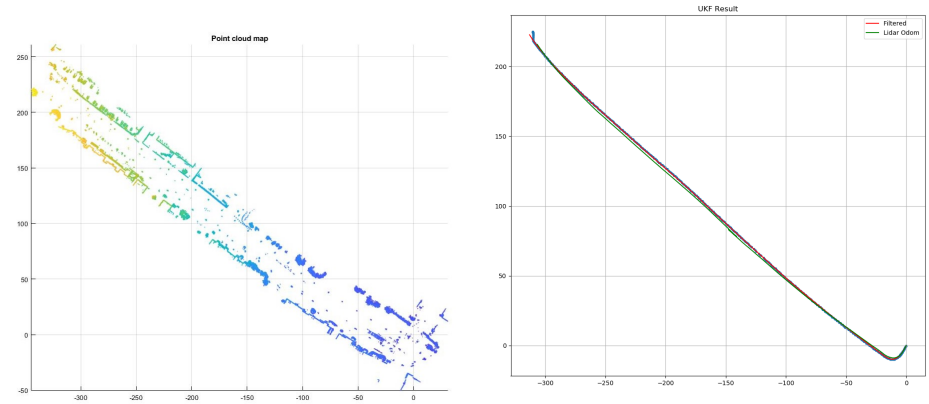

Figure 5.8: Point cloud map and odometry evaluation on dataset 0096.

The quality and accuracy of the map obtained can be inferred from lidar odometry obtained through registration. The lidar odometry with initial guess as the identity matrix for NDT registration is compared against the filtered output of the UKF, which are shown in the Tab. 5.3. The RMSE, maximum deviation, minimum deviation, and the deviation of the final state over the total travelling distance were considered in the evaluation of the 3D reconstruction. The units of the min, max errors in the following table are in meters.

Registration performance with identity matrix as initial guess.							
Data	X RMSE	Y RMSE	X min	Y min	X max	Y max	%final
0005	1.0117	0.9803	0.0146	0.1047	1.4540	1.7168	3.1477
0093	126.6641	37.1244	0.0711	0.1720	230.4530	72.535	25.04
0117	2.7839	3.5074	0.0035	1.4031	4.7290	6.4310	1.8404
0096	3.3919	1.5384	0.1662	0.0127	6.5643	3.2954	1.3689

Table 5.3: Performance with identity matrix as initial guess for NDT.

NDT registration performs better with accurate rotation estimation as the initial guess. Thus, mapping and localization performance with initial guess from only the orientation estimation of the UKF is tabulated below.

Registration performance with orientation as initial guess							
Data	X RMSE	Y RMSE	X min	Y min	X max	Y max	% final
0005	0.6401	0.9841	0.0163	0.0322	1.2397	1.5194	2.3642
0093	124.7118	36.4962	1.4660	0.1053	228.3960	72.5454	24.623
0117	2.7665	2.938	0.0069	0.0048	4.9868	11.039	2.6332
0096	3.3543	1.1188	0.0115	0.1764	6.2183	2.9761	1.7241

Table 5.4: Registration performance with orientation estimation as initial guess.

The mapping and localization accuracy with better initial guess of orientation is higher, but is inconsistent for the 0093 dataset as mentioned above. An initial guess with position estimations along with orientation was tested on the NDT registration. The evaluation results on different datasets are tabulated below.

Registration performance with position and orientation as initial guess							
Data	X RMSE	Y RMSE	X min	Y min	X max	Y max	% final
0005	0.3379	0.9292	0.0056	0.1352	0.3379	0.9292	2.4375
0093	114.2617	33.1179	1.3984	0.1389	189.0074	59.751	29.574
0117	1.5274	2.3951	0.0306	0.0023	3.2505	7.1213	1.4663
0096	7.7982	1.7974	0.0594	0.0554	14.8305	3.4108	3.6508

Table 5.5: Registration performance with position and orientation estimation as initial guess.

After the registration with the initial guess of position and orientation, the details are still reserved. It can be seen from the Fig. 5.9 that the characteristics of the car are still clearly visible.

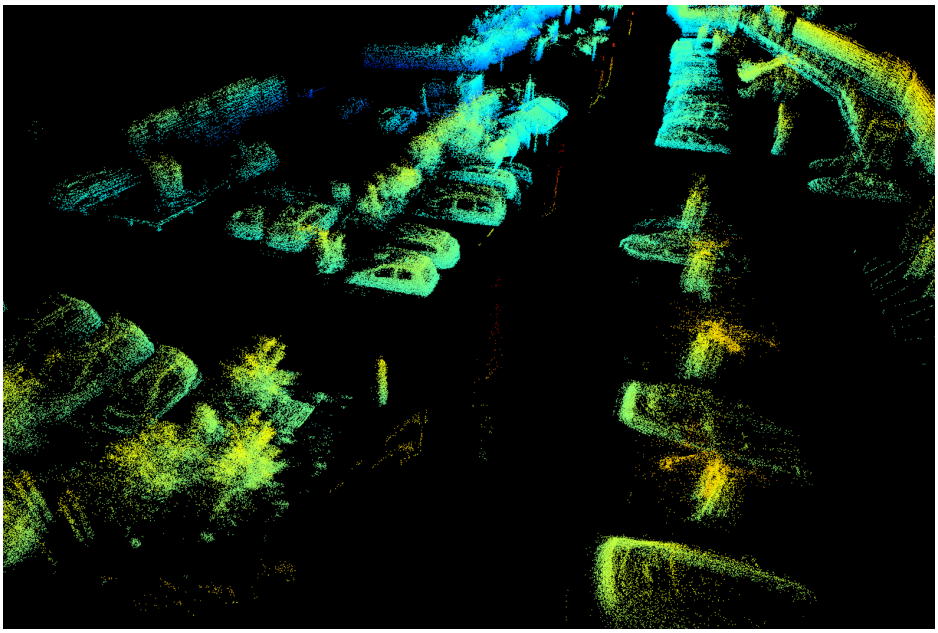

Figure 5.9: Details reserved after registration.

When the position estimation used for initial guess is poor due to the incorrect GNSS measurements for the 0093 dataset, it adversely affects the performance of the 3D reconstruction and lidar odometery. The inconstancy in the map generated with a UKF initialized guess is shown in Fig. 5.10.

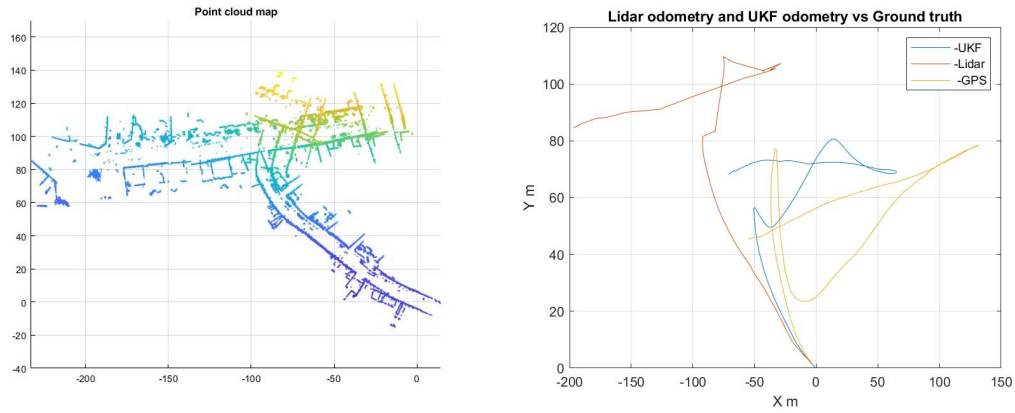

Figure 5.10: Registration failed when the GNSS measurement was wrong on the dataset 0093.

5.4 Object detection

By limiting the geometric form and Euclidean distance of the clustered point cloud, abnormal point clouds of incorrect clustering were removed, such as the walls and environmental interference. However, the clustered point cloud could not be classified, so some stationary objects were also identified by mistake, which is shown in the figure below.

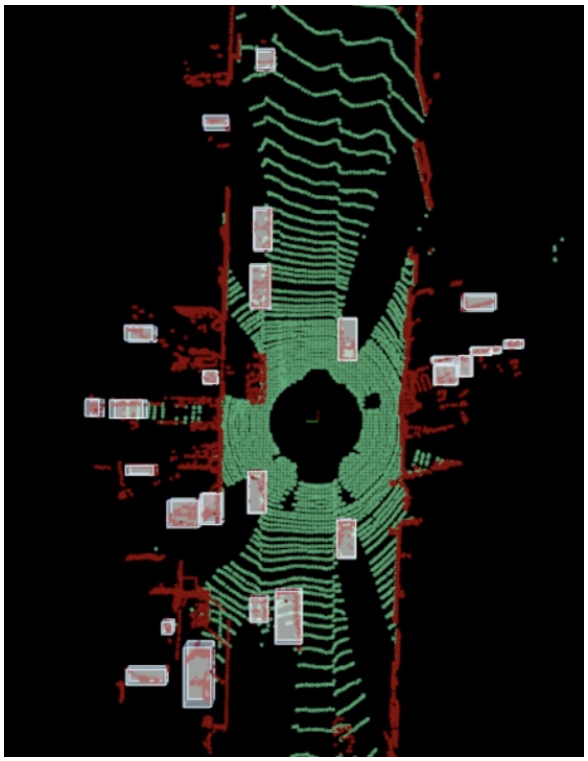

Figure 5.11: Real-time object clustering and bounding boxes.

Due to the movement of surrounding objects during the registration process, a large continuous point cloud were generated, which is called smear phenomenon. This can be seen in Fig. 5.12.

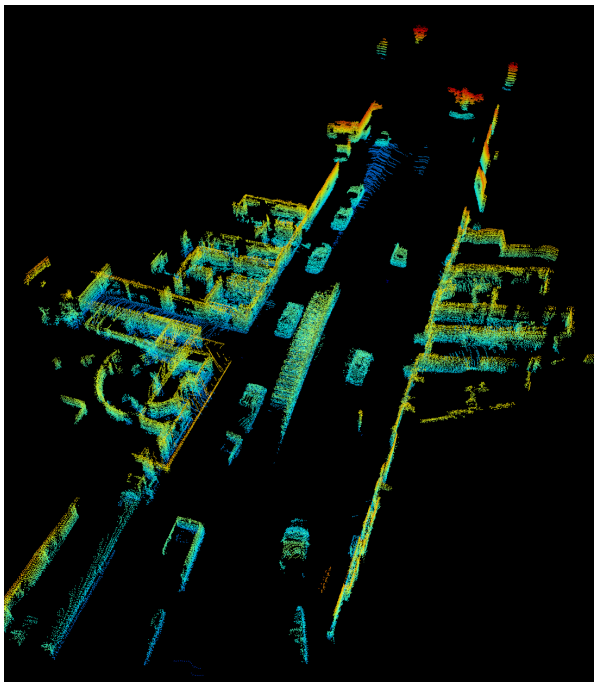

Figure 5.12: Smear phenomenon after registration.

When the registered point cloud was post-processed, the volume of the clustered point cloud was restricted to remove the outlying bounding boxes. By selecting a more precise ROI area, non-road point clouds such as walls were also filtered out, which made the post object detection more accurate.

Figure 5.13: Object clustering after post prostprocessing.

6
Discussion

The performance and potential improvement of UKF are discussed in Sect. 6.1. Further, the mapping and 3D reconstruction and object detection are discussed in Sects. 6.2 and 6.3. Finally, the performance optimization applied in this project and the future work are discussed in Sects. 6.4 and 6.5.

6.1 UKF performance

The current motion model was quite optimized for tracking the 6-DOF motion of vehicles such as cars. Good tuning of the process noise covariance and measurement noise covariance of the UKF ensured accurate 6-DOF state estimation. The UKF could maintain sub-meter accuracy for most of the datasets. Further tuning would only lead to marginal improvement in filter performance, so these tuning parameters were maintained across the datasets. The current motion model assumed the motion in Z axis as a random walk process, which could lead to poor estimation of motion in Z axis and other states. Thus better approximated model for the motion in Z along with velocity in Z axis could potentially improve the filtered 6-DOF estimation. The UKF was not robust to the long-term inaccurate measurements, which can be observed from the test result of 0093 dataset. Therefore, high-precision and reliable sensors are needed to ensure accurate filter estimations. As shown in the residual plot in Fig. 5.5, the residuals were close to the mean of the X and yaw states, but slightly deviated from the Y state. However, the residuals still had the characteristics of Gaussian distribution close to the mean values of zero. The UKF performance can be further improved when it is operated at a higher frequency with an advanced IMU.

6.2 Mapping and 3D reconstruction

The mapping and 3D reconstruction accuracy is mainly affected by the point cloud registration process, which also describes the movement of the vehicle. The lidar odometry described the mapping quality compared with the ground truth.

In the stage of rough point cloud registration, well tuned NDT resulted in better performance in terms of accuracy and speed. A more reliable rough registration was achieved with the help of the initial guess of the position and orientation obtained from the UKF. The performance improvement is noticed from the lower root mean square error in the test result. As observed from Fig. 5.10, using wrong UKF estimation for NDT registration resulted in the failure of lidar odometry. Using only

the accurate orientation estimation obtained by the IMU as part of the initial guess can effectively help continuous mapping. Thus, it is necessary to track the sensor status to indicate the potential UKF failure.

The resolution of the map was mainly impacted by the preprocessing and lidar measurement update frequency. In order to ensure the balance between speed and resolution, 2.5 Hz was the best lidar measurement update frequency under the current hardware conditions. Non-UKF-assisted 3D reconstruction could be used as an alternative method during sensor failure, or to detect possible GNSS interruptions, as shown in the test on dataset 0093 in Fig. 5.4. Initial guess from UKF with better motion model for dynamics in Z axis could also potentially improve the reliability and accuracy of mapping and 3D reconstruction.

6.3 Object detection

As shown in Fig. 5.11, point cloud clustering and bounding box extraction realized the basic object detection function. However, due to the lack of object classification algorithms, moving objects could not be effectively distinguished. Some static walls and obstacles were incorrectly identified especially in the crowded street environment. The fixed ROI setting and clustering tuning parameters also made the algorithm unsuitable for all the sceneries.

During the registration process, the smear phenomenon occurred due to the moving objects, which can be seen in Fig. 5.12. The smear phenomenon generated huge objects that actually didn't exist in the map. These large smear point clouds were detected by the large bounding box size and then removed in the generated map. In the post processing, the abnormal point clouds of incorrect clustering were removed by limiting the geometric form and Euclidean distance of the clustered point cloud. By considering the bounding boxes with unconventional shapes as wrong detections, the recognition of large stationary objects such as walls was greatly reduced. A quantitative assessment of the object detection could not be made for the current implementation due to the lack of datasets with annotations of moving and static objects for the lidar measurements.

6.4 Performance optimization

A series of optimization operations helped the system to achieve a stable 2.5 Hz mapping and 3D reconstruction in this project. After the point cloud preprocessing, the size of point clouds was greatly reduced while retaining the features, which is shown in Fig. 5.1. As a result, the subsequent processing speed was increased by nearly nine times. The decrease in the registration accuracy caused by the fewer points was within the acceptable range and did not have a big impact on the overall reconstruction. As shown in Fig. 5.9, the surrounding details were accurately retained in the reconstructed 3D map, which is very important for various point cloud processing in the later stage.

With the help of the multi-thread processing, a nearly three-times increase in speed of point cloud registration was achieved. Due to the limitation of experimental conditions, we were not able to use a higher specification computer to test whether the mapping and reconstruction system can be faster or more accurate.

6.5 Future work

Because of the time constraints, although the project has fulfilled the requirements of the research problem, there are still several parts that can be optimized and improved.

In order to make the UKF more robust to the vehicle's dynamic characteristics, a more general tracking model can be used, such as the total state model referenced in the discussion. These methods were constructed with the intention to test the performance of the filter over the maritime platform in order to verify the feasibility of the algorithm on different vehicle platforms. The filters can be further improved by implementing and improving novel techniques of adaptive tuning of the noise covariance matrices of the process and measurements.Mapping can be improved with better orientation guess from a *Madgwick orientation filter* [32] microservice, which uses all nine axis of the IMU to obtain continuous orientation estimations. This makes mapping and 3D reconstruction no longer overly dependent on accurate sensor readings.

A SLAM problem can be accurately solved by adding additional components to the current mapping and localization framework. General SLAM architecture includes active loop closing techniques along with a conditional triggered pose optimizer. The transformation of global mapping is optimized through the alignment of a specific set of local maps. In order to completely solve the SLAM problem, two more microservices are required. Firstly, a microservice which detects loops using the point cloud features has to be implemented. And the other microsrevice to recalculate optimized poses upon loop closure detection.

For the detection of moving objects, point cloud clustering is used to recognize moving objects. The object classification function is not realized. In the later stage, machine learning is expected to be used to detect and classify the objects in the bounding boxes, so as to realize different reactions to different moving objects during the mapping and 3D reconstruction process. By identifying and removing the point cloud generated by moving objects close to the vehicle, the smear problem in the reconstruction process can also be solved.

Due to the lack of sufficient data samples, the robustness and reliability of the system in different sceneries could not be verified as it requires testing over other datasets. Thus future work on datasets across different platforms from Chalmers and other research labs can be used to verify the robustness of the proposed method and build reliable maps for autonomous systems.

7
Conclusion

After testing in various scenarios using the *kitti* dataset, the mapping and 3D reconstruction system has been proven to be accurate and reliable. This system could update the map at a frequency of 2.5 Hz and use a nonlinear filter to provide accurate 6-DOF state estimations. After adjusting the parameters for different sensor combinations, the nonlinear filter could provide residuals with close to Gaussian characteristics, which reflected the reliability of filter estimation. By using the filtered 6-DOF state estimations as initial transformation for the reconstruction, mapping of the dataset with unstable measurements was greatly enhanced. For a dataset with reliable sensor measurements, the final localization errors between the results of lidar mapping and nonlinear filter were within five percent of the total travel distance.

It also turns out that the system which used filtered estimation to improve the mapping was counterproductive for datasets with sensor failures. The sensor's failure caused the measurements to exceed the tolerance range of the nonlinear filter, thereby affecting the speed and effect of 3D reconstruction. However, when the GNSS sensor failed, the mapping system was robust enough to be considered a stable alternate localization mechanism. In the subsequent development, this feature can also be used to cross-validate the reliability of the sensor system.

The current map obtained was proven to detect objects in different scenarios quickly and reliably. However, due to the time constraints, the object classification and tracking could not be achieved, which resulted in the framework not being able to distinguish between moving or static objects well. In some complex scenes, the recognition system could not give accurate bounding boxes and position information of the objects. The novel trend of using machine learning techniques to realize object recognition and classification can be explored to reveal the effectiveness of the method but also the limits and advantages between the methods in different environments.

Bibliography

[1] Grand view Research. *Autonomous Vehicle Market Size, Share Trends Analysis Report By Application (Transportation, Defense), By Region (North America, Europe, Asia Pacific, South America, MEA), And Segment Forecasts, 2021 - 2030*. 2021.

[2] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In: 2015. DOI: 10.15607/rss.2014.x.007.

[3] Haoyang Ye, Yuying Chen, and Ming Liu. "Tightly coupled 3D Lidar inertial odometry and mapping". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2019-May. 2019. DOI: 10.1109/ICRA.2019.8793511.

[4] Takahiro Sakai et al. "Large-scale 3D outdoor mapping and on-line localization using 3D-2D matching". In: *SII 2017 - 2017 IEEE/SICE International Symposium on System Integration*. Vol. 2018-January. 2018. DOI: 10.1109/SII.2017.8279325.

[5] Xieyuanli Chen et al. *Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data*. May 2021.

[6] Zdzisław Kowalczuk and Karol Szymański. "Classification of objects in the LIDAR point clouds using Deep Neural Networks based on the PointNet model". In: *IFAC-PapersOnLine* 52.8 (2019). 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019, pp. 416–421. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.08.099. URL: https://www.sciencedirect.com/science/article/pii/S2405896319304331.

[7] Docker. *Docker frequently asked questions [FAQ]*. 2021. URL: https://docs.docker.com/engine/faq/#what-does-docker-technology-add-to-just-plain-lxc (visited on 07/04/2021).

[8] Manon Kok et al. "Calibration of a magnetometer in combination with inertial sensors". In: *15th International Conference on Information Fusion, FUSION 2012*. 2012.

[9] Daehee Won et al. "Performance Improvement of Inertial Navigation System by Using Magnetometer with Vehicle Dynamic Constraints". In: *Journal of Sensors* 2015 (2015). ISSN: 16877268. DOI: 10.1155/2015/435062.

[10] United States Department of Defense. "National Geospatial-Intelligence Agency (Nga) Standardization Document". In: *Nga.Stnd.0036_1.0.0_Wgs84 National* 1 (2014).

[11] Gerald I Evenden and U.S. Geological Survey. *Cartographic projection procedures for the UNIX environment; a user's manual*. ENGLISH. Tech. rep. 1990.

DOI: 10.3133/ofr90284. URL: http://pubs.er.usgs.gov/publication/ofr90284.

[12] Qinrui Yan Xiang Gao Tao Zhang and Yi Liu. *Fourteen Lectures on Visual SLAM: From Theory to Practice*. 2019.

[13] Berthold Horn, Hugh Hilden, and Shahriar Negahdaripour. "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices". In: *Journal of the Optical Society of America A* 5 (July 1988), pp. 1127–1135. DOI: 10.1364/JOSAA.5.001127.

[14] Gianluca Falco, Marco Pini, and Gianluca Marucco. "Loose and tight GNSS/INS integrations: Comparison of performance assessed in real Urban scenarios". In: *Sensors (Switzerland)* 17.2 (2017). ISSN: 14248220. DOI: 10.3390/s17020255.

[15] Paul D. Groves. *Principles of GNSS Inertial and Multi-Sensor Integrated Navigation Systems - GNSS Technology and Applications*. 2008.

[16] D. Titterton and J. Weston. "Strapdown inertial navigation technology - 2nd edition - [Book review". In: *IEEE Aerospace and Electronic Systems Magazine* 20.7 (2005). ISSN: 0885-8985. DOI: 10.1109/maes.2005.1499250.

[17] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Vol. 3068. 1997. DOI: 10.1117/12.280797.

[18] E. A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000*. 2000. DOI: 10.1109/ASSPCC.2000.882463.

[19] Yuhong Yang, Junchuan Zhou, and Otmar Loffeld. "Quaternion-based Kalman filtering on INS/GPS". In: *15th International Conference on Information Fusion, FUSION 2012*. 2012.

[20] Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson. "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity". In: *FUSION 2014 - 17th International Conference on Information Fusion*. 2014.

[21] Ming Lin, Jaewoo Yoon, and Byeongwoo Kim. *Self-driving car location estimation based on a particle-aided unscented kalman filter*. 2020. DOI: 10.3390/s20092544.

[22] Robin Schubert, Eric Richter, and Gerd Wanielik. "Comparison and evaluation of advanced motion models for vehicle tracking". In: *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*. 2008. DOI: 10.1109/ICIF.2008.4632283.

[23] J. Bentley. "Multidimensional binary search trees used for associative searching". In: *Commun. ACM* 18 (1975), pp. 509–517.

[24] J. Tabak. *Geometry: The Language of Space and Form*. Facts on File math library. Facts On File, Incorporated, 2014. ISBN: 9780816068760. URL: https://books.google.se/books?id=r0HuPiexnYwC.

[25] Niloy J. Mitra, An Nguyen, and Leonidas Guibas. "Estimating surface normals in noisy point cloud data". In: *International Journal of Computational Geometry and Applications*. Vol. 14. 4-5. 2004. DOI: 10.1142/s0218195904001470.

[26] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications". In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2017. DOI: 10.1109/ICRA.2017.7989591.

[27] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. 2009. DOI: 10.1007/978-3-642-00234-2.

[28] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24.6 (1933). ISSN: 00220663. DOI: 10.1037/h0071325.

[29] Peter Biber. "The Normal Distributions Transform: A New Approach to Laser Scan Matching". In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 3. 2003. DOI: 10.1109/iros.2003.1249285.

[30] Arun Das, James Servos, and Steven L. Waslander. "3D scan registration using the Normal Distributions Transform with ground segmentation and point cloud clustering". In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2013. DOI: 10.1109/ICRA.2013.6630874.

[31] Kl Low. "Linear Least-squares Optimization for Point-to-plane ICP Surface Registration". In: *Chapel Hill, University of North Carolina* February (2004).

[32] Sebastian Madgwick. "An efficient orientation filter for inertial and inertial / magnetic sensor arrays". In: 2010.

A

Appendix

Appendices

A CTRV motion model

$$\vec{x}_{t+1} = \vec{x}_t + \begin{bmatrix} \frac{v_t}{\dot{\psi}_t}\left(\sin\left(\psi_t + \dot{\psi}_t \Delta t\right) - \sin\left(\psi_t\right)\right) \\ \frac{v_t}{\dot{\psi}_t}\left(-\cos\left(\psi_t + \dot{\psi}_t \Delta t\right) + \cos\left(\psi_t\right)\right) \\ 0 \\ 0 \\ \dot{\theta}_t \Delta t \\ \dot{\phi}_t \Delta t \\ \dot{\psi}_t \Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \cos\left(\psi_t\right) \cdot \nu_{a,t} \\ \frac{1}{2}\Delta t^2 \sin\left(\psi_t\right) \cdot \nu_{a,t} \\ \Delta t \cdot \nu_{az,t} \\ \Delta t \cdot \nu_{ax,t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\ddot{\theta},t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\ddot{\phi},t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\ddot{\psi},t} \\ \Delta t \cdot \nu_{\ddot{\theta},t} \\ \Delta t \cdot \nu_{\ddot{\phi},t} \\ \Delta t \cdot \nu_{\ddot{\psi},t} \end{bmatrix} \tag{.1}$$

B CV motion model

$$\vec{x}_{t+1} = \vec{x}_t + \begin{bmatrix} v_t\left(\cos\left(\psi_t \Delta t\right)\right) \\ v_t\left(\sin\left(\psi_t \Delta t\right)\right) \\ 0 \\ 0 \\ \dot{\theta}_t \Delta t \\ \dot{\phi}_t \Delta t \\ \dot{\psi}_t \Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \cos\left(\psi_t\right) \cdot \nu_{a,t} \\ \frac{1}{2}\Delta t^2 \sin\left(\psi_t\right) \cdot \nu_{a,t} \\ \Delta t \cdot \nu_{az,t} \\ \Delta t \cdot \nu_{ax,t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\theta,t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\phi,t} \\ \frac{1}{2}\Delta t^2 \cdot \nu_{\psi,t} \\ \Delta t \cdot \nu_{\ddot{\theta},t} \\ \Delta t \cdot \nu_{\ddot{\phi},t} \\ \Delta t \cdot \nu_{\ddot{\psi},t} \end{bmatrix} \tag{.2}$$

C UKF prediction step algorithm

Algorithm 1: Prediction step

Result: Predicted mean and covariance
Data: Prior mean $x_{t|t}$ and covriance $P_{t|t}$
Compute Sigma Points from prior $X_{\sigma,t|t}$;
for *i=1 to* $\forall$ *prior Sigma Points* **do**
  Transform through motion model {
  if $\dot{\omega} > \epsilon$ **then**
    $X_{\sigma,t|t} \xrightarrow{f(x,\nu)=CTRV} X_{\sigma,t+1|t}$;
  else
    $X_{\sigma,t|t} \xrightarrow{f(x,\nu)=CV} X_{\sigma,t+1|t}$;
  end
  }
end
Compute predicted mean $\hat{x}_{t+1|t}$ and covariance $P_{t+1|t}$
return Predicted mean and covariance

D UKF update step algorithm

Algorithm 2: Update step

Result: Updated mean and covariance
Data: Predicted mean $x_{t+1|t}$ and covriance $P_{t+1|t}$
Compute Sigma Points from prior $X_{\sigma,t|t}$;
for *i=1 to* $\forall$ *prior Sigma Points* **do**
  Transform through motion model {
  if $\Delta GNSS > 0$ **then**
    $X_{\sigma,t+1|t} \xrightarrow{h(x)=GNSS+IMU} Z_{\sigma,t+1|t}$;
  else
    $X_{\sigma,t+1|t} \xrightarrow{h(x)=IMU} Z_{\sigma,t+1|t}$;
  end
  }
end
Compute Updated mean $\hat{x}_{t+1|t+1}$
Compute expected measurement covariance $S_{t+1|t}$
Compute cross-correlation matrix $T_{k+1|k}$
Compute Kalman gain K
Compute Updated covariance $P_{t+1|t+1}$
return Updated mean and covariance

E Statistical outlier filter algorithm

Algorithm 3: Statistical Outlier Filter algorithm

Result: Inlier Point Cloud Set
Data: Point clouds: $P_1 = p_i, n = nneighbours, s = \sigma$
for *i=1 to* $\forall$ *points in P1* **do**
  Find the N nearest neighbour preferably through KD-Tree searching for the point i ;
  for *j=1 to N points* **do**
    Compute average distance d_i from the point i to its nearest neighbours
  end
  Compute mean of the distances $[\mu_d = \sum_{j=1}^{N} \frac{d_j}{N}]$;
  Compute standard deviation of the distances;
  Compute the threshold $T = \mu_d + \sigma_d$ **for** *j=1 to N points* **do**
    if $d_j > T$ **then**
      outlier=Delete point;
    else
      inlier=Save index in the inlier set;
    end
  end
end
return Only inlier point cloud set

Department of Mechanics and maritime sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY