공학박사학위논문

차량용 2차원 라이다를 이용한 정지 장애물 맵과 기하학적 모델 프리 접근 방식 기반 실시간 주행 환경 인식

Real-Time Driving Environment Perception based on Geometric Model-Free Approach and Static Obstacle Map using Automotive 2D LiDAR

2021년 2월

서울대학교 대학원

기계항공공학부

이 호 준

차량용 2차원 라이다를 이용한 정지 장애물 맵과 기하학적 모델 프리 접근 방식 기반 실시간 주행 환경 인식

Real-Time Driving Environment Perception based on Geometric Model-Free Approach and Static Obstacle Map using Automotive 2D LiDAR

지도교수 이 경 수

이 논문을 공학박사 학위논문으로 제출함

2020 년 10 월

서울대학교 대학원

기계항공공학부

이 호 준

이호준의 공학박사 학위논문을 인준함

2020 년 12 월

위 원 장 : 차 석 원 (인)

부위원장 : 이 경 수 (인)

위 원 : 이 창 건 (인)

위 원 : 유 송 민 (인)

위 원 : 신 동 훈 (인)

Abstract

Real-Time Driving Environment Perception based on Geometric Model-Free Approach and Static Obstacle Map using Automotive 2D LiDAR

Hojoon LEE

Department of Mechanical and Aerospace Engineering

The Graduate School

Seoul National University

This thesis proposes a real-time perception module using autonomous 2D-LiDAR and chassis sensors for urban autonomous driving. The proposed perception module divides the driving environment into static obstacles and moving obstacles, and depicts it as a static obstacle map (STOM) and tracks, respectively. The module is available any road without support from infrastructures such as V2X and HD map. Also, the module's processing time at a multi-purpose computer equipped autonomous vehicle is less than LiDAR's sensing period.

The proposed module consists of two parallel submodules. The first is the STOM construction part, which expresses static obstacles as a two-dimensional probability distribution using LiDAR, ego motion information, and tracks of the previous step. Since static obstacles are represented as a two-dimensional grid map without detection, it effectively describes various static barriers. Also, since moving obstacles are classified through interaction with the tracking module, it is more efficient than the existing iterative method.

The second sub-module is geometric model-free approach (GMFA), which uses the STOM of the previous step to classify moving parts, detects the moving obstacles by tracking, and estimates its state. After distance based clustering (DBC), tracks are created and tracked using adjacent clusters in a four-dimensional space consisting of shapes and positions in a consecutive scan. Since it uses set of points on a rigid body (SPRB), objects of various forms such as pedestrians, motorcycles, and trucks are effectively described without geometry model.

Two methods are used for estimating the dynamic state using the tracked results. The first method uses extended-Kalman filter (EKF). The displacement of the moving obstacle is extracted from matched clusters through point registration and used as a measurement of EKF to estimate the dynamic state. The method requires less computation and shows stable performance in sparse areas. However, EKF has a limit when the shape is changed dramatically, or the objects are dense. The second method uses particle filter (PF) to overcome the first method's drawback. The dynamic state is estimated using a point cloud directly. As a result, robust performance is achieved even with large shape changes or dense traffic.

The proposed perception module has been implemented in Labview /MATLAB and ROS/c++, and qualitative and quantitative analysis was performed through autonomous driving data from our autonomous vehicles. As a result of evaluating the detection performance using labeling data and the estimation performance using RT-range, the both performances are improved compared to the existing perception algorithm in urban environments.

Keywords: Autonomous vehicles, Detection and tracking of moving objects, LiDAR, Multiple objects tracking, Perception, Sparse point cloud

Student Number: 2016-23468

List of Figures

Contents

Chapter 1. Introduction

1.1 Background and Motivation

In 2010, Road injuries killed 1.33 million people globally, of which 95% were attributed to road crashes. These were the 8th-leading cause of death, and the death toll from road injuries exceeded that of malaria [Bhalla et al.'14]. According to predictive models, 1.9 million people would be injured in traffic accidents each year globally by 2020 [WHO'11], and researches show that human error accounts for 94% of all traffic accidents [Singh'15]. In Korea, according to Statistics Korea, 85.2% of all traffic accidents are caused by human factors [KOSIS'18]. To prevent the unacceptably high number of traffic accidents and road injuries caused by human factors, the research on autonomous vehicles has been actively conducted for the last couple of decades based on the improvements of the sensor, processor, and actuator technologies [AutoNet2030'14]. It is expected that AV not only prevents road injuries but also improves traffic efficiency by percept the driving environment and situation through communication and sensors and controlling the vehicle in an optimal driving method.

In the last couple of decades, Advanced Driver Assistance Systems (ADAS) and autonomous driving have made remarkable progress with improving

various automotive range sensors such as sound navigation ranging (sonar), radio detection and range (radar), camera, and Light Detection and Ranging (LiDAR). In particular, many functions of ADAS have already been commercialized by major automotive makers.

Sonar measures the range using the time of flight by sound waves [Lee et al.'10]. It had been studied and installed for Parking Distance Warning (PDW) [Gaus et al.'96] and Blind Spot Detection (BSD) [Uselmann et al.'04]. Recently, using the sensor, Autonomous Parking Assistance System (APAS) [Boehme et al.'13] has been equipped on the vehicle. The main drawback of sonar is its short field of view (FOV) around 10m [Ainslie'10].

Radar, a range sensor using radio waves, can detect multiple objects within the maximum distance with a single scan. Radar is robust against the dust, fog, rain, and brightness, and measures relative speed in the distance direction. Therefore, BSD and Adaptive Cruise Control (ACC) have been developed using radar. Radar has a wide FOV, but has the disadvantage of low resolution.[Waldschmidt et al.'14, Dickmann et al.'16]

Camera has been mounted on the vehicle as a black box for recording accident images and an Around View Monitor (AVM) for assisting the driver. As image recognition technology improved dramatically by deep learning, Forward Collision Warning (FCW), Lane Departure Warning (LDW), Lane Keeping Assist System (LKAS), Lane Change Assist (LCA), and ACC have been equipped using camera. Camera is regarded as one of the most important

sensors for autonomous driving because it could obtain most of the information for drive, and the sensor's cost is low. Despite the dramatic improvement of deep learning, camera still needs to improve their accuracy in distance measurement. In this text, camera will be called vision sensor or vision.[Geiger et al.'12, Ros et al.'15]

LiDAR, a range-sensing device using laser, is considered the most important sensor in autonomous driving because it has a higher resolution and accuracy than sonar or radar due to short wavelength. Because of this advantage, Audi improved ADAS using LiDAR. In addition, the measured points of LiDAR contain additional information about free space between sensor and measured points that due to the straightness of laser.[Kutila et al.'16, Warren'19]

Figure 1.1. Urban driving environment with LiDAR and camera.

Figure 1.1. shows the data measured by LiDAR in an actual urban road

environment with a camera. The red dots are measured by VLP-16 from Velodyne, and the white dots are measured by 2010lux from Ibeo. It can be seen that the 3D shape of a motorcycle 5m in front is measured, but the vehicles more than 40m in front are measured sparse. It seems quite clear that the only LiDAR provides enough accurate scans for urban autonomous driving. Figure 1.1, however, shows that 3D scanning using LiDAR does not provide sufficient FOV for autonomous driving. Therefore, we made an autonomous vehicle to maximize the FOV and minimize the change in the vehicle's appearance via LiDAR parallel to the ground.

1.2 System Overview

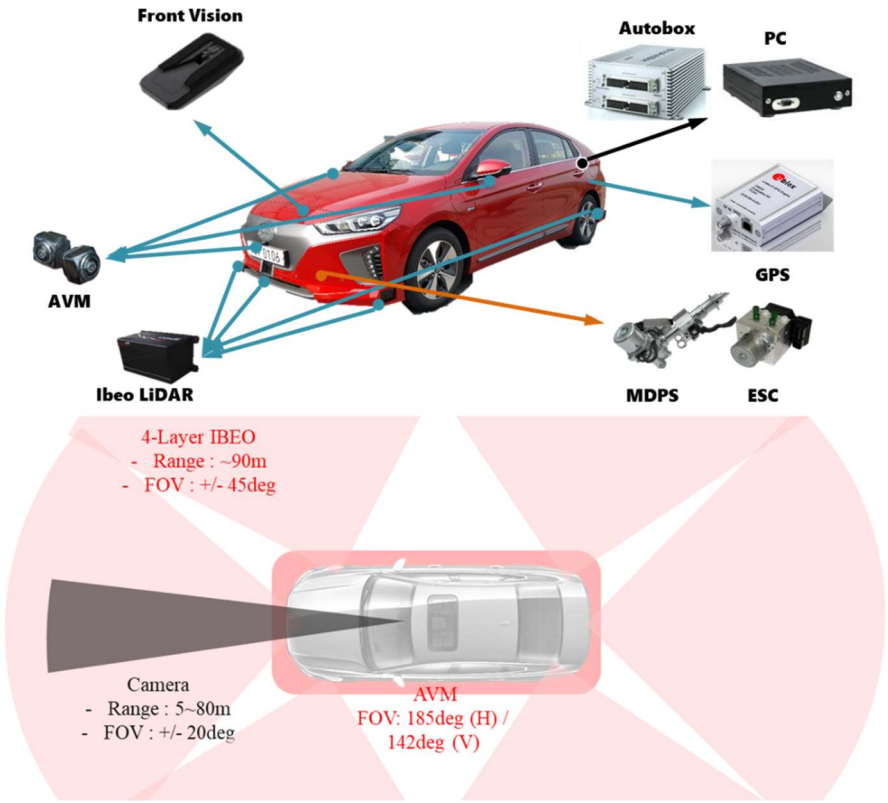


Figure 1.2. Configuration and FOV of Red Ioniq.

In this section, the test platforms to which the algorithm proposed in this dissertation is applied and validated are explained in terms of hardware and software. These platforms are designed for urban autonomous driving.

In this thesis, two Ioniqs of Hyundai Motor Company (HMC) were remodeled to autonomous vehicles and used as test platforms. The first vehicle is Red Ioniq, equipped with six 2010 lux from Ibeo, enabling real-time

omnidirectional obstacle measures. The 2010 lux can synchronize the measurement timing through a sync box and can measure at 12.5, 25 or 50Hz. In addition, Red Ioniq percepts the surrounding environment through the front camera and AVM and measures the ego vehicle's states through INS and GPS. The measured information is processed through a computer, and the computer controls the vehicle through the MDPS and ESC modules. Hardware configuration and sensor FOV of Red Ioniq are presented in Figure 1.2.

The second test platform is White Ioniq, which has three lux, four VLP-16, and five radars. Other sensors and computers are identical to Red Ioniq. Unlike lux, VLP-16 can accurately measure objects within 35m by using wide horizontal and vertical FOVs of 360 degrees and 30 degrees, respectively, and is cheaper than lux. All around of AV could be measured using four VLP-16s at four sides. For the front and rear sides of AV that require a long FOV, Lux is used to getting a sufficient sensing range for autonomous driving. VLP-16s are asynchronous, but its magnitude is less than 50ms and does not change after power on, so the sensors' asynchronous problem is not covered in this thesis. The hardware configuration and sensor FOV of White Ioniq are as shown in Figure 1.3.

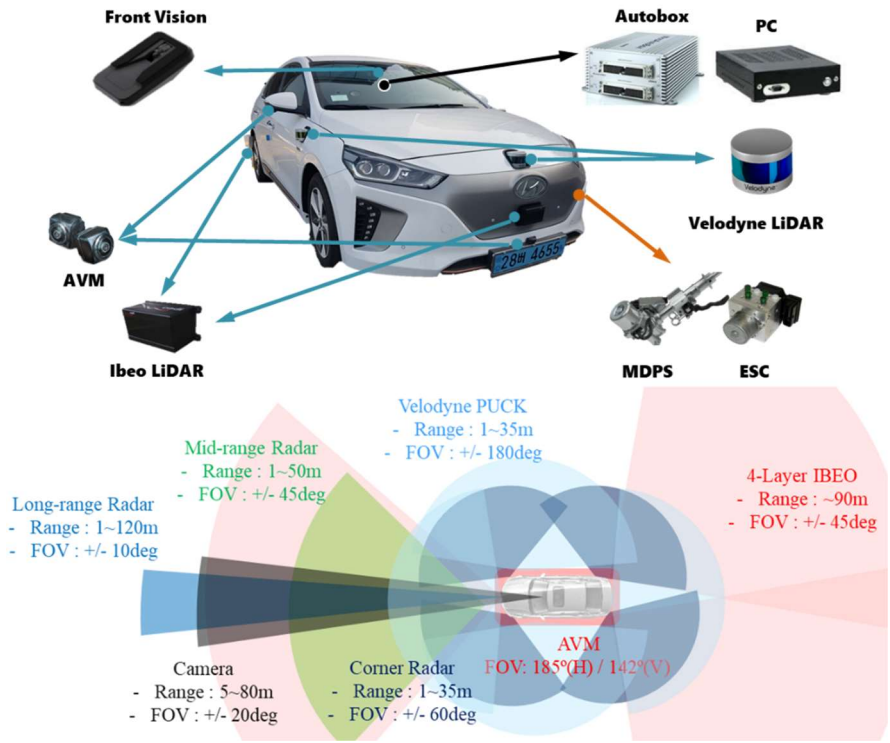


Figure 1.3. Configuration and FOV of White Ioniq.

Both autonomous vehicles used close-to-market sensors and minimized changes in the appearance of the vehicle. Besides, the computing device can be easily mounted on a vehicle using only one i7-7700 CPU. Although there are differences in the two vehicles' sensor configuration, both vehicles can apply the same algorithm because LiDAR is used as a 2D LiDAR by mounting it horizontally with the ground, and there is no blind spot on the FOV.

So far, we have explained the hardware configuration of AVs used for data collection and experimental validation. From now on, we will explain the software structure of the autonomous driving system of the control computer to

achieve the goal of the AVs. The autonomous driving system comprises four modules, from sensing data to feedback control input, as shown in Figure 1.4. Communication between the vehicle and the autonomous driving system consists of Ethernet, CAN, and Serial.

The data measured by the AV is input to localization module and perception module. The localization module estimates the states of the ego vehicle on global coordinate using the measured data, and the perception module detects objects around the ego vehicle and estimates the state of the objects. The results are passed on to the motion planning module and used to plan an optimal path to achieve the goal. The resulting optimal path and target behavior are transmitted to the control module to control the vehicle to move in the planned action.

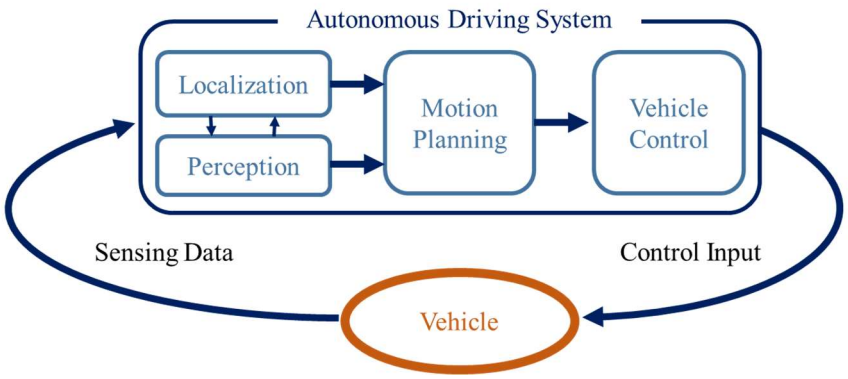


Figure 1.4. Software configuration of autonomous driving system.

1.3 Thesis Objectives and Contribution

This thesis focuses on developing a perception module that percepts the driving environment using the 2D LiDARs installed on the AV introduced in the previous section and represents the driving environment around the AV in real-time to enable urban autonomous driving. The position of the perception module in the autonomous driving system is shown in Figure 1.4. There are three requirements that the perception module should satisfiy as follows:

Modularity: The perception module should be able to operate using only LiDAR and chassis sensor of AV. The researches using High-Definition (HD) maps or V2X is actively studied, but the infrastructure is still experimental. Therefore, the perception module should be able to operate on any road with minimal information.

Real-time: In this thesis, real-time means that the perception module finishes processing within LiDAR's sensing period in the vehicle environment. If the processing time exceeds the LiDAR's sensing period, some LiDAR scan data would be skipped.

Sparse point cloud. As shown in Figure 1.1, even if 3D LiDAR is used, the 3D shape of obstacles is measured within 40m. As shown in Table 1.1, the range increases up to 60m when 32 or 64 channel LiDAR is used, but this is also insufficient for autonomous driving. In order to fully utilize LiDAR's FOV, the perception module must also operate with the sparse point cloud.

TABLE 1.1. VELODYNE AND IBEO LIDAR SPECIFICATION

	Velodyne HDL-64E	Velodyne HDL-32	Velodyne VLP-16	Ibeo Lux
Feature				
Channels	64	32	16	4
Range [m]	100-120	80-100	100	120-200
Horizontal Resolution	5Hz: 0.08° 10Hz: 0.17° 20Hz: 0.36°	5Hz: 0.08° 10Hz: 0.17° 20Hz: 0.35°	5Hz: 0.1° 10Hz: 0.2° 20Hz: 0.4°	12.5Hz0.125° 25Hz: 0.25°
Vertical Resolution[°] (Gap at 40m [cm])	**0.4 (27.9)**	**1.33 (92.8)**	**2 (139.6)**	**0.8 (55.8)**
Data Rate [pts/sec]	1,300,000	700,000	300,000	38,000

To satisfy the three requirements, the proposed perception module represents the driving environment through a static obstacle map (STOM) and tracks using point cloud within 0.5 to 2.5m above the ground and chassis information. STOM is defined as a 2D grid map, and the probability of each grid represents the chance of static obstacles occupy the grid. Moving obstacles are represented as tracks represented by a set of points on a rigid body (SPRB) and velocity, estimated through the Geometric Model-Free Approach (GMFA).

Estimating the state of moving and static obstacles become a coupled problem when the ego vehicle's state is a variable. It cannot meet the real-time requirements because iterative methodologies such as expectation-maximization are needed. Therefore, the states of the ego vehicle are assumed

fully known input in our thesis. It is also computational demanding to find the correspondence between every point in consecutive scans to estimate the obstacle's state. The correspondence estimation is accomplished through the interaction between STOM and GMFA. As a result, the driving environment is efficiently expressed in real-time.

The three main contributions of this study are as follows:

1) The interaction between STOM and GMFA leads to efficient representation of the surrounding static and moving obstacles and establishes correspondence between consecutive scans in real-time. The approach utilizes all the measured points without the region of interest (ROI).

2) The proposed approach is robust to object shape, sparse points due to far distance (>40m), and partial occlusion resulting in an increased F_1 score.

3) GMFA improves the accuracy of estimated speed and yaw angle in all scenarios, including lane change and traffic jams.

1.4 Thesis Outline

The thesis is structured in the following manner. In Chapter 2, a literature survey is conducted on the perception algorithm using LiDAR for autonomous driving. Perception algorithms are classified into tracking-based and detection-based algorithms, and each algorithm is classified again into two classes according to methodology. In Chapter 3, the entire structure of the proposed

module is described. Classifying the surrounding environment into static obstacles and moving obstacles, and tracking moving obstacles is described. Chapter 4 describes how to estimate the dynamic state of moving obstacles using tracked point clouds. The estimation method is developed into two types, an EKF-based method, and a PF-based method. In Chapter 5, quantitative and qualitative evaluation of the developed module is conducted. The detection and estimation performance are quantitatively analyzed through the reference data, and qualitative analysis is performed simultaneously to explain quantitative analysis results. The proposed module's real-time characteristics are analyzed in a limited computing environment through processing time about urban driving data. Chapter 6 summarizes the developed module's contributions and limitations, and plans future work to address the limitations.

Chapter 2. Related Work

Detection and tracking of moving objects (DATMO) using LiDAR is an important problem in several areas. Since the 2007 DARPA Urban Challenge [Urmson et al.'07, Buehler et al.'09], various methods have been proposed to solve DATMO in autonomous driving situations. Wang et al.['07] establish a mathematical framework to integrate simultaneous localization and mapping (SLAM) with generalized objects. Although the mathematical framework for SLAM with generalized objects has not changed from them, research to solve the problem is still active because, as they mentioned, the exact solution of the problem is computationally demanding and generally infeasible. For feasible solution, Wang['07] decompose the estimation problem into two separate problems and Vu et al.['07] propose an online algorithm in dynamic outdoor environments using the framework.

As a decoupled problem, DATMO with LiDAR has been studied continuously [Moras et al.'11, Börcs et al.'17, Magnier et al.'17]. To the best of our knowledge, the previous studies for DATMO can be classified into a detection-based method, *detect before track*, and a tracking-based method, *track before detect*, which are again divided into two classes according to the methodology. From this point of view, we will summarize previous studies.

2.1 Detect before Track

Detect before track framework focuses on detection rather than tracking in order to resolve DATMO. Therefore, rather than the time continuity of the LiDAR scans, it focuses on identifying objects in a single scan. The initial studies are classified as feature-based detection, which uses features expressing each cluster's shape after clustering point However, as in speech and image recognition, feature extraction by neural networks and data is more prominent than feature extraction by developers. Therefore, learning-based detection is being actively studied as well DATMO using LiDAR. From now on, a literature survey on the two detection methodologies is conducted.

2.1.1 Feature-based Detection

Feature-based detection consists of clustering and feature extraction. Most extract features after clustering, but some use feature extraction to do clustering. The latter shows better clustering performance than the former, though computational demanding. Clustering has been studied in various ways, such as distance-based clustering (DBC) [Ester et al.'96], k-means [Likas et al.'03], KF-based [Roumeliotis et al.'00, Adams'01], etc. Feature extraction also has been developed into iterative end-point fit (IEPF) [Ramer-Douglas-Peucker'72, Hershberger et al.'94], Random sample consensus (RANSAC) [Chum et al.'03, Schnabel et al.'07], and linear regression method (LRM) [Vandorpe et al.'96],

etc. Figure 2.1 represents the IEPF process at the point cloud. Feature extraction provides feasible and condensed information of the point cloud.

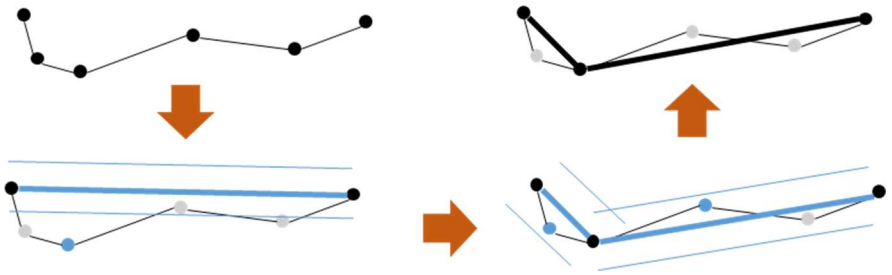


Figure 2.1. IEPF framework representation.

The clustering and feature extraction have been investigated and broadly used to solve DATMO using LiDAR for autonomous driving. Ye et al.['16] group Velodyne HDL-32E's point cloud into clusters using KF-based clustering and then extract geometry feature using IEPF. They model objects as box to describe the geometry of each moving object, and the boxes are extracted from the features. Zhao et al.['11] extract rectangular feature, including corner points and directional vectors from clusters to detect moving objects in intersection using two SICK LMSs. Ground subtraction and clustering have proceeded at each client computer, and grouping, feature extraction, and tracking are achieved at server. There have been various studies using feature-based perception and box models [Azim et al.'12, Börcs et al.'14, Cho et al.'14, Scheel et al.'16, Zhang et al.'17, Lee et al.'19].

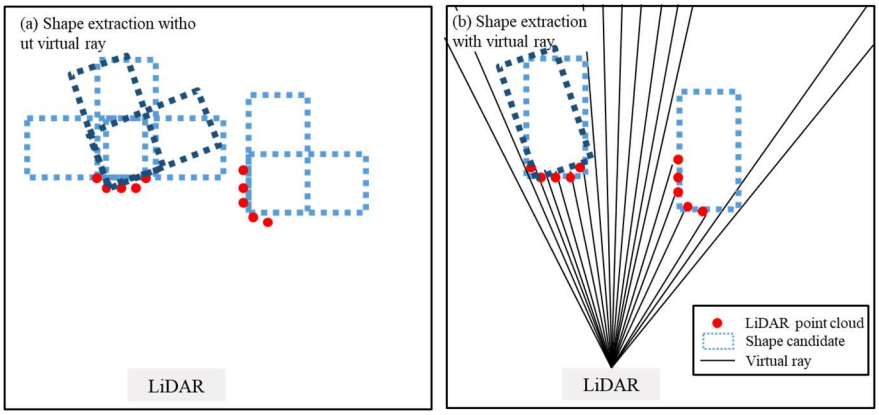


Figure 2.2. Various rectangle candidates with/without virtual ray. The darker candidates generated due to streamlined shape.

The method using the box model has various vulnerabilities. Because the object is not rectangular, the actual measurements exhibit streamlined, as shown in Figure 2.2(a). As a result, the various rectangles can be estimated through one cluster, as shown in Figure 2.2(a). Although diversity is reduced through virtual rays that take into account the straightness of LiDAR, as shown in Figure 2.2(b), it still cannot be determined uniquely. Also, more candidates are needed to track objects of various sizes. This diversity makes the algorithms computational demanding and inaccurate.

To solve various candidates, some researches represents objects to low abstracted feature: line, edge, and corner. Mertz et al.['13] segments point cloud to clusters using DBC, and then extract features from each cluster using line and corner fit. The end points of the line and the corner represent the moving object, and are tracked directly. Dominguez et al.['11] also track I-shape and L-

shape directly. Schueler et al.['12] propose multi reference point to compensate box model's drawback from different view angle and FOV. This method compensates the drawbacks of the box model, but cannot cope with the dense situation and rapid shape change. It also requires additional computation.

2.1.2 Learning-based Detection

As the feature-based detection algorithms are an early method for DATMO using LiDAR, they have a clear weakness in using clusters. Because clustering has to be preceded, the detection is impossible in dense objects. It isn't easy to detect a partially obscured object, and the classification could not be achieved. Because of this, the false positive rate is high, so researchers have tried to overcome that limitation through tracking. In the latest research, a learning-based detection methods have been introduced for human-like detection without clustering and feature extraction.

Artificial intelligence before deep learning was used to determine the optimal classification rule in a multidimensional feature space. Cheng et al.['14] propose Ring Gradient based Local Optimal Segmentation (RGLOS) algorithm to improve clustering performance and three novel features to improve detection rate. The kernel based Support Vector Machine (SVM) with the features is trained to classify vehicle from clusters. Merdrignac et al.['15] propose a supervised Adaboost trained classifier using 2D LiDAR to classify multiclass objects: vehicle, pedestrian, bicyclist, and stationary objects.

However, they assume that point cloud has been correctly segmented and consecutive observations of the same objects are correctly associated.

Thanks to the advancement of GPU and CUDA, deep learning has become a hot research topic in various fields. Chen et al.['17] propose Muti-View 3D network for sensory-fusion frameworks. They use 3D proposal network to generate 3D objects proposals from the bird's eye view representation of point cloud. As a result, the network can detect objects in dense situations because clustering does not use. He et al.['20] propose structure-aware single-stage detector and achieve state-of-the-art performance with high efficiency. In addition, a number of studies have been conducted using deep learning and KITTI benchmark [Bhattacharyya et al.'20, Pang et al.'20, Shi et al.'20, Yoo et al.'20].

Despite the breakthrough detection rate based on deep learning, the limitations associated with *detect before track* framework are obvious. Detection using point cloud of current scan leads to wrong shape via mix with a stationary object, sparse point cloud, and occlusion. In addition, since KITTI benchmark data collected using a 64-channel LiDAR, the same performance cannot be guaranteed when using close to market sensors.

2.2 Track before Detect

Human drivers recognize the surrounding environment not only as a momentary image but also as a continuous video. *Track before detect* framework finds an object through continuity over time rather than a momentary shape based on the considerations. The two frameworks are complementary rather than mutually contradictory, so they must be integrated for the ultimate perception module. The early methods of the *detect before track* framework utilized tracking to reduce false alarms [Blackman et al.'99, Petrovskaya et al.'08, Granström et al.'11]. However, in this thesis, we distinguish the two for an effective literature survey.

The framework is divided into two categories. One, geometric model-based tracking (MBT) uses the likelihood between several particles and point cloud. MBT assumes that a point cloud is spatially distributed around the target, and the probability distribution is modeled using geometric model [Gilholm et al.'05, Granström et al.'14, Wahlström et al.'15] and LiDAR model via ray-tracing [Petrovskaya'08, Scheel'16]. The other, pointwise tracking establishes the correspondence between points in successive scans using registration algorithm such as ICP [Men et al.'11, Zheng et al.'15], descriptor [Salti et al.'14, He et al.'16, Rhodes et al.'16], NDT [Zhou et al.'17, Lee et al.'20], etc., and then clustering and estimation are accomplished through the correspondence [Gu et al.'19, Jo et al.'19, Kulkarni'20]. The proposed approach imitates pointwise

tracking with efficient computation.

2.2.1 Geometric Model based Tracking

Geometric model-based tracking (MBT) models measurement using probabilistic tool: geometric model and sensor model. Figure 2.3 depicts probabilistic measurement model using geometry and sensor model. Bayes filter and measurement models are utilized to estimate the geometry and dynamic state of moving vehicles [Petrovskaya et al.'09]. The virtual ray-based measurement model and the Rao-Blackwellized particle filter (RBPF) facilitate multi-vehicle tracking without clustering or data association, and lead to simultaneous estimation of vehicle shape. However, only vehicles within 50m are detected, and cyclists with a different shape of rectangle are not detected. If the side of vehicle is not measured, error in the yaw angle estimation is occurred. They utilize the prior knowledge of the road information to initialize track and ease computation load.

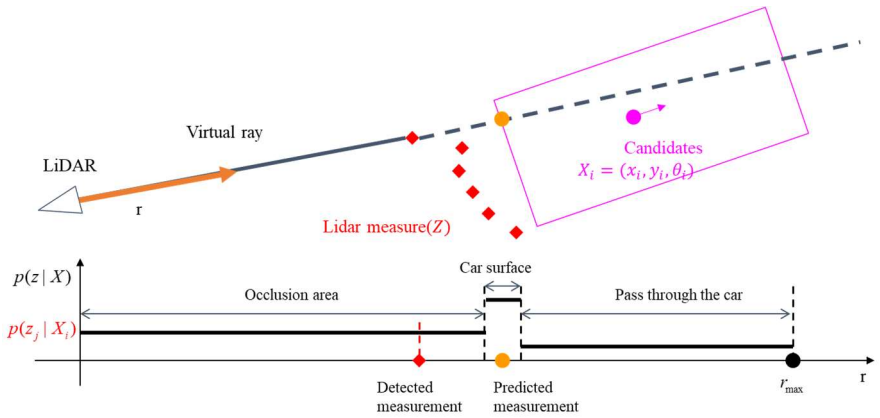


Figure 2.3. Likelihood via probabilistic geometry and sensor model.

By Liu et al.['18], the pose estimation based on coherent point drift using likelihood-field-based model is proposed and its recall and F_1 score reached 0.86 and 0.35 from KITTI benchmark dataset, respectively. However, buses or cyclists could not be detected because they assume the fixed size vehicle. The biggest drawback of the MBT is that, as seen in the previous study, various types of shape models are required to track various types of objects, which causes additional computing load and classification problems.

2.2.2 Pointwise Tracking

Pointwise tracking is the point cloud counterpart of optical flow. Gu['19] call pointwise tracking with 3D point cloud as scene flow. Pointwise tracking track each point using correspondence of successive scans. Figure 2.4 depicts correspondence of successive scans in motorway with multiple surrounding vehicles. The correspondence is estimated using SHOT descriptor [Salti'14]. By

Moosmann et al.['10], the relationship between the neighboring points of each pixel is expressed using the feature vector via local descriptor. The correspondence between the pixels of consecutive scans are established through the closest neighbor. Kaestner et al.['12] proposed the generative object detection algorithm via direct point state estimation but it run as 0.5Hz due to heavy computational load. Detection based on the correspondence is a powerful method theoretically. However, in reality, it is computational demanding to directly establish the correspondence for more than 20,000 points generated in each scan as noted Kaestner['12].

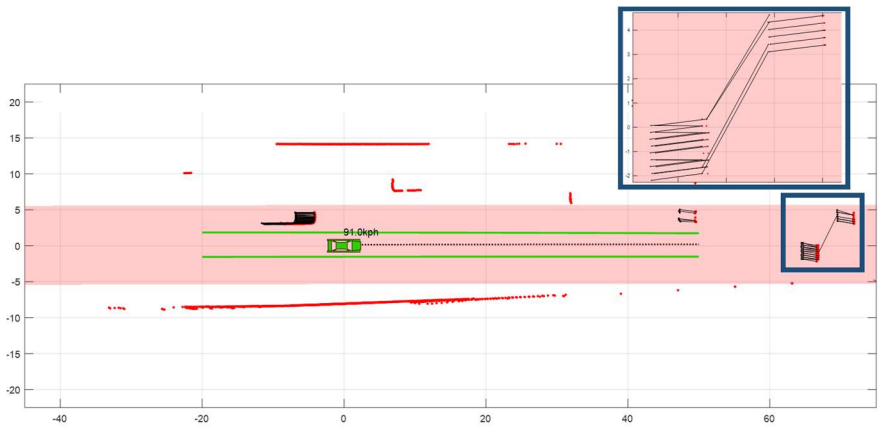


Figure 2.4. Pointwise correspondence estimation using SHOT descriptor.

In order to overcome heavy computational load, there have been attempts to detect moving objects efficiently using estimating static obstacles. Ferri et al.['15] detect moving obstacles through ray-casting on spherical voxelization and Zhong et al.['18] distinguish static obstacles through the temporal and spatial correlation. These studies only tested in low-speed robot (below 20 kph).

Wang et al.['15] use global static background map to distinguish point clouds belong static obstacles. Vaquero et al.['19] accumulate data for one second (10 frames) and track all clusters using multiple hypothesis tracking (MHT) to determine static objects. When data accumulates for a long time, it is difficult to apply it in a high-speed vehicle, because it takes a long time to determine the static objects. It is also inefficient in urban environments because it clusters all point clouds without using information previously determined to be static. For this reason, the algorithm is only tested in a low-speed truck.

The literature survey on DATMO using LiDAR for AV has been conducted as two frameworks. We can find clues on how our perception module could satisfy the requirements in a limited environment through the literature survey. As mentioned in Chapter 1, it is not the top priority to classify each type of object for autonomous driving. If the objects' position and dynamic state can be accurately estimated, it is possible to secure the autonomous vehicle's safety without classification. Also, because our testbeds are equipped close-to-market sensors, even humans cannot classify the object's type with 2D LiDAR. Since the detection-based algorithm uses the KITTI benchmark, the performance of the detection-based algorithm is not guaranteed with 2D LiDAR. In addition, the detection-based algorithm requires additional estimation to estimate the dynamic state of the moving obstacles.

For this reason, the perception module has been configured with *track before detect* framework. Our perception module represents static obstacles as STOM

and distinguishes moving obstacles through STOM. The distinguished moving obstacles are tracked through GMFA to estimate the dynamic state. Thanks to these interactions, we are able to imitate pointwise tracking while complying with real-time requirements. In addition, moving obstacles with various geometry could be tracked without classifying types by using a set of points on a rigid body (SPRB) [Granstrom et al.'16], not a geometric model.

Chapter 3. Static and Moving Obstacle Detection

This chapter deals with how to express the driving environment as static and moving obstacles through the interaction between STOM and GMFA. First of all, the problem is expressed as a mathematical formulation, and the problem is decoupled using assumptions. The structure of the perception module to solve the decoupled problem will be described, and the detailed structure of the two sub-modules will be described. The following variables are used to describe the problem.

k	: k-th time steps
$\hat{x}, \bar{x}$	: Estimation and Prediction of the variable x
x^e	: Dynamic state of ego vehicle
o^n	: States of n-th track
O	: True states of the moving obstacles
m^i	: Motion state of STOM's i-th grid (static = 1, unknown = 0)
M	: True location of the static obstacles
P^e	: Covariance of ego vehicle
P^n	: Covariance of n-th track
Y_k	: Point clouds from time 0 to k
y^o	: Measurements of moving objects
y^c	: Measurements of moving objects before tracked
y^m	: Measurements of static objects
Z	: Set of the clusters from current scan
Z^n	: Validated clusters for measurement of n-th track
S^n	: Set of points on a rigid body of n-th track
$\mathrm{T}_{\mathrm{A}\rightarrow\mathrm{B}}$	: Coordinate transformation from A to B

3.1 Interaction between Static Obstacle Map and Geometric Model-Free Approach

In this section, we describe the problem as mathematical formulation and the configuration of proposed perception module. Most generalized problem for AV in urban road is defined as SLAM with generalized objects by Wang['07]. The generalized problem can be expressed as follows:

$$p(x_k^e, G_k | U_k, Y_k) \quad \text{where}, G_k \triangleq \{g_k^1, g_k^2, \dots, g_k^n\} \qquad \text{Eq. 3.1}$$

g_k^i means the true state of the generalized object including static and moving objects. U_k represents measured input of ego vehicle. Using Bayes' rules with assumptions that no interaction among objects and 1st Markov process, the recursive formula can be derived as follows:

$$\begin{aligned} & p(x_k^e, G_k | U_k, Y_k) \\ & \propto p(y_k | x_k^e, G_k) \iint p(x_k^e | x_{k-1}^e, u_k) p(G_k | G_{k-1}) \\ & \cdot p(x_{k-1}^e, G_{k-1} | U_{k-1}, Y_{k-1}) dx_{k-1}^e dG_{k-1} \end{aligned} \qquad \text{Eq. 3.2}$$

$p(y_k | x_k^e, G_k)$ denotes measurements update, $p(x_k^e | x_{k-1}^e, u_k)$ and $p(G_k | G_{k-1})$ denotes prediction of ego vehicle and generalized objects respectively. If we use Eq. 3.2, both computation load and memory are required exponentially according to the number of objects because ego vehicle's state and surrounding objects as well as whether the object is static or moving are

updated concurrently at $p(y_k|x_k^e, G_k)$.

For the real-time algorithm, three assumptions are applied, and through this assumption, the static obstacle and the dynamic obstacle tracking are decoupled. Also, since the dynamic state of the ego vehicle is received from the previous module, the ego vehicle's dynamic state (x^e) is regarded as a known input. The mathematical formulation is based on previous study of Wang['07]. The assumptions are as follows:

① The LiDAR point cloud can be decomposed into belong static obstacles and moving obstacles.

$$y_k = y_k^o \cup y_k^m \quad \text{hence} \quad Y_k = Y_k^o \cup Y_k^m \qquad \text{Eq. 3.3}$$

It means conditional independence of point cloud of static and moving obstacles.

$$\begin{aligned} & p(y_k|x_k^e, O_k, M_k) \\ = \; & p(y_k^m|x_k^e, O_k, M_k) \cdot p(y_k^o|x_k^e, O_k, M_k) \\ = \; & p(y_k^m|M_k) \cdot p(y_k^o|O_k) \end{aligned} \qquad \text{Eq. 3.4}$$

② The moving obstacles and measurements belonging to moving obstacles carry no information about STOM and vice versa.

$$\begin{aligned} & p(O_k, M_k|x_k^e, Y_{k-1}) \\ = \; & p(M_k|x_k^e, Y_{k-1}) \cdot p(O_k|x_k^e, Y_{k-1}) \\ = \; & p(M_k|x_k^e, Y_{k-1}^m) \cdot p(O_k|x_k^e, Y_{k-1}^o) \end{aligned} \qquad \text{Eq. 3.5}$$

③ There is no interaction between the ego vehicle and the moving objects.

Also, each grid of STOM is independent of each other.

$$P(O_k|O_{k-1}) = \prod_i^n p(o_k^i|o_{k-1}^i) \quad \text{Eq. 3.6}$$

$$P(M_k|M_{k-1}, x_k^e) = \prod_i^n p(m_k^i|m_{k-1}^i, x_k^e) \quad \text{Eq. 3.7}$$

The DATMO problem can be posed as estimating the posterior probability

$$p(O_k, M_k|x_k^e, Y_k) \quad \text{Eq. 3.8}$$

where the variable $O_k = \{o_k^1, o_k^2, \dots, o_k^n\}$ denote true state of moving obstacles at time k, and variable $M_k = \{m_k^1, m_k^2, \dots, m_k^l\}$ denote true state of each STOM's grid. The recursive formula can be derived using Bayes' rule.

$$\begin{aligned} & p(O_k, M_k|x_k^e, Y_k) \\ & \propto p(y_k|O_k, M_k, x_k^e, Y_{k-1}) \cdot p(O_k, M_k|x_k^e, Y_{k-1}) \end{aligned} \quad \text{Eq. 3.9}$$

$p(y_k|O_k, M_k, x_k^e, Y_{k-1})$ does not depend on Y_{k-1} according to 1st order Markov assumption hence we have

$$\begin{aligned} & p(O_k, M_k|x_k^e, Y_k) \\ & \propto p(y_k|O_k, M_k, x_k^e, Y_{k-1}) \cdot p(O_k, M_k|x_k^e, Y_{k-1}) \\ & = p(y_k|O_k, M_k, x_k^e) \cdot p(O_k, M_k|x_k^e, Y_{k-1}) \end{aligned} \quad \text{Eq. 3.10}$$

And then, we can divide the measurement into static and moving using Eq. 3.3 and Eq. 3.4

$$
\begin{aligned}
& p(O_k, M_k | x_k^e, Y_k) \\
& \propto p(y_k | O_k, M_k, x_k^e) \cdot p(O_k, M_k | x_k^e, Y_{k-1}) \\
& = p(y_k^o | O_k, M_k, x_k^e) \cdot p(y_k^m | O_k, M_k, x_k^e) \\
& \cdot p(O_k, M_k | x_k^e, Y_{k-1}) \\
& = p(y_k^o | O_k) \cdot p(y_k^m | M_k) \cdot p(O_k, M_k | x_k^e, Y_{k-1})
\end{aligned}
\qquad \text{Eq. 3.11}
$$

By Eq. 3.5, the last term is developed as follow:

$$
\begin{aligned}
& p(O_k, M_k | x_k^e, Y_{k-1}) \\
& = \; p(M_k | x_k^e, Y_{k-1}^m) \cdot p(O_k | x_k^e, Y_{k-1}^o)
\end{aligned}
\qquad \text{Eq. 3.12}
$$

And now, the desired posterior probability can be decupled.

$$
\begin{aligned}
& p(O_k, M_k | x_k^e, Y_k) \\
& \propto p(y_k^o | O_k) \cdot p(y_k^m | M_k) \cdot p(O_k, M_k | x_k^e, Y_{k-1}) \\
& = p(y_k^m | M_k) \cdot p(M_k | x_k^e, Y_{k-1}^m) \\
& \cdot p(y_k^o | O_k) \cdot p(O_k | x_k^e, Y_{k-1}^o)
\end{aligned}
\qquad \text{Eq. 3.13}
$$

$p(y_k^m | M_k) \cdot p(M_k | x_k^e, Y_{k-1}^m)$ means STOM construction, and $p(y_k^o | O_k) \cdot p(O_k | x_k^e, Y_{k-1}^o)$ means DATMO. The prediction terms can be formulated as recursive form as follow:

$$
\begin{aligned}
& p(M_k | x_k^e, Y_{k-1}^m) \\
&= \int p(M_k | M_{k-1}, x_k^e, Y_{k-1}^m) \cdot p(M_{k-1} | x_k^e, Y_{k-1}^m) dM \\
&= \int p(M_k | M_{k-1}, x_k^e) \cdot p(M_{k-1} | x_{k-1}^e, Y_{k-1}^m) dM \\
& p(O_k | x_k^e, Y_{k-1}^o) \\
&= \int p(O_k | O_{k-1}, x_k^e, Y_{k-1}^o) \cdot p(O_{k-1} | x_k^e, Y_{k-1}^o) dO \\
&= \int p(O_k | O_{k-1}, x_k^e) \cdot p(O_{k-1} | x_{k-1}^e, Y_{k-1}^o) dO
\end{aligned}
\qquad \text{Eq. 3.14}
$$

Finally, the posterior can be expressed as recursive form.

$$
\begin{aligned}
& p(O_k, M_k | x_k^e,\ Y_k) \\
& \propto \underbrace{p(y_k^m | M_k)}_{\text{STOM update}} \\
& \cdot \underbrace{\int p(M_k | M_{k-1}, x_k^e) \cdot p(M_{k-1} | x_{k-1}^e,\ Y_{k-1}^m) dM}_{\text{STOM predict}} \\
& \cdot \underbrace{p(y_k^o | O_k)}_{\text{GMFA update}} \\
& \cdot \underbrace{\int p(O_k | O_{k-1}, x_k^e) \cdot p(O_{k-1} | x_{k-1}^e,\ Y_{k-1}^o) dO}_{\text{GMFA predict}}
\end{aligned}
\qquad \text{Eq. 3.15}
$$

From the second line in Eq. 3.15, STOM update, STOM prediction GMFA update, and GMFA prediction are expressed in order.

Using the three assumptions, the estimation of the static obstacles and the moving obstacles has been decoupled. The structure of the perception module

to solve the decoupled DATMO problem will be described. The perception module with Labview/MATLAB is implemented as a single-threaded process, and in ROS/c++ environment, it is implemented as a multi-threaded process to optimize processing time.

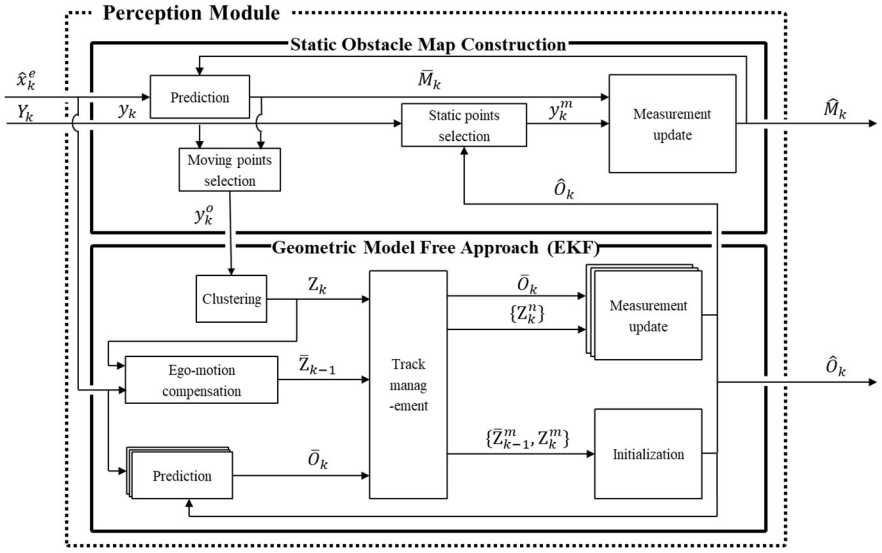


Figure 3.1. Configuration of the proposed perception module with extended-Kalman filter.

Figure 3.1 shows the structure of the perception module consisting of STOM construction and GMFA with extended-Kalman filter (EKF) to solve decoupled DATMO problem without finding correspondence between every point of the consecutive point clouds. The module estimate static obstacles position (M_k) and moving obstacles states (O_k) from ego vehicle's state $(\hat{x}_k^e)$ and time series of LiDAR point clouds $(\mathrm{Y_k})$.

The two submodules have to interact even though the problem is decoupled

because of the assumption 1. In the ssumption 1, we can distinguish between a moving object and a stationary object among all measurements, but in reality, it is impossible to distinguish between the measurements of LiDAR alone. To distinguish this, the two modules interact, as shown in Figure 3.1. y_k^o is classified from y_k using $\bar{M}_{k-1}$ and y_k^m is classified from y_k using $\hat{O}_k$. The configuration of Figure 3.1 is based on a single-threaded process. In a multi-threaded process, two modules are operated in parallel, so they interact with each other using the previous step results. As a results, y_k^o is classified from y_k using $\bar{M}_{k-1}$ and y_k^m is classified from y_k using $\bar{O}_{k-1}$.

STOM construction module calculate $\hat{M}_k$ using $\hat{x}_k^e$, $y_k, \hat{O}_k$, and $\hat{M}_{k-1}$. $M_k = \{m_k^1, m_k^2, \dots, m_k^l\}$ is represented as two dimensional grid map. Each grid of STOM represents the probability of static obstacle be in the grid $p(m_k^i = 1)$. It is 1.0 when static obstacle occupies the grid, and 0.0 when the grid is moving, free, or unknown. Unlike the occupancy grid map, ray casting is not required, and computational load is low because STOM is defined without distinguishing between unknown and free. Since STOM is defined in the local coordinate system of the ego vehicle, memory usage is low.

GMFA module calculate $\hat{O}_k$ using $\hat{x}_k^e$, $y_k, \hat{M}_{k-1}$, and $\hat{O}_{k-1}$. n moving obstacles $O_k = \{o_k^1, o_k^2, \dots, o_k^n\}$ are represented as n tracks. The track consists as follows:

$$o_k^i = [S^i,\ p^{x,i},\ p^{y,i},\ \theta^i,\ v^i,\ \gamma^i,\ a^i,\ \dot{\gamma}^i] \qquad \text{Eq. 3.16}$$

The RHS of Eq. 3.16 represents set of points on a rigid body (SPRB), mean position, yaw angle, speed, yawrate, acceleration, and yawrate dot in order. Since EKF cannot use point cloud directly as measurement, GMFA with EKF deal clusters Z^{i} generated from y_k^o via distance based clustering (DBC).

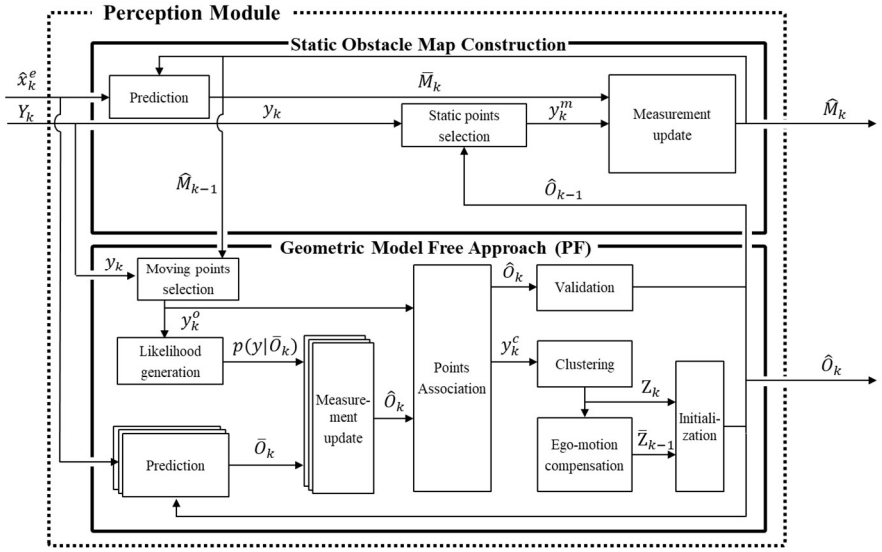


Figure 3.2. Configuration of the proposed perception module with particle filter.

Figure 3.2 depicts the configuration of perception module with particle filter (PF). As show in Figure 3.2, despite which filter is used, i/o and interaction configuration are identical. However, since the particle filter can use the point cloud directly for an update without data association, the existing track's prediction and update are the first. Clustering is used only in the initialization. More details are covered in Chapter 4.

It is theoretically correct to establish the correspondence between all points

in the successive scan using Signature of Histograms of OrienTations (SHOT) descriptor [Salti'14] or Conditional Random Fields [Van De Ven et al.'10], and then detect objects from the point cloud based on the correspondence. However, it is impossible to specify the number of iterations required for convergence, and the high computational complexity is required for practical application. In this context, the perception module estimates the states of each point and clusters the points move similar, without establishing the correspondence between every point in successive scans.

Until now, the DATMO problem has been mathematically defined and decoupled in a formulation that can solve the coupled problem in real-time through some assumptions. In addition, the overall structure, input/output, and the interaction between the internal submodules to solve the decoupled problem have been explained. The detailed design of each sub-module will be described in the following sections.

3.2 Static Obstacle Map

The problem we are trying to solve through STOM is

$$p(y_k^m|M_k) \cdot \int p(M_k|M_{k-1}, x_k^e) \cdot p(M_{k-1}|x_{k-1}^e, Y_{k-1}^m) dM$$

at Eq. 3.15. STOM proposed in our dissertation is a grid representation of the local coordinate system in the ego vehicle involving $p(m^i = 1)$, which is

different from the SLAM-based occupancy grid map (OGM) proposed in the previous study [Collins et al.'07]. Occupancy grid map has been continuously used to describe the surrounding stationary environment for autonomous driving [Bouzouraa et al.'10, Xu et al.'19]. In real-time application algorithms, 2D maps are mainly used [Kohlbrecher et al.'11, Meyer-Delius et al.'12, Sabatini et al.'18], and 3D maps [Schmid et al.'10, Berrio et al.'17, Ho et al.'18] are also being studied using octree [Jessup et al.'14, Yue et al.'16]. Despite octree's efficient representation of 3D space, 3D OGM requires too much memory and computation time. Therefore, when online algorithms deal 3D space, the extracted features or point clouds are mainly used instead of 3D OGM [Shan et al.'18, Shan et al.'20]. Therefore, OGM and STOM represents 2D grid map in this thesis. OGM classifies each grid into Occupied, Free, and Occluded, after discretizing the space in the global coordinate. Figure 3.3 presents OGM; black, dark gray, bright gray, and white represents Occupied, Occluded, Free, and LiDAR, respectively. The map’s objective is to map the unknown space and estimate the global position and orientation of the ego vehicle.

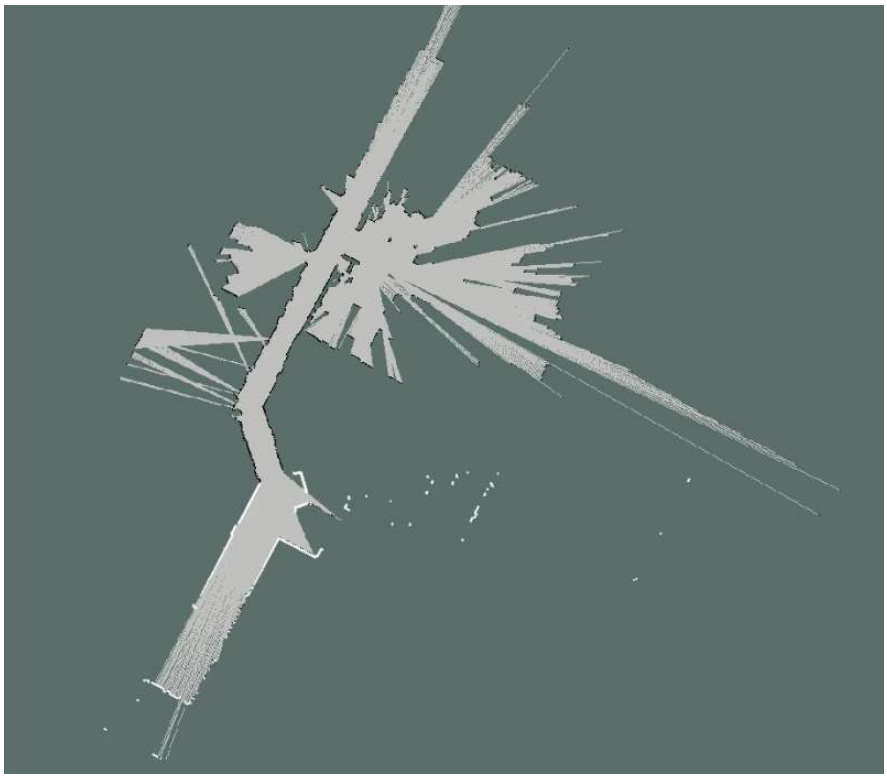


Figure 3.3. 2D occupancy grid map example. [Grisetti et al.'07]

To classifies Free and Occluded, ray casting must be needed. Ray casting is a method of modeling the laser beam of LiDAR in code, as shown in Figure 3.4. Despite well-developed line drawing library such as Bresenham's line algorithm [Bresenham'65], drawing thousands of lines on millions of grids every 50 milliseconds is a computational demanding.

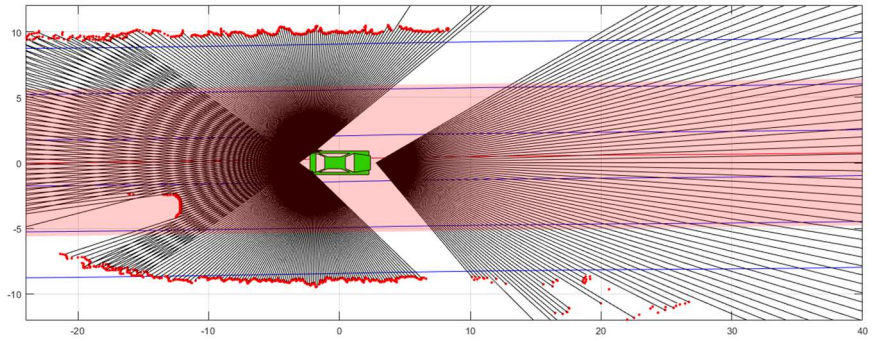


Figure 3.4. Visualization ray casting about an autonomous vehicle.

Distinguishing Free and Occluded as well as Occupied in global coordinate through the straightness of LiDAR, clearly has benefits for autonomous driving. However, with respect to safety in a limited computing environment, it is the best option to express only the occupied area in the local coordinate system considering the calculation time and memory.

STOM represents the environment around the ego vehicle locally as Static and Unknown (including Moving, Occluded, and Free) grid. STOM is proposed to distinguish the static and the moving LiDAR points, and to efficiently represent static obstacles with various shapes including accident vehicles, scaffolds, and cones.

Unlike a moving object, a stationary object can have very close distances between objects even if they are different objects. It's easy to understand when you compare moving and parked vehicles. Therefore, it is difficult even for humans to classify a stationary object using 2D LiDAR. However, from the point of view of an autonomous vehicle's safety, there is no need to distinguish

static objects. When stationary objects are represented as static obstacles in a 2D grid map, safety control is possible. STOM does not recognize each of the stationary objects as objects but expresses them as static obstacles in the 2D grid map. Therefore, even if objects of any shape exist in any form, they can be effectively represented.

Since STOM does not distinguish between free and occluded, ray casting is not required, and the computational load for prediction and update is reduced. The active cell can be tracked and updated without update all grids every time. The method may be inefficient than updating all grids depending on the number of activated cells. However, in our problem, this method is more efficient because only a few thousand grids out of the total millions of grids are active. In fact, it took 50ms to predict all grids in the ROS/c++ environment, but updating only the active cells took only a few milliseconds. Figure 3.5 show difference the two grid update method.

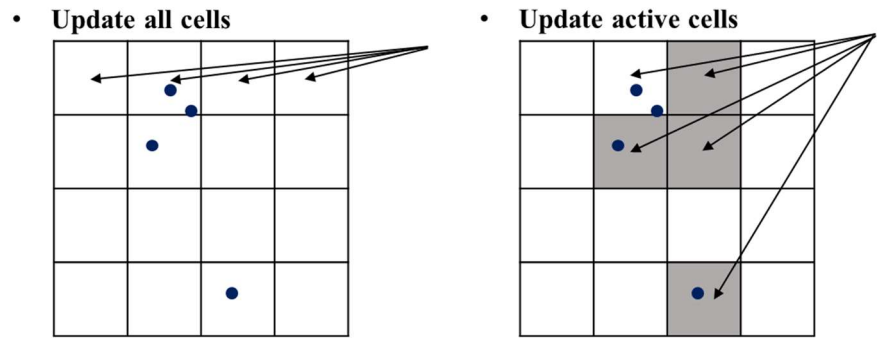
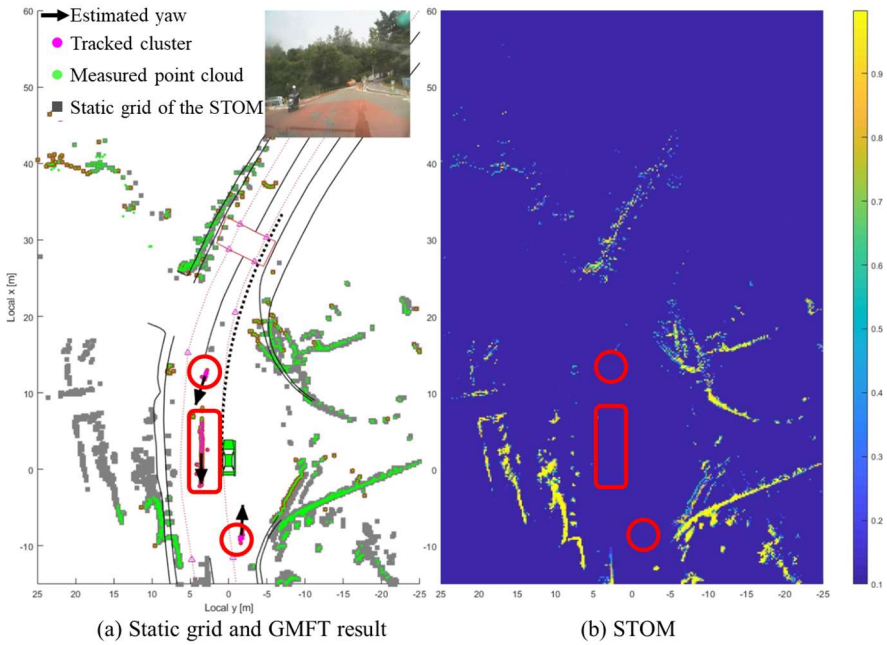


Figure 3.5. Compare all cells update vs active cells update.

From now on, we will explain the overall structure of STOM construction.

STOM construction calculate STOM at time k ($\widehat{M}_k$) using dynamic state of ego vehicle ($\hat{x}_k^e$), LiDAR point cloud (y_k), tracks ($\hat{O}_k$) and STOM at time k-1 ($\widehat{M}_{k-1}$) as Figure 3.1 and Figure 3.2. First, predict STOM ($\bar{M}_k$) using previous STOM ($\widehat{M}_{k-1}$) and dynamic state of ego vehicle ($\hat{x}_k^e$). Then, point cloud belong moving objects (y_k^o) is identified using predicted STOM ($\bar{M}_k$), and point cloud belong static objects (y_k^m) is identified using tracks ($\hat{O}_k$). Finally, STOM at time k, $\widehat{M}_k$, is calculated via Bayes' rule using predicted STOM ($\bar{M}_k$), and point cloud belong static objects (y_k^m).



(a) Static grid and GMFT result (b) STOM

Figure 3.6. Histogram representation of STOM.

Each grid of STOM bound between 0.05 and 0.95. As a result, even though the state transition matrix is an identity matrix, the probability is not saturated

to 1.0 or 0.0. Figure 3.6-(b) represents the STOM as a two-dimensional histogram, and the point cloud and tracking results are shown in Figure 3.6-(a). In these terms, the errors from moving objects are corrected via interaction with GMFA. If the points belong to a moving object next to the ego vehicle as shown in the red box in Figure 3.6-(a), the grids containing the points are considered static without interaction with GMFA, because LiDAR points are measured continuously on the same grid. The LiDAR points measure different parts of the object, but STOM does not consider whether it is the same parts. However, the points are identified as moving via interaction with GMFA, which decreases the static probability of the moving object region, as shown in Figure 3.6-(b). STOM represents a static environment efficiently and supports the detection and tracking performance of moving objects. Therefore, STOM is configured locally and predicted using only Inertial Navigation System (INS). Accordingly, it is noted that its accuracy is not the primary focus of our study.

During the initialization phase, STOM is initialized to 0.05. The static estimation of points through GMFA needs at least 2 steps, and the moving estimation based on GMFA requires at least 7 steps, so STOM is initially updated in the same manner as scan differencing.

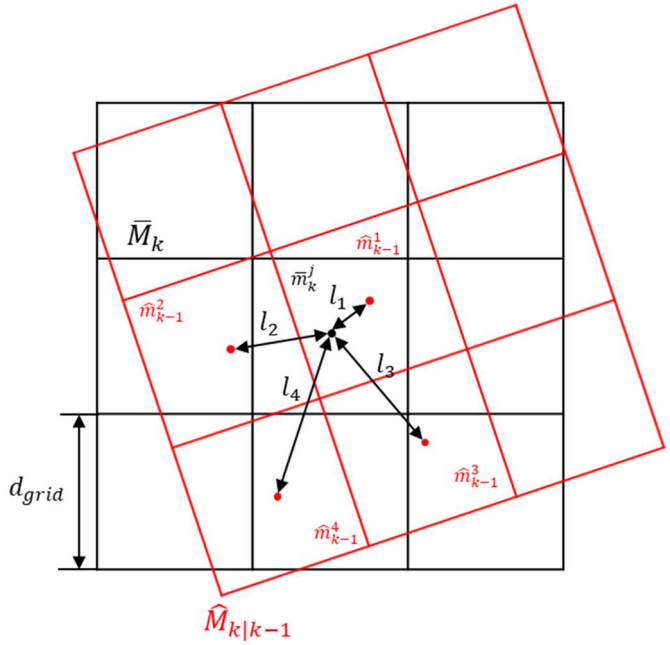


Figure 3.7. Correlation between the STOM of previous and current step.

3.2.1 Prediction of Static Obstacle Map

Since STOM represents the local environment, the ego motion in consecutive steps must be corrected through the state of the ego vehicle. In Figure 3.7, the red grid represents the previous STOM corrected via ego motion ($\hat{M}_{k|k-1}$), and the black grid refers to the current STOM ($\bar{M}_k$).

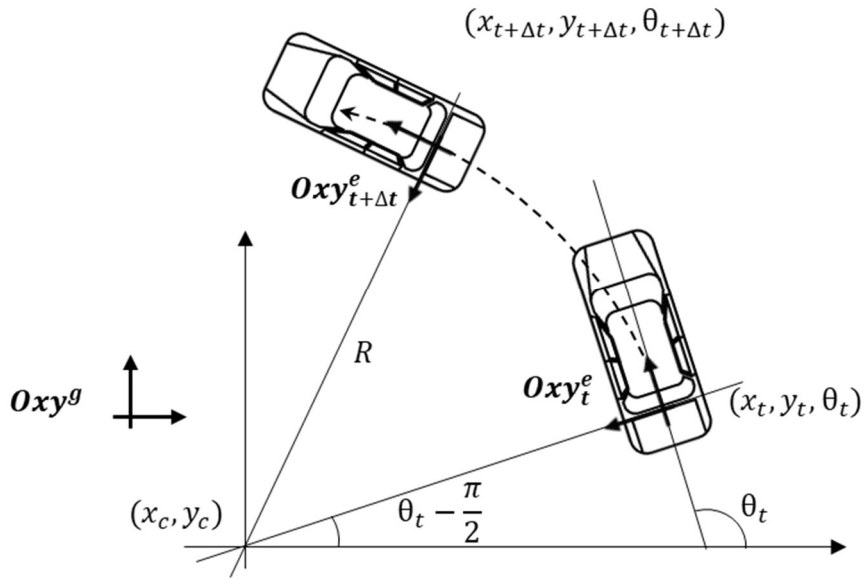


Figure 3.8. Ego vehicle motion with constant velocities $\boldsymbol{v}$ and $\boldsymbol{\gamma}$.

In order to compare STOMs at different times, as shown in Figure 3.7, it is necessary to compensate for the ego motion during the time difference (Δt) using the dynamic state of the ego vehicle ($\hat{x}_k^e = [v, \gamma]^T$). The ego vehicle motion with the constant velocity model and a point mass model is shown in Figure 3.8. $\boldsymbol{Oxy_t^e}$ and $\boldsymbol{Oxy^g}$ represent the ego vehicle's local coordinate system and the global coordinate system at time *t*, respectively. The ego vehicle coordinate system is centered at the rear axle because the location is suitable for a point mass model. Since a point mass model ignores the beta, the difference in the direction of the vehicle's heading angle and the actual velocity, the assumption is correct kinematically at the center of rear axle. The derivation of the position after Δt in the global coordinate system is as follows:

$$R = \left|\frac{v}{\gamma}\right| \quad and \quad v = \gamma \cdot R \qquad \text{Eq. 3.17}$$

The constant velocity model makes a circular motion, and at that time, the radius of the circular trajectory is derived as Eq. 3.17. The center of the circular motion

$$x_c = x_t - \frac{v}{\gamma}\sin(\theta_t)$$
$$y_c = y_t + \frac{v}{\gamma}\cos(\theta_t) \qquad \text{Eq. 3.18}$$

Hence, the ego vehicle's position and yaw at $t + \Delta t$ is

When $\gamma \neq 0$,

$$x_{t+\Delta t} = x_t - \frac{v}{\gamma}\sin(\theta_t) + \frac{v}{\gamma}\sin(\theta_t + \gamma\Delta t)$$
$$y_{t+\Delta t} = y_t + \frac{v}{\gamma}\cos(\theta_t) - \frac{v}{\gamma}\cos(\theta_t + \gamma\Delta t)$$
$$\theta_{t+\Delta t} = \theta_t + \gamma\Delta t \qquad \text{Eq. 3.19}$$

If $\gamma = 0$,

$$x_{t+\Delta t} = x_t + v\cos(\theta_t)\Delta t$$
$$y_{t+\Delta t} = y_t + v\sin(\theta_t)\Delta t$$
$$\theta_{t+\Delta t} = \theta_t$$

Therefore, the transformation matrix between $\boldsymbol{Oxy_t^e}$ and $\boldsymbol{Oxy_{t+\Delta t}^e}$ as follow:

$$
\mathrm{T}_{\boldsymbol{Oxy_{t+\Delta t}\rightarrow Oxy_t}}
$$

$$
= \begin{bmatrix} \cos(\gamma\Delta t) & -\sin(\gamma\Delta t) & \frac{v}{\gamma}\sin(\gamma\Delta t) \\ \sin(\gamma\Delta t) & \cos(\gamma\Delta t) & \frac{v}{\gamma} - \frac{v}{\gamma}\cos(\gamma\Delta t) \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathrm{T}_{\boldsymbol{Oxy_t\rightarrow Oxy_{t+\Delta t}}} \qquad \text{Eq. 3.20}
$$

$$
= \begin{bmatrix} \cos(\gamma\Delta t) & \sin(\gamma\Delta t) & -\frac{v}{\gamma}\sin(\gamma\Delta t) \\ -\sin(\gamma\Delta t) & \cos(\gamma\Delta t) & \frac{v}{\gamma} - \frac{v}{\gamma}\cos(\gamma\Delta t) \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathrm{T}_{\boldsymbol{Oxy_t\rightarrow Oxy_{t+\Delta t}}} = \mathrm{T}^{\boldsymbol{-1}}_{\boldsymbol{Oxy_{t+\Delta t}\rightarrow Oxy_t}}
$$

Eq. 3.20 is also derived through Figure 3.9. Substituting $-\Delta t$ into Δt at $\mathrm{T}_{\boldsymbol{Oxy_{t+\Delta t}\rightarrow Oxy_t}}$ yields $\mathrm{T}_{\boldsymbol{Oxy_t\rightarrow Oxy_{t+\Delta t}}}$, and $\mathrm{T}_{\boldsymbol{Oxy_t\rightarrow Oxy_{t+\Delta t}}}$ product $\mathrm{T}_{\boldsymbol{Oxy_{t+\Delta t}\rightarrow Oxy_t}}$ is identity matrix**.** As a result, we can confirm Eq. 3.20 is correct. We define $\mathrm{T}_{\boldsymbol{e}} \triangleq \mathrm{T}_{\mathbf{Oxy_t\rightarrow Oxy_{t+\Delta t}}}$ for readability in this thesis. $\mathrm{T}_{\boldsymbol{e}}$ denote transformation from previous local coordinate ($\boldsymbol{Oxy^e_{t-\Delta t}}$) to current local coordinate ($\boldsymbol{Oxy^e_t}$) for static obstacle. Using $\mathrm{T_e}$, Two STOMs at different times are able to be compared as Figure 3.7.

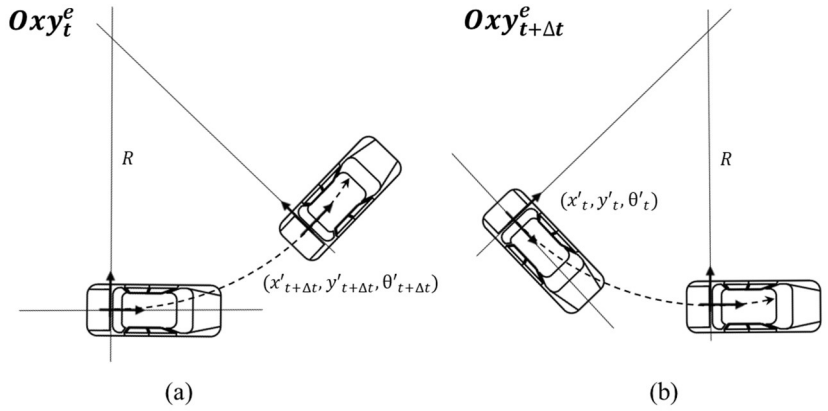


Figure 3.9. Dead reckoning from local coordinate at time $\mathbf{t}$ and $\mathbf{t} + \Delta\mathbf{t}$.

When calculating the probability of the j-th grid in the current STOM ($\bar{m}_k^j$), the four grids of the previous STOM $\left(\widehat{M}_{k|k-1}\right)$ can be determined according to the distance from the center of the current j-th grid, l_i $[i = 1,2,3,4]$, less than $\sqrt{2}d_{grid}$. Therefore, the predicted probability of the j-th grid is calculated by the weighted average of the probability of the four grids whose distance is less than $\sqrt{2}d_{grid}$ as shown in Eq. 3.21.

$$L = \sum_{i=1}^{4} l_i^{-1}$$

$$p\left(\bar{m}_k^j\right) = \begin{cases} \sum_{i=1}^{4} \frac{l_i^{-1}}{L} p\left(\hat{m}_{k-1}^i\right) & (l_i \neq 0) \\ p\left(\hat{m}_{k-1}^i\right) & (l_i = 0) \end{cases} \quad \text{Eq. 3.21}$$

If $l_i = 0.0$, the j-th grid is physically equivalent to the previous grid, such that the probability of the j-th grid reflected the probability of the previous grid.

It is assumed that each grid is independent and transition matrix is identical

when prediction. The lack of transition probability between states is based on the absence of valid state transition matrix because the physically represented position by each grid changes by the motion of the ego vehicle. Since, however, the probability of each grid is bounded, it can be estimated accurately via measurement update despite the lack of state transition.

3.2.2 Update of Static Obstacle Map

In this section, measurement update of STOM and interaction STOM and GMFA are explained. The measurement update means to updating the probability of a stationary object in each grid of the predicted STOM ($\bar{M}_k$) via the point cloud and tracks.

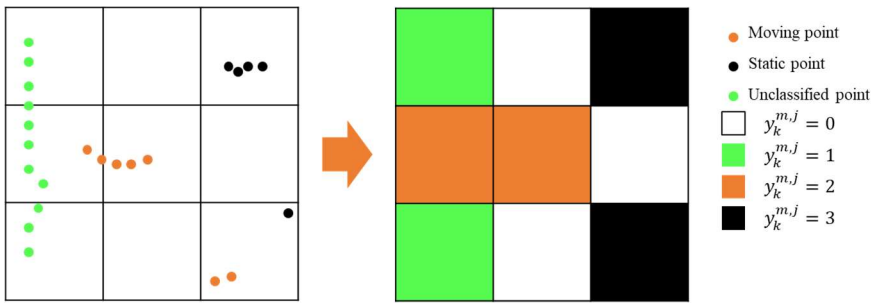


Figure 3.10. Measurements for each grid of STOM according to tracks and the point cloud.

According to the motion state of the points included in the j-th grid (m^j), the measurement of j-th grid ($y^{m,j}$) is determined as shown in Figure 3.10. One of the following four values can be used to measurement of the j-th grid: Free, 0; Unclassified, 1; Moving, 2; and Static, 3. The measurement is 3 for static

points, 2 for moving points without static points, 1 for only unclassified points, and 0 for any measure. As shown in Figure 3.10, the orange and black dots indicate moving and static points, respectively, classified by tracks form GMFA ($\hat{O}_{\mathrm{k}}$). Since each track contains set of points on a rigid body (SPRB) and speed, the motion state could be determined. In GMFA with EKF, only track's current speed and tracking age are used to determine the motion state. In GMFA with PF, tracks trajectory of speed is also used. The green dot indicates that the unclassified state of motion. When the points on each grid are similar to the left, the motion measurement of each grid is determined as shown on the right side. The white, green, orange, and black colors on the grid represent free, unclassified, moving, and static motion measurements, respectively. The measurement model is valid because the grid is small enough that the motion of each grid can be expressed as a single state. The grid size (d_{grid}) is 0.2m and 0.1m at Labview/Matlab and ROS/c++ respectively.

$$
\begin{aligned}
& p(\hat{m}_k^j = 1) \\
& \quad = p(m_k^j = 1 | y_k^{m,j}, y_{k-1}^{m,j}, \dots, y_0^{m,j}) \\
& = \frac{p(y_k^{m,j} | m_k^j = 1) p(m_k^j = 1 | y_{k-1}^{m,j}, \dots, y_0^{m,j})}{\sum_{i=0,1} p(y_k^{m,j} | m_k^j = i) p(m_k^j = i | y_{k-1}^{m,j}, \dots, y_0^{m,j})} \\
& \quad = \frac{p(y_k^{m,j} | m_k^j = 1) p(\bar{m}_k^j = 1)}{\sum_{i=0,1} p(y_k^{m,j} | m_k^j = i) p(\bar{m}_k^j = i)}
\end{aligned}
\qquad \text{Eq. 3.22}
$$

After each grid is measured as described above, the motion state of each grid can be estimated using the Eq. 3.22 with 1st order Markov assumption via the predicted STOM and the likelihood of measurement. The likelihood is predetermined as Table 3.1. It is tuned through the actual data stream.

TABLE 3.1 LIKELIHOOD FOR STOM UPDATE

m^j \ $y^{m,j}$	Free (0)	Unclassified (1)	Moving (2)	Static (3)
Unknown (= 0)	0.30	0.14	0.33	0.23
Static (= 1)	0.15	0.47	0.01	0.37

The important characteristics for determining likelihood are as follows: If a moving obstacle is being tracked, ensure that the STOM does not represent the moving obstacle as a static obstacle. After an obstacle enters into FOV, it should appear on the STOM or track within three steps. Ensure there is no more than one step that an obstacle is excluded both STOM and tracks. The characteristics are affected not only by the likelihood of STOM but also by the tracking parameters of GMFA, so the parpmeters must be tuned simultaneously.

3.3 Geometric Model-Free Approach

This section describes Geometric Model-Free Approach track creation, initialization, tracking, association, and deletion process. The creation, initialization, and deletion tracks are same, but the association between tracks and measurements is different depending on the filtering method. Therefore,

GMFA with EKF be explained this section, and the track association process of GMFA with PF will be explained in Section 4.2. The updating the track and estimating the state from the tracked result is covered in Chapter 4.

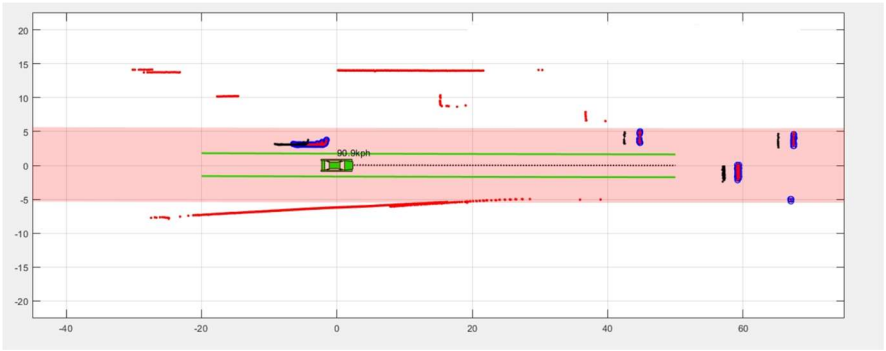
GMFA uses Y_k^o to track the moving obstacles and estimate their states. Thus, it is possible to construct the correspondence between non-static points in the consecutive scan and to update the STOM based on the motion state of each point. In our approach, compared with previous studies, each point depended on clustering using Euclidean distance. Since the correspondence between points is determined by the distance through the mean points and the shape of the cluster, the correspondence between points in consecutive scans can be established with a small calculation.

The problem we are trying to solve through GMFA is

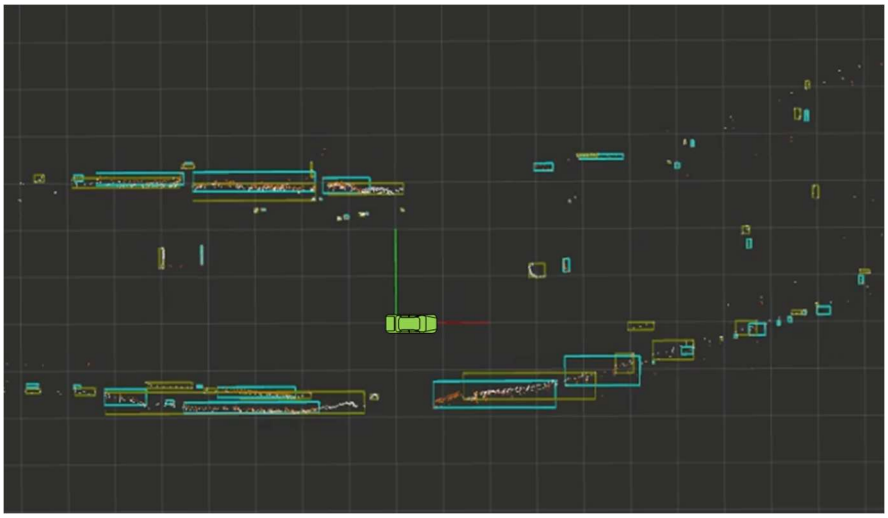
$$p(y_k^o|O_k) \cdot \int p(O_k|O_{k-1}, x_k^e) \cdot p(O_{k-1}|x_{k-1}^e, Y_{k-1}^o) dO$$

at Eq. 3.15. In the first step to this problem, there is no predicted track, $p(O_{k-1}|x_{k-1}^e, Y_{k-1}^o)$, so there are only present and past point clouds (y_k^o, y_{k-1}^o). The past point cloud can be presented at current local coordinate $(\boldsymbol{Oxy_k^e})$ as Figure 3.11 through a transformation matrix $(\mathrm{T}_{\boldsymbol{e}})$ defined Eq. 3.20. In Figure 3.11, the green car represents ego vehicle, and the driving scene is expressed in bird's eye view at Labview and ROS. In Figure 3.11-(a), red and black dots denote y_k and $y_{k-1|k}$ respectively, and blue background denote clustering results. The time interval between consecutive scans is 80ms. In Figure 3.11-

(b), y_k and $y_{k-1|k}$ are denoted white and orange dots, and yellow and blue box represents distance based clustering (DBC) results of time k and $k-1$. The time interval between consecutive scans is 1000ms at (b) for visualization.



(a) Labview



(b) ROS

Figure 3.11. The problem situation for GMFA with driving data at Labview and ROS.

If there are no clustering results in the Figure 3.11, we have only two point clouds and we cannot even know how many objects are in them. Previous

studies detect objects via the shape of a point cloud or track points via correspondence or hypothesis in this situation. However, as pointed out in Chapter 2, the former is difficult to detect with 2D LiDAR, and latter is also difficult to establish a point-by-point correspondence. Therefore, previous algorithms require a lot of computation time. In order to develop a perception module that satisfies the requirements, we focus on similarity of the DBC results in successive scan. In Figure 3.11, it can be seen that the cluster of the same object is measured in a similar shape at a nearby position.

In order to estimate the number of objects, there have been studies that repeatedly performed k-means clustering while increasing *k* [Na et al.'10], but this is inappropriate for a real-time algorithm because it repeats the iterative algorithm until the number of objects is found. Clustering with a simple DBC can lead to large errors in data at a moment. However, while continuously tracking, points of the same motion converges into a cluster.

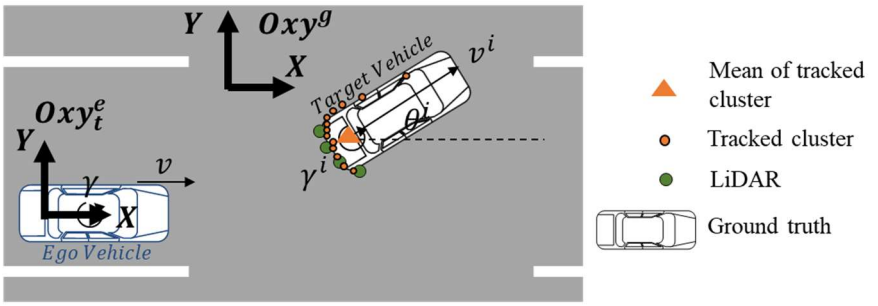


Figure 3.12. Definition of target's states for GMFA.

GMFA uses two coordinates: $\boldsymbol{Oxy^g}$, which is a fixed global coordinate system, and $\boldsymbol{Oxy_t^e}$, which is a local moving coordinate system at time *t* that

moves with the rear axle of the ego vehicle (Figure 3.12). There are eight states, $o_k^i = [S^i,\ p^{x,i},\ p^{y,i},\ \theta^i,\ v^i,\ \gamma^i,\ a^i,\ \dot{\gamma}^i]$, which express the *i*-th track. $p^{x,i}$, $p^{y,i}$ represent the mean position of SPRB (S^i) with respect to $\boldsymbol{Oxy_t^e}$. After completing the measurement update at every step, it is replaced with the new mean point when the SPRB configuration changes. θ^i denotes the yaw angle of the moving object with respect to $\boldsymbol{Oxy_t^e}$. v^i indicates the speed in the direction with respect to $\boldsymbol{Oxy^g}$. γ^i, a^i, and $\dot{\gamma}^i$ indicate the yaw rate, the acceleration, and the angular acceleration with respect to $\boldsymbol{Oxy^g}$, respectively. v and γ represent speed and yaw rate of ego vehicle at $\boldsymbol{Oxy^g}$ respectively. The SPRB (S^i) is in a queue format and points accumulated more than four steps have been removed as Figure 3.13. Z_k^i denote a point cloud associated track *i* at time *k*.

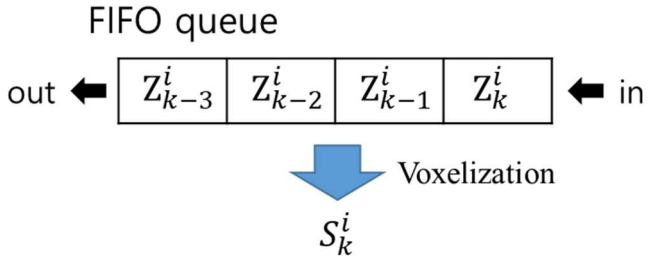


Figure 3.13. SPRB data configuration for i-th track using FIFO queue.

3.3.1 Track Initialization

So far, the overall structure of GMFA, the problem situation, and the track's data structure have been described. From now on, we will explain how the GMFA creates and initializes tracks from two point clouds, as shown in Figure 3.11.

As explained earlier, GMFA detects and tracks an object by utilizing the fact that the points measuring a moving object are isolated in 2D space and has a similar shape in successive scans. When a point cloud belongs to moving obstacles is input through STOM, the point cloud is clustered by DBC. To remove noise, only clusters with more than three points are used. The distance threshold for clustering increases as the distance from the ego vehicle increases. The previous step clusters are corrected to the current time using the stop assumption through $\mathrm{T}_{\boldsymbol{e}}$ (Eq. 3.20). In ROS, because the cluster is expressed as an index vector for the point cloud, the entire previous point cloud is transformed through $\mathrm{T}_{\boldsymbol{e}}$. This process is expressed as clustering and ego motion compensation in Figure 3.1. the present and previous clusters are denoted by Z_k and $\bar{\mathrm{Z}}_{k-1}$, respectively.

In order to create a track, it is necessary to find the relationship between clusters through distance and shape in two cluster sets (Z_k and $\bar{\mathrm{Z}}_{k-1}$). For this, a distance considering the cluster's shape and 2D Euclidean distance must be defined first. In order to define the distance, we defined the following four-

dimensional feature space.

$$f \triangleq [x, y, \lambda_{MAX}, \lambda_{min}]^T$$

$$\text{When, } [x, y] = mean(\mathrm{Z}_k^i) \qquad \text{Eq. 3.23}$$

$$[\lambda_{MAX}, \lambda_{min}] = eig(cov(\mathrm{Z}_k^i))$$

The feature vector (f) is a 4D real vector consisting of the cluster's (Z_k^i) mean point and eigenvalues of cluster's covariance matrix. The eigenvalues represent the long and short axis of the ellipse when the cluster is fitted to Gaussian distribution. The shape of cluster in a 2D space is expressed invariant to the rotation by eigenvalues. Invariant to the rotation denotes that if the relative distribution of the points is the same, the value does not change even if the entire point rotates or moves with a rigid body motion. The meaning of each feature is expressed through Figure 3.14. The orange dots denote points of the cluster; orange triangle represents the cluster's mean point. The ellipse means fitted Gaussian distribution.

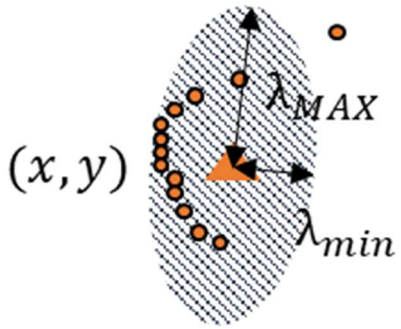


Figure 3.14. The graphical meaning of mean point and eigenvalues.

In the 4D feature space, distance is defined as a weighted 2-norm. Using the

distance, the correspondence between the two groups of clusters (Z_k and $\bar{\mathrm{Z}}_{k-1}$) is established through global nearest neighbor (GNN) algorithm. Among the correspondences established in this way, the two clusters, $\{\bar{\mathrm{Z}}_{k-1}^m, \mathrm{Z}_k^m\}$, which distance is less than the predetermined threshold are identified as continuous clusters for an new obstacle. In order to prevent a track from being created in already tracked clusters, the cluster association for the previous tracks takes precedence over the track creation process.

Tracks (o_k^m) are created through established correspondence between Z_k and $\bar{\mathrm{Z}}_{k-1}$. However, initialization is needed for states of the created tracks ($o_k^m = [S^m,\ p^{x,m},\ p^{y,m},\ \theta^m,\ v^m,\ \gamma^m,\ a^m,\ \dot{\gamma}^m\ ]$). Through the track creation process, we found black and red dots in Figure 3.15. To set the track's initial value through these two clusters, we used point registration via ICP. If the cluster of the previous scan is matched to the cluster of the current scan through ICP, it becomes a blue dot. If the mean point moves from the black triangle to the blue triangle, the displacement becomes $\vec{d}$. The direction and magnitude of $\vec{d}$ divided by time differance are set as the initial values of yaw angle (θ^m) and speed (v^m), respectively. In this way, ghost motion due to shape change can be prevented.

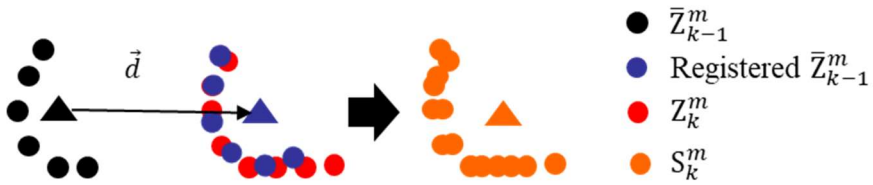


Figure 3.15. Track initialization using point registration.

$\bar{Z}_{k-1}^{m}$ and Z_k^m are combined and voxelized to initialize S^m. $p^{x,m}$ and $p^{y,m}$ is initialized with the mean point of S^m. All other values are initialized to 0. When the speed is very slow, the mean point's displacement is small, so the yaw angle, θ^m, is greatly shaken even with small noise. Therefore, if the displacement is less than 5kph, the yaw angle is initialized to 0.

3.3.2 Track Management

When a track is created through the track initialization, obstacles are continuously tracked through the association between the predicted tracks and the measured cluster. The tracks must be properly discontinued when the obstacles disappear. If this process is inappropriate, unnecessarily many tracks may be created, false alarms may increase due to afterimages, or obstacles may not be continuously tracked. The clusters (Z_k) are assigned to each track $\{Z_k^n\}$ through track management as Figure 3.1.

For the association between track and measurement, the prediction of the track is preceded. The track is predicted using a constant acceleration model in EKF and a constant velocity model in PF. S^m can be properly converted only

when its center is predicted. In principle, it is necessary to estimate the center of mass of the tracked moving object, but since we represent the object as an obstacle, we assumed the mean point of S^m as the center of the yaw rate. Because the mean point of S^m is different from the actual center of the object, the shape may be distorted. However, this effect is negligible because it is less than 100ms, and S^m keeps the cluster only four steps.

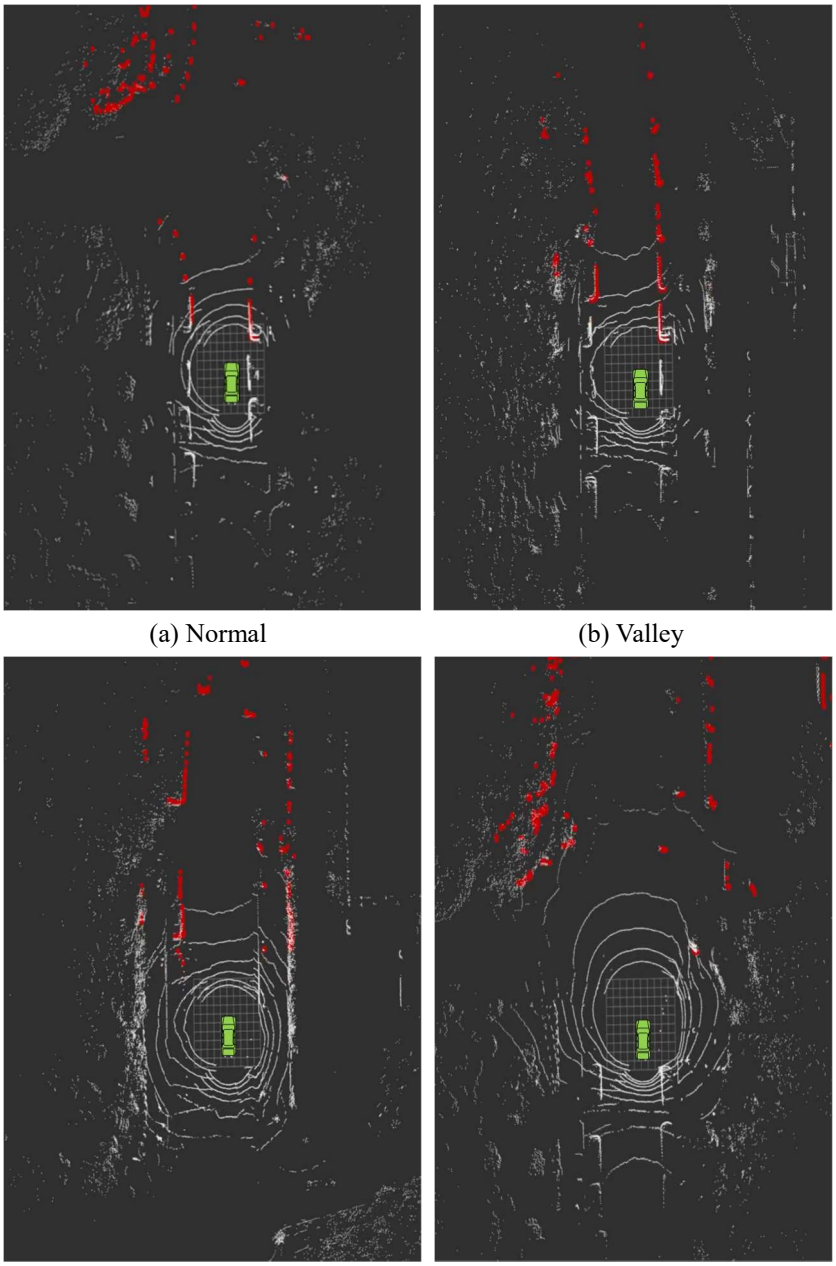
The predicted S^m and cluster are associated through GNN in the 4D feature space, as in initialization. At this time, to prevent the iterative algorithm, we associate the nearest cluster within the threshold to track in order of a lifetime. The threshold smaller than the one used at initialization since the predicted S^m and the measured cluster is closer than at the initialization.

In order to delete a proper track, we check how long it has been tracked consistently. If the track has been tracked for a long time and is not measured once over every two steps, the track is less reliable than the track has been steady tracked even though only four steps. To apply this idea, we manage the confidence index for each track. This confidence index is initialized to 2 when the track is created and increases by one each time it is associated with a measurement. This increase is held until the maximum bound (50) is reached. Therefore, the confidence index can basically be understood as the number of steps the track has been tracked. If no measurements are associated with the track, the index is deducted by 30%. This index consists of 0.5 units, and if it is

less than 8, it is decreased by 3. If this value is less than 2, the track is deleted. These values are tuned through actual driving data. The important point in tuning is that if a track is deleted too quickly, the track disappears even if a measurement is missed in one or two steps, and if it is deleted too slowly, false alarms and afterimage increase.

3.4 Ground Rejection

Ground rejection is an important research field for autonomous driving through LiDAR. The reason can be seen in Figure 3.16 (a). In Figure 3.16, the red dot is the measured value of 2010lux, the white dot is the measured value of VLP-16 HiRes, and the green vehicle is the ego vehicle. Except for the objects we are looking for, such as trees, vehicles, and poles, we can see that there are measurements in the shape of concentric circles around the ego vehicle. This is the result of measuring the ground from the downside layer. we can see from Figure 3.16, a number of measurements measure the ground, and these points must be removed so that we can find the object we are looking for. If the ground is perceived as an object, the autonomous vehicle can never go forward.



(a) Normal (b) Valley

(c) Uphill (d) Downhill

Figure 3.16. The ground appearance according to slope measured through four VLP-16 HiRes.

Therefore, research for ground rejection has been actively conducted in the last two decades. There have been various previous studies using RANSAC [Konolige et al.'09, Li'19], point registration [Lam et al.'10, Douillard et al.'11], learning [Pomares et al.'18, Velas et al.'18], reflectance [Yunfei et al.'08, Li et al.'15], and local shape using planar models [Shan et al.'05, Tongtong et al.'11, Wu et al.'19]. Most of this research assumes most points are located on a specific plane and find that plane. However, in the actual driving environment, the road consists of an arbitrary number of planes, and there are many roads that are difficult to fit as a plane, as Figure 3.17. In Figure 3.17, each color means a plane of the same slope that makes up the road. In fact, the Figure 3.16 (b)-(d) shows how the ground measurement differs from the normal situation in each driving environment. In addition, the broking planer assumption occurs instantaneously when the road's slope around the AV changes rapidly due to bumping or joining the road. Therefore, the estimation of the number of planes and the slope must be made instantaneously using a single scan.

In order to find the number of planes, an iterative fitting algorithm like clustering of a random number must be repeatedly performed while increasing the number of planes, and convergence is not guaranteed. Besides, the algorithm using plane fitting assumes that the data measuring ground occupies most number of the total data. Therefore, it is difficult to erase the measured ground between vehicles in a traffic jam because the surrounding vehicles are measured more than the ground. The problems are caused by finding the ground,

not the plane, assuming a plane. Due to the invalid assumption, the algorithms do not work well in the real driving environment, even if an advanced algorithm is used. Therefore, we used the simplest method; the height value of the measurements is used to classify and delete ground. In this case, false-positive occurs in front of the ego vehicle instantly when the road slope changes or when the bump is passed, but the calculation time is close to zero and the false positive also disappears after 1 or 2 steps.

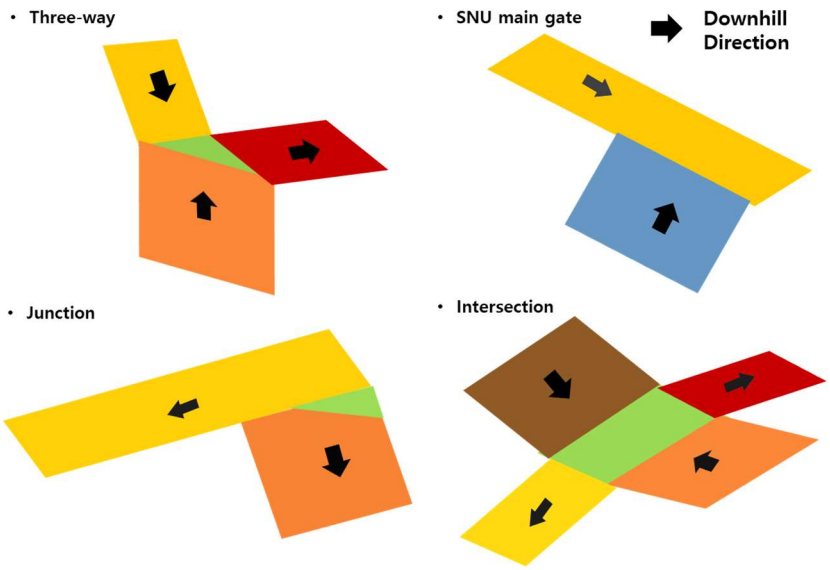
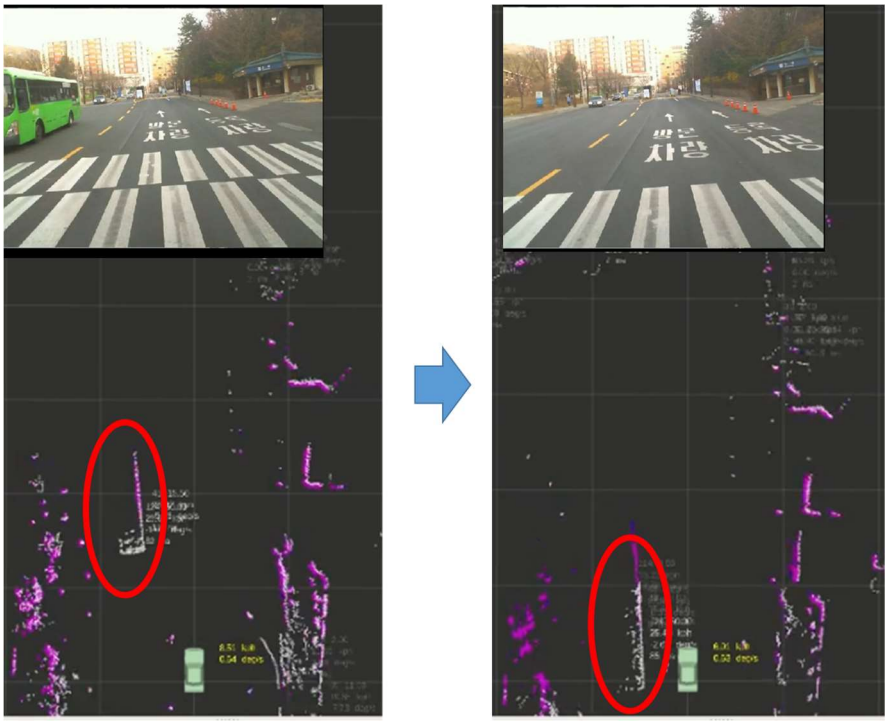


Figure 3.17. Road shape consisting of various numbers of planes.

So far, in Chapter 3, we have seen how the proposed perception module is divided into two sub-modules, how the sub-modules interact, and how each sub-module percepts moving and static obstacles using a straight-forward algorithm. The sub-modules are configured as simple as possible and without

iteration. Also, the sub-modules can be configured independently or sequentially, depending on the environment. As a result, the proposed perception module can process all measurements in real-time without ROI selection.

Due to this simple operation, strict recognition cannot be guaranteed in one step. For example, before being tracked, the bus's side may be determined as a static obstacle, and when the first track is created, one object may be divided into several or the estimation of the yaw angle may be incorrect. As Figure 3.18 (a), the side of the bus is initially identified as static obstacle (purple grid). However, it can be seen from Figure 3.18 (b) that the interaction between the two modules occurs using all point clouds measured at 25Hz in real-time and converges to the true value. Verification of detection performance is covered in Chapter 5.



(a) (b)

Figure 3.18. Initial guess and tracked result about oncomming bus.

Chapter 4. Moving Obstacle States Estimation

Chapter 4 explains how to estimate the state of moving obstacles. In the previous chapter, the STOM estimation and the creation and management of tracks have been described. Therefore, this chapter describes an estimation method using the assigned measurement. The process means estimating $\hat{o}_k^n$ using $\hat{o}_{k-1}^n$ and Z_k^n. In addition, the track management method of GMFA with particle filter (PF) is explained in this chapter.

4.1 Extended Kalman Filter

EKF is a method of estimating the state through local linearization and Gaussian assumptions when the process model and measurement model are nonlinear. Since the Gaussian assumption and linearization are used, the posterior probability is expressed as a Gaussian distribution, and through this, the optimal state can be obtained through a simple calculation. However, there is a limitation in estimation performance because the actual probability distribution of the state and measurement is different from Gaussian, and the nonlinear model is locally linearized.

Kalman filter consists of predictions and measurement updates. In EKF, prediction means estimating $\bar{o}_k^n$ using $\hat{o}_{k-1}^n$ and $\hat{x}_{k-1}^e$. Of course, a process

model is required for prediction. GMFA with EKF used a constant acceleration model as the process model. Measurement update means estimating the optimal $\hat{o}_k^n$ using $\bar{o}_k^n$ and Z_k^n. Here, optimal may represent a minimum covariance, maximum posterior probability, or maximum likelihood. In a Gaussian assumption, the minimum covariance and the maximum posterior probability are equal.

4.1.1 Prediction

The constant acceleration model is

$$\begin{aligned}
\dot{x}^{\mathrm{n}} &= \mathbf{a}(x^n, u) + \mathbf{q} \\
&= [\mathrm{a}_1 \ \mathrm{a}_2 \ \mathrm{a}_3 \ \mathrm{a}_4 \ \mathrm{a}_5 \ \mathrm{a}_6 \ \mathrm{a}_7]^{\mathrm{T}} + \mathbf{q} \\
\text{when}, u &\triangleq x^e = [v, \gamma] \\
x^n &\triangleq [p^{x,n},\ p^{y,n},\ \theta^n,\ v^n,\ \gamma^n,\ a^n,\ \dot{\gamma}^n] \\
\mathrm{a}_1 &= v^n \cos(\theta^n) - v + p^{y,n} \cdot \gamma \\
\mathrm{a}_2 &= v^n \sin(\theta^n) - p^{x,n} \cdot \gamma \\
\mathrm{a}_3 &= \gamma^n - \gamma \\
\mathrm{a}_4 &= a^n \\
\mathrm{a}_5 &= \dot{\gamma}^n \\
\mathrm{a}_6 &= \mathrm{a}_7 = 0
\end{aligned} \qquad \text{Eq. 4.1}$$

Eq. 4.1 is discretized with zero order hold assumption as follow:

$$
\begin{aligned}
x_{k+1}^{n} &= \boldsymbol{f_k}(x_k^n, u_k, w_k) \\
&= [f_1 \;\; f_2 \;\; f_3 \;\; f_4 \;\; f_5 \;\; f_6 \;\; f_7]^{\mathrm{T}} \\
f_1 &= p_k^{x,n} + v_k^n \cos(\theta_k^n)\, dt - v_k dt + p_k^{y,n} \cdot \gamma_k \Delta t \\
f_2 &= p_k^{y,n} + v_k^n \sin(\theta_k^n) dt - p_k^{x,n} \cdot \gamma_k \Delta t \\
f_3 &= \theta_k^n + \gamma_k^n \Delta t - \gamma_k \Delta t \\
f_4 &= v_k^n + a_k^n \Delta t \\
f_5 &= \gamma_k^n + \dot{\gamma}_k^n \Delta t \\
f_6 &= a_k^n + w_k^1, \qquad f_7 = \dot{\gamma}_k^n + w_k^2 \\
w_k &\sim (0, Q_k)
\end{aligned}
\qquad \text{Eq. 4.2}
$$

Then, predicted state is

$$
\bar{x}_{k+1}^n = \boldsymbol{f_k}(\hat{x}_k^n, \hat{x}_k^e, 0) \qquad \text{Eq. 4.3}
$$

And predicted covariance is

$$
F_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial x^n}\right|_{\hat{x}_k^n}
$$

$$
= \begin{bmatrix}
1 & \gamma_k \Delta t & -v_k^n \sin(\theta_k^n)\, \Delta t & \cos(\theta_k^n)\, \Delta t & 0 & 0 & 0 \\
-\gamma_k \Delta t & 1 & v_k^n \cos(\theta_k^n) \Delta t & \sin(\theta_k^n) \Delta t & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \Delta t & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \Delta t \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \qquad \text{Eq. 4.4}
$$

$$
L_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial w}\right|_{\hat{x}_k^n}
$$

$$
\overline{\mathrm{P}}_{k+1}^{\mathrm{n}} = F_k \widehat{\mathrm{P}}_k^{\mathrm{n}} F_k^T + L_k Q_k L_k^T
$$

according to Simon['06]

Since the state of the track is expressed in the dynamic state with the SPRB,

$$o_k^n = [S^n, p^{x,n}, p^{y,n}, \theta^n, v^n, \gamma^n, a^n, \dot{\gamma}^n] = [S^n, x^n] \quad \text{Eq. 4.5}$$

the prediction of S^n should proceed.

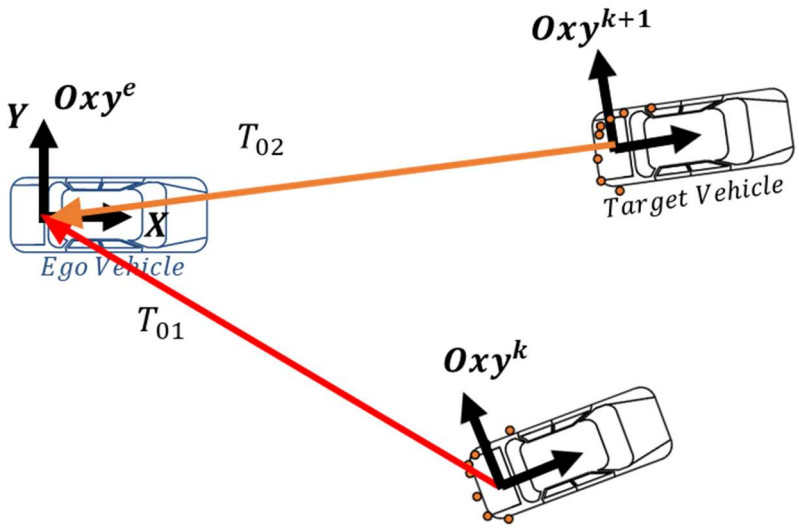


Figure 4.1. SPRB prediction.

T_{AB} denote transformation from coordinate B to coordinate A. The point represented coordinate B can transfer coordinate A via product transform matrix as follows:

$$[p_A^x, p_A^y, 1]^T = T_{\mathrm{AB}} \cdot [p_B^x, p_B^y, 1]^T \quad \text{Eq. 4.6}$$

We define follows: 1 denote coordinate $\boldsymbol{Oxy^k}$; 2 denote $\boldsymbol{Oxy^{k+1}}$; 0 denote ego vehicle's local coordinate ($\boldsymbol{Oxy^e}$). We know SPRB about $\boldsymbol{Oxy^e}$ (S_0^k) and transforms (T_{01}, T_{02}). We assume S on rigid body that center is tracked cluster's mean point. As a result, $S_1^k = S_2^{k+1}$. Then we can calculate S_0^{k+1} as

follow:

$$
\begin{aligned}
S_0^{k+1} &= T_{02} \cdot S_2^{k+1} \\
&= T_{02} \cdot S_1^k \qquad \left(\because \ S_1^k = S_2^{k+1}\right) \\
&= T_{02} \cdot T_{10} \cdot S_0^k \\
&= T_{02} \cdot T_{01}^{-1} \cdot S_0^k
\end{aligned}
$$

When, Eq. 4.7

$$
T(x^n) \triangleq \begin{bmatrix} \cos(\theta^n) & -\sin(\theta^n) & p^{x,n} \\ \sin(\theta^n) & \cos(\theta^n) & p^{y,n} \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
T_{01} = T(\hat{x}_k^n)
$$

$$
T_{02} = T(\bar{x}_{k+1}^n)
$$

T_{01}^{-1} always exists because of the shape of *T*. $\bar{o}_k^n$ is represented via $\hat{o}_{k-1}^n$ and $\hat{x}_{k-1}^e$.

4.1.2 Measurement Update

We will discuss the measurement update using the assigned cluster for *n*-th track, Z_k^n. In EKF, the measurement model must be able to represent the measurement as a state. Our measure is a cluster, and the points of the cluster cannot be directly expressed on our track, which has no state about shape. Therefore, we explain how to extract measurements from the assigned cluster.

In the proposed approach, the three measurements obtained through Z_k^n include the position and the yaw angle of the moving objects. When z^{n} is a

measurement of the EKF, z^{n} is expressed in $[h_1^{\mathrm{n}},\ h_2^{\mathrm{n}},\ h_3^{\mathrm{n}}]^{\mathrm{T}}$ as a 3D vector. The three elements of z^{n} represent the position and yaw angle of the moving object at $\boldsymbol{Oxy^e}$, respectively. The position of the object is considered as the mean of the matched $\bar{\mathrm{S}}_k^n$ that matching $\bar{\mathrm{S}}_k^n$ to Z_k^n by ICP. The moving direction of the object refers to the direction of the displacement vector from the mean of $\hat{S}_{k-1}^n$ to the mean of matched $\bar{\mathrm{S}}_k^n$. These measurements are shown in Figure 4.2, and the measurement model based on these measurements is linear as shown in Eq. 4.8, assuming that the measured values display white Gaussian noise with a covariance matrix of V_k. As shown in Figure 4.2, there is no ghost motion due to geometry changes, since the movement of the mean point in the identical cluster is used as a measure

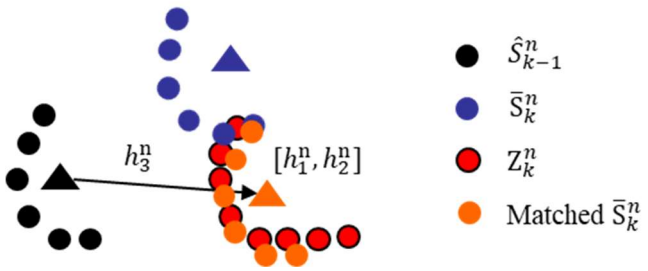


Figure 4.2. Measurements extraction from assigned cluster

$$z_k^{\mathrm{n}} = Hx_k^n + v_k$$

$$v_k \sim (0, V_k)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq. 4.8}$$

Measurements update is achieved via extracted measurements and Eq. 4.9.

$$K_k = \bar{P}_k H^T (H\bar{P}_k H^T + V_k)^{-1}$$
$$\hat{x}_k^n = \bar{x}_k^n + K_k[z_k^n - H\bar{x}_k^n] \quad \text{Eq. 4.9}$$
$$\hat{\mathrm{P}}_k^n = (I - K_k H_k)\bar{\mathrm{P}}_k^n$$

As prediction, $\hat{S}_k^n$ can be obtained as follow:

$$\hat{S}_k^{n\prime} = T_{03} \cdot T_{02}^{-1} \cdot \bar{\mathrm{S}}_k^n$$
$$\text{when}, T_{02} = T(\bar{x}_k^n) \quad \text{Eq. 4.10}$$
$$T_{03} = T(\hat{x}_k^n)$$
$$\hat{S}_k^n = \hat{S}_k^{n\prime} \cup \mathrm{Z}_k^n$$

$\hat{S}_k^n \cup \mathrm{Z}_k^n$ represents points augmentation using FIFO queue as shown in Figure 3.13

4.2 Particle Filter

This section describes the algorithm structure and filtering method of GMFA with PF. GMFA with PF requires more computation than GMFA with EKF, but has the advantage of being able to directly utilize the point cloud as a measurement without feature extraction. In EKF, the measured values must be expressed through states and parameters, but our proposed track does not directly use point clouds as measurement values because there is no state for the obstacle's shape. Therefore, it is necessary to perform clustering for every step and use each cluster as a measurement. Clustering leads to two drawbacks.

The first is that two objects that are not distinguished in clustering cannot be tracked even if they have been tracked sufficiently long period. The second is that because measurement values are extracted through registration via ICP, if the shape changes rapidly, ghost motion occurs due to an error in ICP. The problem caused by clustering and ICP are expressed in Figure 4.3.

Figure 4.3 shows the shortcomings of the GMFA with EKF measurement model in the traffic jam scenario. The green vehicle represents the ego vehicle, and the green arrow, cyan cluster and box represent moving obstacles. Figure 4.3 (a) shows the shortcomings due to the clustering fault of GMFA with EKF. The figure on the left is a situation where a motorcycle approaches from behind in a traffic jam. The confidence index of the motorcycle represented by the red circle is the maximum, and its state is well estimated. On the right, the motorcycle passes between two vehicles moving slowly (the vehicles located rear and rear right side of ego vehicle). At this time, the distance between the motorcycle and the vehicle on rear right became one cluster due to the distance between the two objects lower than the clustering threshold. As a result, the two objects merged into a single obstacle even though both objects had been tracked continuously.

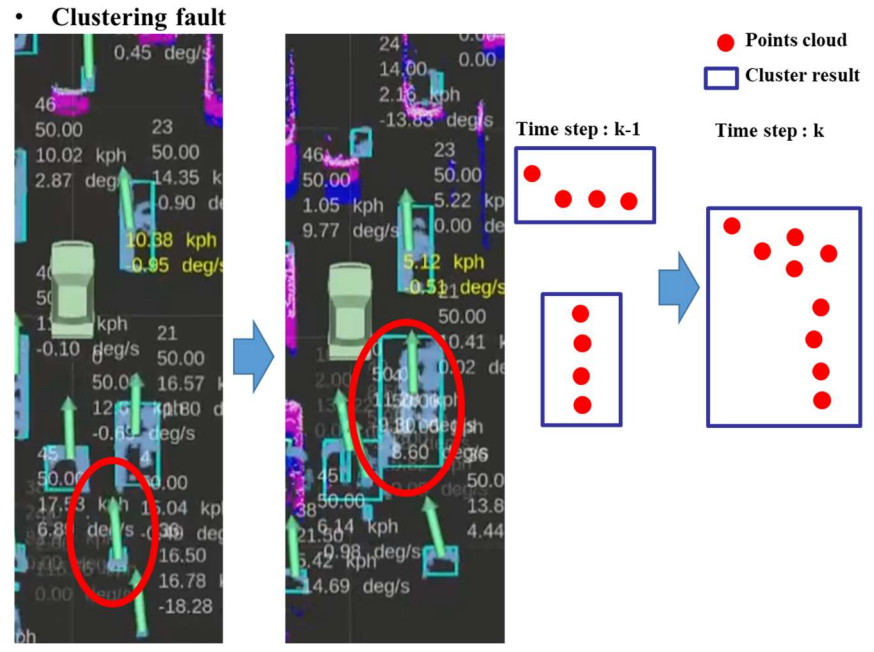


Figure 4.3. The drawbacks of GMFA with EKF due to clustering.

Figure 4.3 (b) shows the shortcomings of GMFA with EKF due to the ICP fault. The target highlighted red circle goes straight with the high relative speed at the ego vehicle's right side. The cluster's shape of the object has changed dramatically as right figure when passing through the ego vehicle. If the shape changes rapidly, the matching result is inaccurate because the shape is not considered in ICP registration. Since the cost of ICP considers the distance between the closest points, the erroneous registration result has the minimum cost. Therefore, the measurements obtained through the erroneous registration is skewed to the left as Figure 4.3 (b). As a result, the yaw angle is tilted toward the ego vehicle like an obstacle in a red circle.

Figure 4.4 shows the PF-related parts in Figure 3.2 and shows the structure of GMFA with PF. I/O is the same as GMFA with EKF, except that clustering is only utilized to initialize track. When a point cloud (y_k^o) is entered, a likelihood field, $p(y|\bar{O}_k)$, is created through the entered point cloud. In principle, the likelihood field should be made through the state of moving obstacles. However, we generate the likelihood through the measurement, and through this, PF can be used in real-time. If the likelihood field is created through the state, the likelihood field has to be created as much as the number of particles in all tracks. It is computational demanding. The rationale for generating the likelihood field through measurements is covered in 4.2.2.

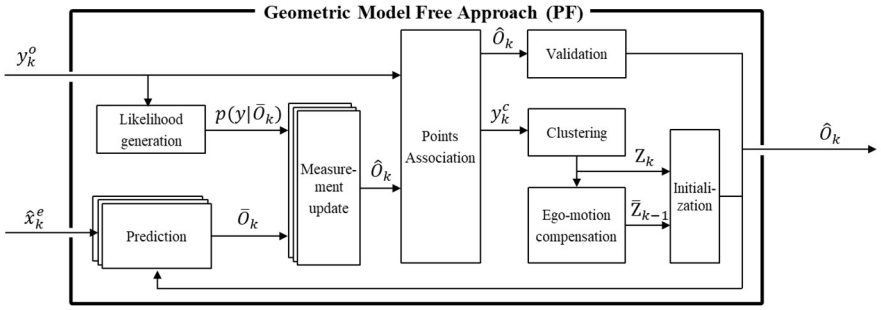


Figure 4.4. Configuration of GMFA with particle filter.

EKF expresses the probability distribution of states through mean and covariance under Gaussian distribution assumption as Eq. 4.11.

$$o^n = [S^n, x^n, \mathrm{P}^n] \qquad \text{Eq. 4.11}$$

However, in reality, this assumption does not hold due to nonlinearity. To cope with this, PF directly expresses the probability distribution of the state via *N* particles. Although it requires a lot of computation and memory, it can take advantage of parallel processing and can represent various types of probability distribution. Therefore, the composition of the track in PF is as follows:

$$o^n = \{o^{n,i}\} \qquad (i = 1, \dots, N)$$
$$o^{n,i} = [S^{n,i}, x^{n,i}, q^{n,i}] \qquad \text{Eq. 4.12}$$

$o^{n,i}$ denotes i-th particle of n-th track and $q^{n,i}$ denotes relative likelihood of the particle. The track does not include covariance (P) because the probability distribution is represented via *N* particles.

Prediction calculate $\bar{O}_k$ using $\hat{x}_{k-1}^e$ and $\hat{O}_{k-1}$ via constant velocity model. After the prediction, each predicted particle's relative likelihood is updated

using the likelihood field ($p(y|\bar{O}_k)$) and resampling is conducted to prevent degeneracy at measurement update. Details of prediction and update will be described later.

In the point association, the SPRB (S^n) of the updated track ($\hat{O}_k$) and the input point cloud (y_k^o) are compared, and the points are assigned to the nearest track among tracks that are closer than threshold. The distance to the track is calculated by comparing it with the nearest point of the SPRB. The assigned points are added to each track's S^n in the same way as Figure 3.13.

After point association, points that are not assigned to any track are used to create a new track through clustering. This method is the same as described in 3.3.1. In the validation, the confidence index is updated through the number of points allocated to the track, and tracks with low confidence index are deleted.

4.2.1 Prediction

Since PF uses a constant velocity model differently from EKF, predictions are made using exact solutions without first-order approximation. In addition, since a and $\dot{\gamma}$ are too high frequency to estimate using 20Hz measure, the state of PF consists of five states as follows:

$$x^n \triangleq [p^{x,n},\ p^{y,n},\ \theta^n,\ v^n,\ \gamma^n] \qquad \text{Eq. 4.13}$$

Prediction for the five states as follows using Eq. 4.14 and Eq. 4.15:

$$
\begin{aligned}
x_{k+1}^{n,i} &= \boldsymbol{f_k}\left(x_k^{n,i}, u_k, w_k^i\right) \\
&= [\mathrm{T}_e[f_1; f_2]; f_3;\ f_4;\ f_5] \\
f_1 &= p_k^{x,n} - \frac{v_k^{n,i}}{\gamma_k^{n,i}}\sin\left(\theta_k^{n,i}\right) + \frac{v_k^{n,i}}{\gamma_k^{n,i}}\sin\left(\theta_k^{n,i} + \gamma_k^{n,i}\Delta t\right) \\
f_2 &= p_k^{y,n} + \frac{v_k^{n,i}}{\gamma_k^{n,i}}\cos\left(\theta_k^{n,i}\right) - \frac{v_k^{n,i}}{\gamma_k^{n,i}}\cos\left(\theta_k^{n,i} + \gamma_k^{n,i}\Delta t\right) \\
f_3 &= \theta_k^{n,i} + \gamma_k^{n,i}\Delta t - \gamma_k \Delta t + w_k^{1,i} \\
f_4 &= v_k^{n,i} + w_k^{2,i} \\
f_5 &= \gamma_k^{n,i} + w_k^{3,i} \\
w_k &\sim U(0, Q_k)
\end{aligned}
\qquad \text{Eq. 4.14}
$$

$$
\mathrm{T}_e = \begin{bmatrix} cos(\gamma\Delta t) & sin(\gamma\Delta t) & -\frac{v}{\gamma}sin(\gamma\Delta t) \\ -sin(\gamma\Delta t) & cos(\gamma\Delta t) & \frac{v}{\gamma} - \frac{v}{\gamma}cos(\gamma\Delta t) \\ 0 & 0 & 1 \end{bmatrix}
$$

Ego vehicle motion affects only position and yaw because the position and yaw angle are based on the ego vehicle's local coordinate system, and speed and yaw rate are based on the global coordinate system. The process noise (w_k) is assumed as uniform distribution because it depicts the intention of the target's driver and we don't have any clue for the intention.

Using $\bar{x}_{k+1}^{n,i} = \boldsymbol{f_k}\left(\hat{x}_k^{n,i}, u_k, w_k^i\right)$ and Eq. 4.7, $S^{n,i}$ can be predicted as

$$\bar{S}_{k+1}^{n,i} = T_{02} \cdot T_{01}^{-1} \cdot \hat{S}_k^{n,i}$$

When,

$$T(x^n) \triangleq \begin{bmatrix} cos(\theta^n) & -sin(\theta^n) & p^{x,n} \\ sin(\theta^n) & cos(\theta^n) & p^{y,n} \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Eq. 4.15}$$

$$T_{01} = T(\hat{x}_k^{n,i})$$

$$T_{02} = T(\bar{x}_{k+1}^{n,i})$$

Figure 4.5 depict prediction explained in Eq. 4.14 and Eq. 4.15.

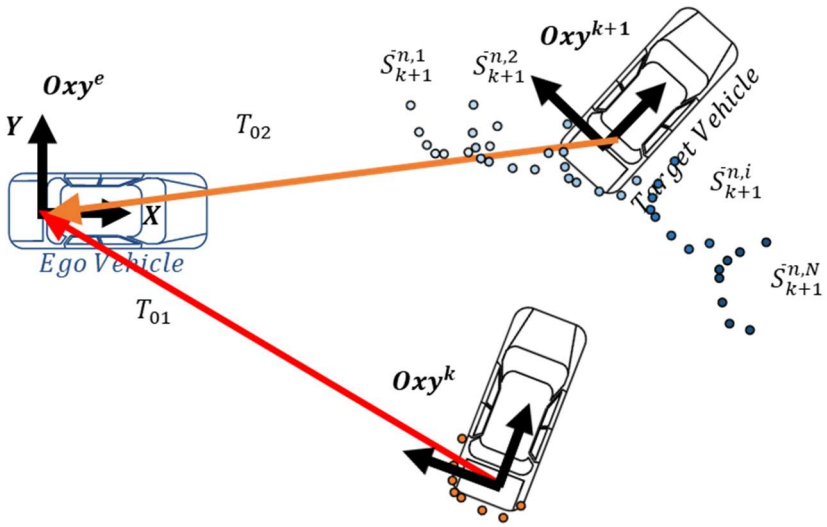


Figure 4.5. Prediction of particle filter.

PF compares each particle's predicted states with the measurement, updates each particle's weight in measurement update. Thus, the states of particle are changed only in the prediction. Therefore, to increase PF's estimation performance, it is important to maintain a number of particles similar to the

measurement. In this way, there is a method of increasing particles around the predicted position by reducing process noise. However, in this case, divergence can occur if a measurement deviates from the prediction. To avoid degeneracy problems, we need to spread particles to various locations. If the number of particles can be increased to infinity, both requirements are satisfied, but this is impossible in a real-time algorithm. In order to satisfy the real-time requirement in our environment, about 200 particles were maximum.

To utilize a limited number of particles efficiently, process noise is applied only to the yaw and speed. Yaw rate is estimated through a simple alpha filter using the estimated yaw. When the state's dimension increases, the number of particles required increases exponentially due to Curse of dimensionality. To model the target's intention, process noise is assumed to be a uniform distribution because we don't have any clue for the target's intention.

4.2.2 Measurement Update

The measurement update of PF refers to updating the relative likelihood (= importance weight) through comparison of the predicted state and the measurement, and resampling the track where sample impoverishment has occurred. Relative likelihood is defined as follow:

$$q_k = \frac{target\ \ distribution}{proposal\ \ distribution}$$

$$= \frac{\eta p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1})}{p(x_k|x_{k-1})\pi(x_{0:k-1}|z_{1:k-1})} \qquad \text{Eq. 4.16}$$

$$\propto p(\mathrm{z}_k|x_k)q_{k-1}$$

To calculate relative likelihood, $p(\mathrm{z}_k|x_k)$ is need to available. According to the definition of likelihood, it should be defined in the form of state as a parameter and measurement as a variable. By definition, a new likelihood should be calculated for every particle if the likelihood is asymmetric. Since this is too computational demanding to real-time process, we proposed the measurement model as Eq. 4.17.

Measurement model (z, S)

$$p(\mathrm{z}_k|x_k) = \prod_{i=1}^{l} N(p_k - s_i, R)$$

$$\text{when}, S = \{s_1, \dots, s_l\} \qquad \text{Eq. 4.17}$$

$$\mathrm{z} = y^o = \{p_1, \dots, p_m\}$$

s, p : 2-dimensional point

p_k : nearest neighbor about s_i among y^o

Since Gaussian distribution is symmetric and only the closest point among constituting the SPRB is used, a likelihood field can be generated using the input point cloud (y^o).

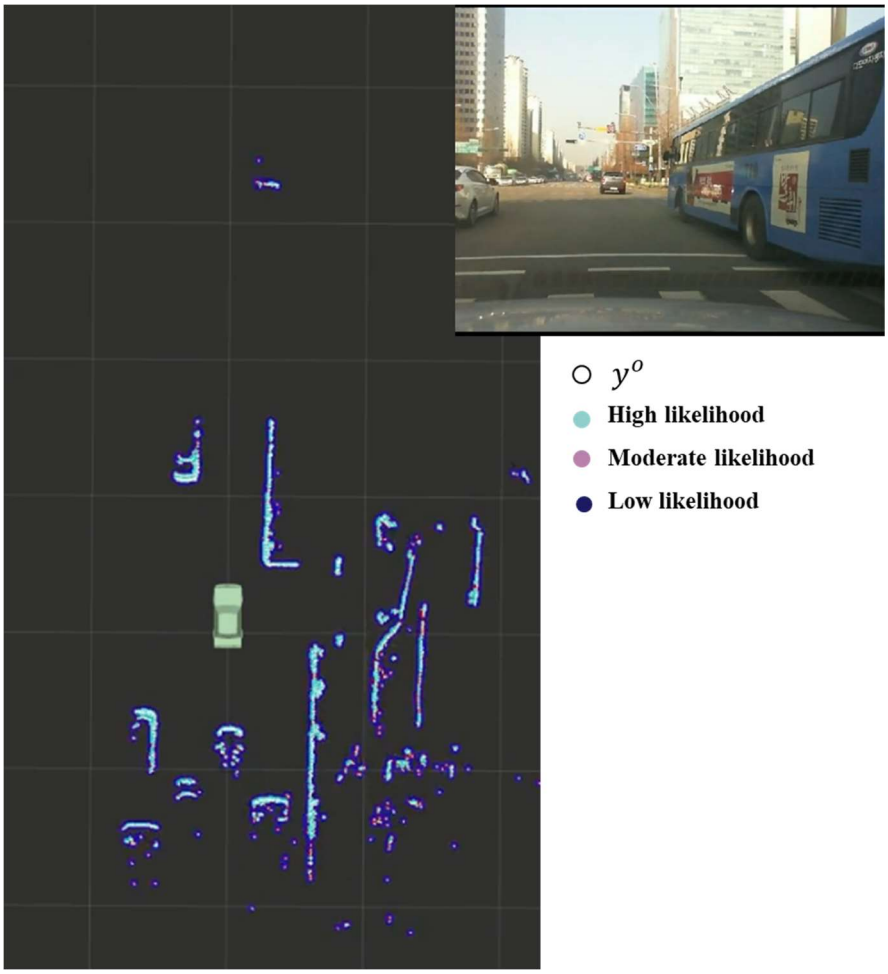


Figure 4.6. Likelihood field generated via point cloud.

Figure 4.6 depicts likelihood field generated via a point cloud. Since the likelihood field is configured in the form of a grid map, the likelihood can be quickly calculated using the look up table with O(l) complexity when calculating the likelihood. Using the likelihood, the relative likelihood is updated as follow:

$$q_k^{n,i} = \frac{p(y_k^o|S_k^{n,i})q_{k-1}^{n,i}}{\sum_{i=1}^{N} p(y_k^o|S_k^{n,i})q_{k-1}^{n,i}} \quad \text{Eq. 4.18}$$

After update weight, we need to determine which track occurs sample impoverishment. To determine sample impoverishment, the effective sample size N_{eff} is introduced in [Bergman'99] and [Liu et al.'98] and defined as

$$N_{eff} = \frac{N}{1 + Var(q_k^{*,i})} \quad \text{Eq. 4.19}$$

Where $q_k^{*,i}$ is denote true weight. The true weight cannot be evaluated, so estimation of N_{eff} can be obtained by

$$\widehat{N}_{eff} = \frac{N}{\sum_{i=1}^{N}\left(q_k^{n,i}\right)^2} \quad \text{Eq. 4.20}$$

In our approach, the track is resampled that $\widehat{N}_{eff}$ is lower than 0.5*N*. Finally, estimated state as

$$\hat{x}_k^n = \sum_{i=1}^{N} q_k^{n,i} \cdot \bar{x}_k^{n,i} \quad \text{Eq. 4.21}$$

$\hat{S}_k^n$ is obtained via Eq. 4.10 and $\hat{x}_k^n$. Point association is based on $\hat{S}_k^n$. After augmenting new points to $\hat{S}_k^n$, $\bar{S}_k^{n,i}$ of all particles is updated to $\hat{S}_k^{n,i}$ being same shape as $\hat{S}_k^n$.

Since yaw rate is identical among all particle, yaw rate need to update using $\hat{x}_k^n$. The yaw rate is updated using alpha filter as follow:

$$\hat{\gamma}_k^n = \hat{\gamma}_{k-1}^n + \alpha(\hat{\theta}_k^n - \hat{\theta}_{k-1}^n + \Delta t(\gamma^e - \hat{\gamma}_{k-1}^n))/\Delta t \quad \text{Eq. 4.22}$$

Chapter 5. Performance Evaluation

The proposed approach has been verified by comparing with geometric model-based tracking (MBT). MBT extracts the possible shape candidates of targets from the current point cloud, and tracks the shape candidates using the multiple hypothesis tracking (MHT) framework proposed in [Cho'14].

Figure 5.3 explains how MBT works step by step. The front cam image in the upper right at Figure 5.3 shows the driving situation. First, the driving environment measured using LiDAR is shown in Figure 5.3 (a). As shown in Figure 5.3 (b), the ROI is set based on the lane, the points included in the ROI are clustered using the DBC algorithm, and then the bounding box is fitted using Eq. 5.1 for each cluster. Box fitting is performed by creating a convex hull of a cluster as shown in Figure 5.1. And then, creating a rectangle containing all points about each side of the convex hull, and calculating the cost of each box. Using the created box and virtual ray, the final candidate is generated using the method of Figure 5.2. The results of the extracting candidates is represented in Figure 5.3 (c). Finally, the estimation results via EKF using the candidates as measure are depicted in Figure 5.3 (d).

The tracks updated continuously for more than three steps are identified as moving objects and we only treat the points on the road for MBT using a pre-configured HD map to prevent mixing with static obstacles. Various studies reported object detection and tracking using point cloud, however, based on

real-time characteristics, the proposed approach is compared with the MBT and each other.

$$J_i = \sum_{k=1}^{n} exp(d_k) \qquad \text{Eq. 5.1}$$

$$Bounding\ Box = arg\,min\, J_i$$

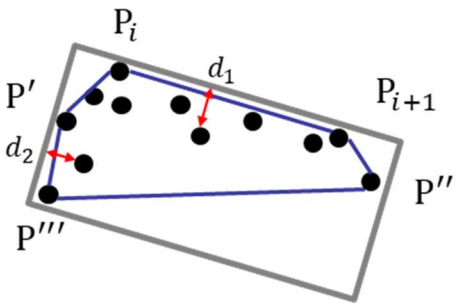


Figure 5.1. Bounding box fitting using convex hull.

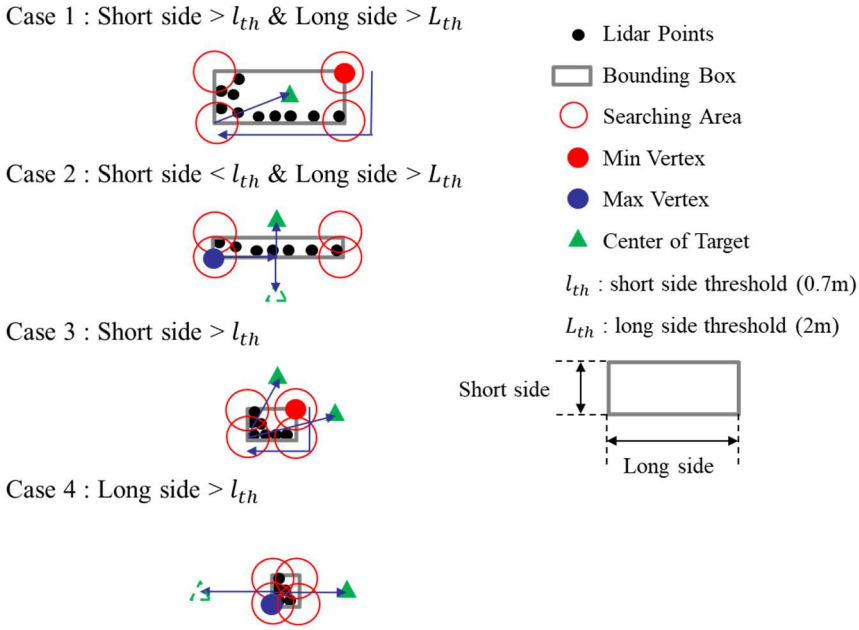
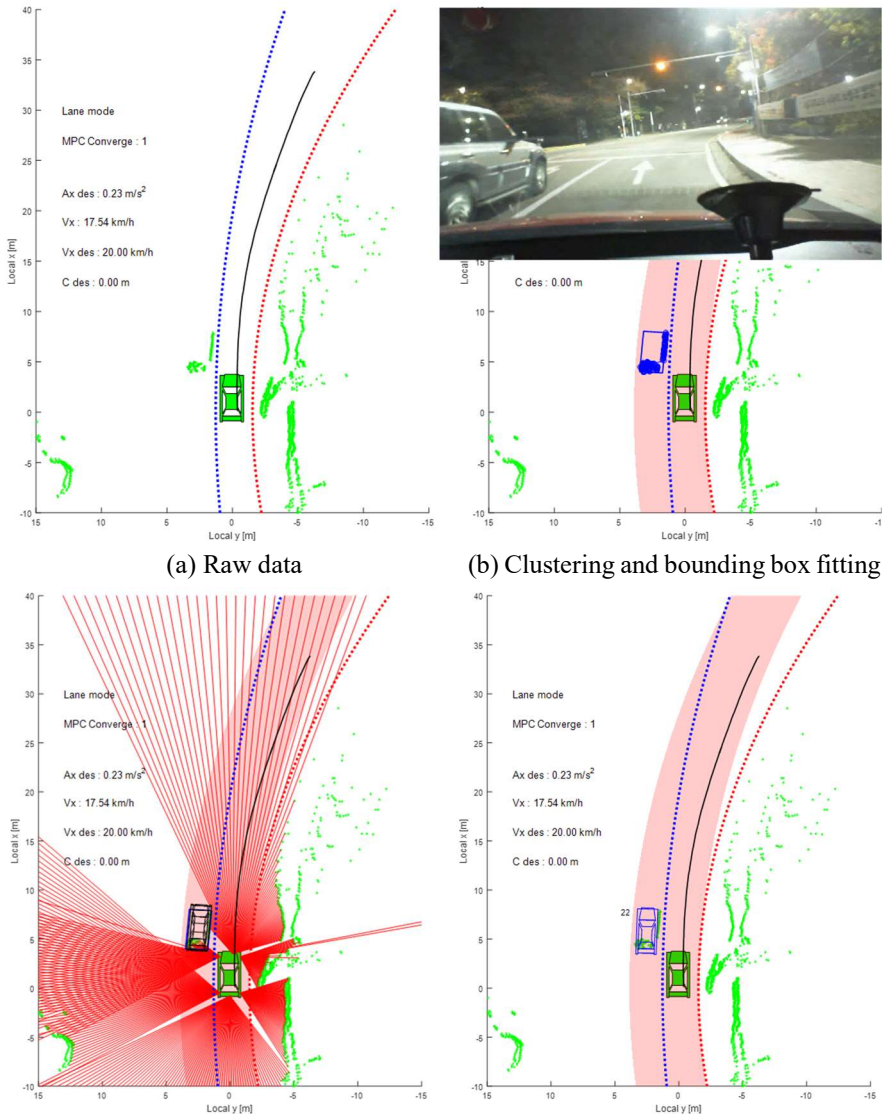


Figure 5.2. Center point extraction using virtual ray and box fitting.



(a) Raw data (b) Clustering and bounding box fitting

(c) Virtual ray (d) EKF filtering

Figure 5.3. Step by step results of MBT.

The proposed perception module has been evaluated in the driving environment using the AVs described in Section 1.2 about detection and estimation performance. The detection performance has been verified by

labeling the data acquired from the Nambu-Beltway and Seoul National University (SNU) campus. Since the detection performance is mainly influenced the perception module's interaction structure, the detection performance has been compared with MBT and GMFA in Labview/Matlab environment. Estimation performance has been compared in both Labview/Matlab and ROS/c++ environments, and analyzed through data acquired using RT-range in Future Mobility Technical Center (FMTC). In addition, qualitative evaluation using data acquired in various urban environments has been conducted. In order to check whether the algorithm complies with real-time requirements, the distribution of calculation time for driving data has been analyzed.

Labview/Matlab environment operates the proposed perception module in real time with i7-4790 4.00GHz CPU. In the environment, the perception module is implemented as a single-threaded process in a sequential structure. ROS/c++ environment operates the proposed perception module in real time with i7-9700K 3.60GHz CPU. In the environment, the perception module is implemented as a multi-threaded process in a parallel structure.

5.1 Moving Obstacle Detection

To obtain the detection results, the driving data from Nambu-Beltway and Seoul National University (SNU) campus using the Red Ioniq described in

Section 1.2 have been labeled with moving obstacles and analyzed via precision, recall, and F_1 scores. The performance index is defined as

$$Precision = \frac{True\ \ Positive}{True\ \ Positive\ +\ False\ \ Positive}$$

$$Recall = \frac{True\ \ Positive}{True\ \ Positive + False\ \ Negative} \qquad \text{Eq. 5.2}$$

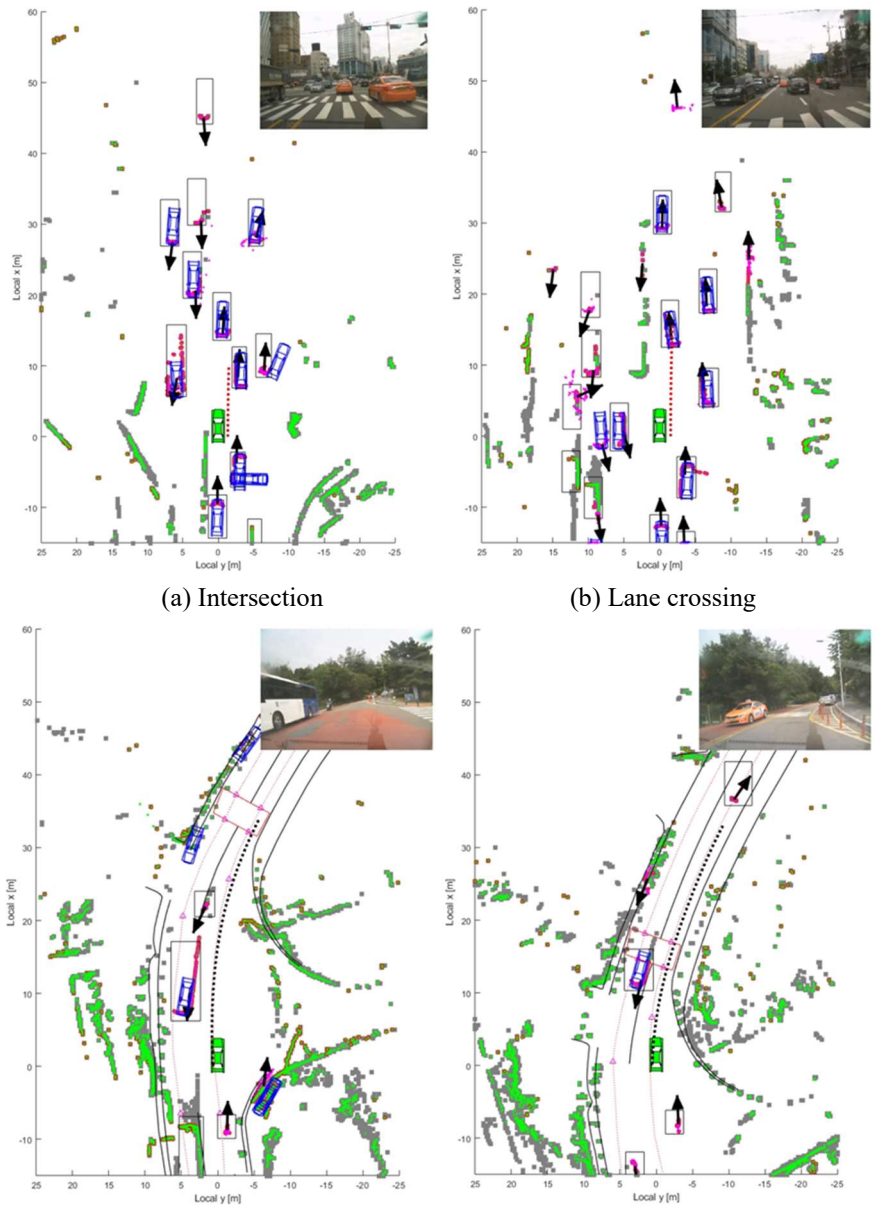
$$F_1 score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Data of 80m > x > -15m, | y | < 25m have been used for labeling using a front camera. Red Ioniq perceives the surroundings in the MATLAB-Labview environment.

Each data frame is labeled with moving objects using a front camera, and 4455 moving objects are labeled including cars, trucks, buses, and motorcycles with various scenarios as Figure 5.4. In this study, the grid size of STOM is 0.2 m. The objects moving faster than 13.5kph are labeled as a moving obstacle, because an object is considered moving when it moved more than 1.5 grid at a time interval of a consecutive scan of 0.08s on average. The test road included a variety of urban environments such as intersections, pocket roads, speed bumps, and crosswalks.

In Figure 5.4, the green vehicle represents the ego vehicle, whereas the magenta clusters and the black arrows indicate the results of GMFA. The blue vehicles denote the results of MBT, and the black squares represent the label. Green dots and gray square denotes LiDAR points cloud and STOM,

respectively.



(a) Intersection (b) Lane crossing

(c) Bus and Motorcycle (d) Bump

Figure 5.4. Various labeled scenarios for detection performance evaluation.

TABLE 5.1. THE MOVING OBJECT DETECTION RESULT OF GMFA AND MBT

Method	Dataset	Moving Objects	Detected Objects	Correctly Detected	Precision	Recall	F_1score
GMFA	Nambu-Beltway	3915	3828	3508	0.916	0.896	0.906
	SNU Campus	540	568	486	0.856	0.900	0.877
	Overall	4455	4396	3994	**0.909**	**0.897**	**0.902**
MBT	Nambu-Beltway	3915	3359	2690	0.801	0.687	0.740
	SNU Campus	540	287	228	0.794	0.422	0.551
	Overall	4455	3646	2918	**0.800**	**0.655**	**0.720**

The detection results are presented in Table 5.1. The F_1score of the proposed algorithm is approximately 25% higher than MBT because both precision and recall are increased, as shown in Table 5.1. In this respect precision is improved by 0.109, and recall is improved by 0.242. An increase in precision indicates the reduced number of false alarms, and a significant improvement in recall denotes a decrease in the false-negative outcomes.

Figure 5.5 and Figure 5.6 depicts a frame for the evaluation to intuitively explain the improvement in the results of detection. In the case of (i) in Figure 5.5, the moving object within three frames from the first measured value cannot be detected via GMFA due to the nature of *track before detect* framework. MBT is also failed to detect the object (i) because the object is tracked for less than 3 steps. The components (ii), (iii), (iv), and (v) are detected in the case of GMFA without shape assumption. However, MBT fail to detect the objects because of the difference between the assumed vehicle shape and the actual cluster shape caused by occlusion. In the case of (vi), GMFA accurately perceive a single

object. However, in the case of MBT, false positives occurred because all the possible object shapes are generated using a point cloud and MBT tracks the all possible shapes.

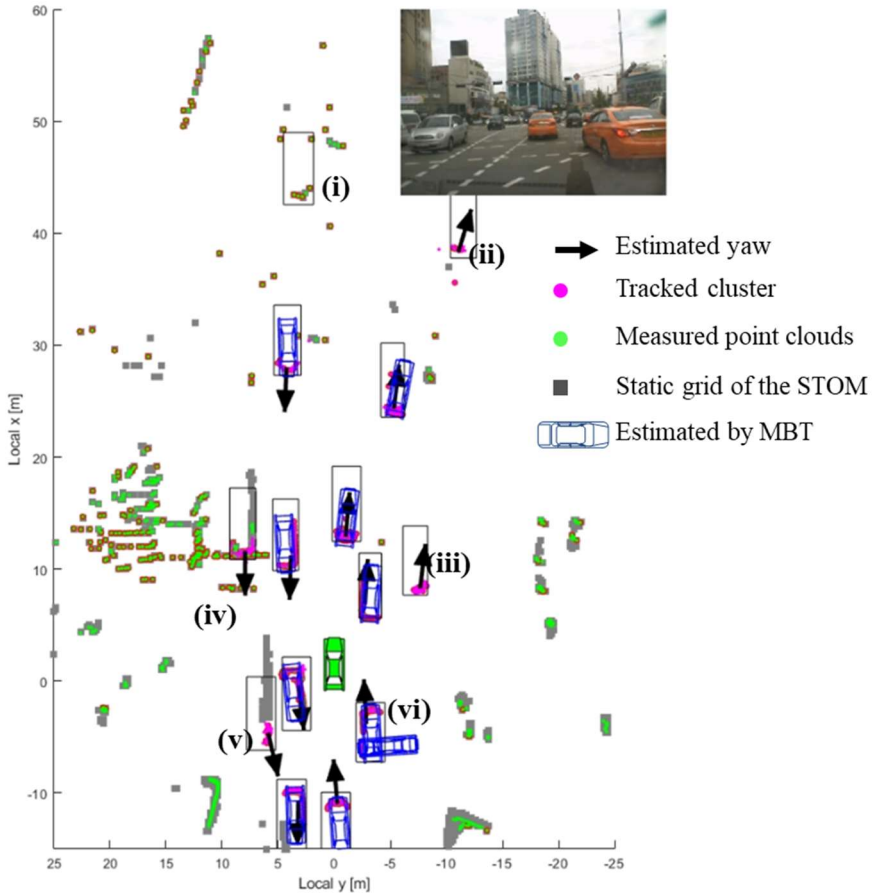


Figure 5.5. Detection result at intersection on Nambu-Beltway.

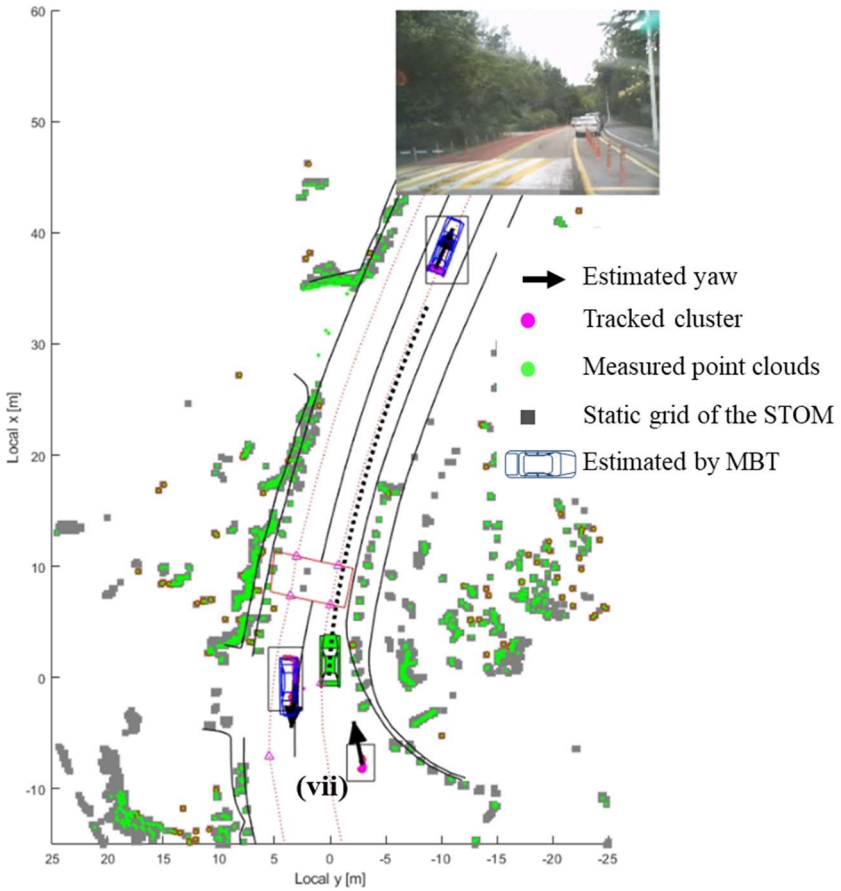
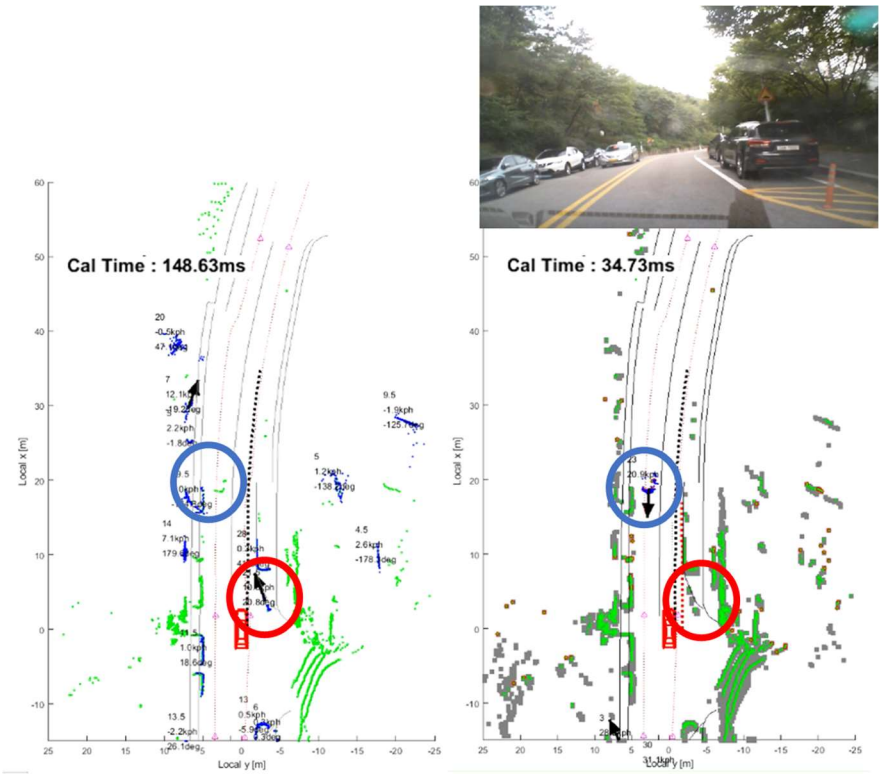


Figure 5.6. Detection result about motorcycle at SNU.

In Figure 5.6, both the front and left vehicles are adequately matched with the geometrical assumption of MBT. As a result, both GMFA and MBT detects the vehicles accurately. However, in the case of (vii), which measures a motorcycle, MBT fail to detect because of the difference between the actual shape of the object and the assumed shape. However, it is confirmed that GMFA detects accurately without shape assumption. Due to the various forms of traffic and occlusion under an urban environment, moving objects are represented by

various shapes that cannot be fully expressed using a shape model. Therefore, GMFA gets higher detection performance compared with MBT.



(a) without STOM (b) with STOM

Figure 5.7. Improving detection performance and calculation time via interaction.

Figure 5.7 shows the effect of interaction with STOM on detection performance and computation time. First of all, it can be seen that the calculation time has been reduced to a quarter.

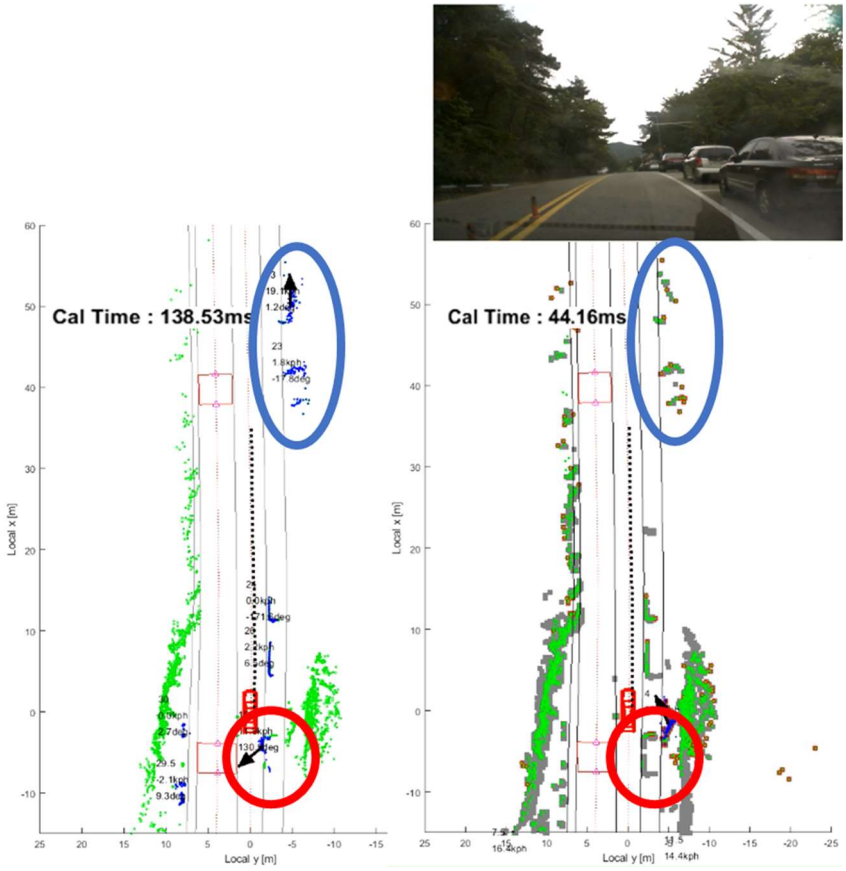
If we look at the red circle in Figure 5.7 (a), we can find that the pole is confused for a moving obstacle without interaction. The false-positive is caused by three consecutive polls being clustered differently in every scan. However,

in the red circle in Figure 5.7 (b), it can be seen that the false positives disappeared because the static obstacle is filtered through STOM.

At the blue circle in Figure 5.7 (a), it can be seen that the moving taxi is not perceived as a moving obstacle. The false-negative occurs when the car parked on the left and the moving taxi merge into one cluster. Such the false-negative frequently occurs in urban roads where the distance from surrounding obstacles is often close. Such examples can be found in vehicles passing by a lane next to a central divider or a vehicle passing by parked vehicles. At the blue circle in Figure 5.7 (b), we can confirm that the taxi is correctly identified as a moving obstacle because of STOM filters out the parked car.

Figure 5.8 shows the reduction of false-positive of parked vehicles through STOM on the SNU campus. First of all, it can be seen that the calculation time has been reduced to a third due to STOM. The reason for this decrease in computation time is that clustering and tracking are performed only for the remaining points except for the static obstacle. At the red circle in Figure 5.8 (a), The parked vehicle next to the ego vehicle is estimated as moving toward the lane. The parked vehicle is estimated as moving because the vehicle's shape changes dramatically when passing through. Therefore, the tracked SPRB and the measured cluster are greatly different. If the tracked shape and the measured cluster shape differ significantly, a false registration is made during the ICP registration, as shown in Figure 4.3 (b). As a result of estimating by using the extracted measurement through false registration, the erroneous estimation

occurs, as shown in Figure 5.8 (a). However, in Figure 5.8 (b), since it is determined as a static obstacle through STOM, the parking vehicle's misrecognition is removed.



(a) without STOM (b) with STOM

Figure 5.8. Improving detection performance and calculation time about parked cars.

The moving object detected on the red circle in Figure 5.8 (b) results from measuring sidewalk, and these points are measured as if they are actually moving at the same velocity as the ego vehicle. Therefore, the sidewalk

misrecognition cannot be resolved by GMFA framework, and an additional algorithm using HD map or detection is required.

At the blue circle in Figure 5.8 (a), the two parked vehicles are measured. The vehicle in front is estimated to moving forward as 19 kph, and the vehicle in the rear is estimated to stop. If the same isolated shape is measured steadily as the rear, it can be estimated correctly by GMFA alone. However, even if it is an isolated shape like the front, it has to be misrecognized as a moving object when the shape changes rapidly due to the road's slope and the pitch motion. On the other hand, in Figure 5.8 (b), it can be confirmed that the misrecognition has disappeared in the same situation because the object is recognized as a static obstacle by STOM before GMFA tracking.

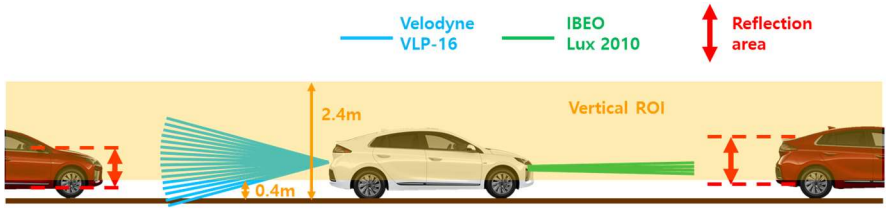


Figure 5.9. Factors affecting the maximum detection range.

TABLE 5.2 DETECTION RANGE ABOUT FOUR CASES

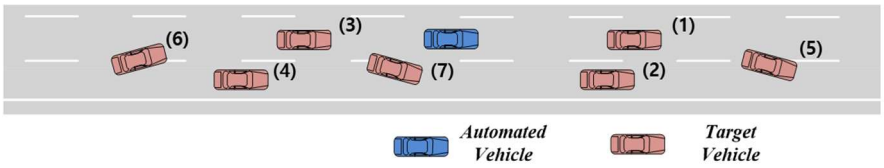
	Rear in			Rear out			Front in			Front out		
	MBT	EKF	PF	MBT	EKF	PF	MBT	EKF	PF	MBT	EKF	PF
Mean [m]	23.1	**28.9**	28.7	21.1	33.8	**34.7**	50.6	77.6	**80.0**	55.6	**88.4**	85.5
Min [m]	20.4	**28.2**	26.9	18.5	29.1	**29.1**	36.7	64.8	**71.3**	47.5	**82.3**	77.7
Max [m]	26.3	30.2	**31.1**	23.7	38.5	**40.4**	55.8	85.6	**86.1**	65.5	95.7	**95.7**

The maximum detection range is one of the most important characteristics of AV for safety. As shown in Figure 5.9, the maximum detection range using LiDAR is affected by the type of LiDAR and the shape of the object (front and rear shapes of the vehicle). Also, since the proposed module utilizes the *track before detect* framework, the object's state of the previous step affects the detection range. Therefore, we conducted repeated experiments by dividing them into four cases: rear in, rear out, front in, and front out based on the ego vehicle. Since we experimented with white Ioniq as a Hunter and red Ioniq as Target, the rear result means the maximum detection range when measuring the front of Ioniq with VLP-16. The front result depicts each algorithm's maximum detection range when measuring the rear of Ioniq with Lux 2010. The average and minimum-maximum of the maximum detection range are shown in Table 5.2.

From Table 5.2, it can be seen that GMFA guarantees a wider detection range than MBT in all cases. The detection performance in the sparse point cloud is the main factor for the maximum detection range. Since it is difficult even for humans to extract the shape using a sparse point cloud, the proposed geometric model-free algorithm shows better performance than the geometric model-based algorithm. In addition, GMFA with EKF and PF show similar performance because clustering and ICP failures are rare in sparse point cloud.

5.2 Moving Obstacle State Estimation

In this section, the estimation accuracy of the yaw angle and the speed of moving obstacle have been validated by driving data with RT range. The proposed approach is reliable for position due to tracking the point clusters of moving objects directly. The reference data have been acquired using RT-range and autonomous vehicles under various driving scenarios as Figure 5.10. Notably, the lane keeping (LK) scenarios involve target vehicle driving along the lane at the same speed as the hunter vehicle in the front, front side, rear, and rear sides based on the hunter vehicle speeds of 40 kph and 80 kph. Lane changing (LC) scenarios are also collected in which the target changes lanes in the front and rear of the ego vehicle with a driving speed of 40 kph. Overtaking scenarios include both being overtaken and overtaking.



	Speed[kph]	Position
Lane keeping	40	(1) (2) (3) (4)
	80	(1) (2) (3) (4)
Lane changing	40	(5) (6)
Overtaking	40	(7)

Figure 5.10. Driving scenarios to evaluate estimation performance.

The estimated performance has been evaluated in both Labview/MATLAB

environment and ROS/c++ environment. First, we analyze the result of comparing MBT and GMFA with EKF in Labview/MATLAB environment, and then compare MBT and GMFA with EKF and PF in ROS/c++ environment.

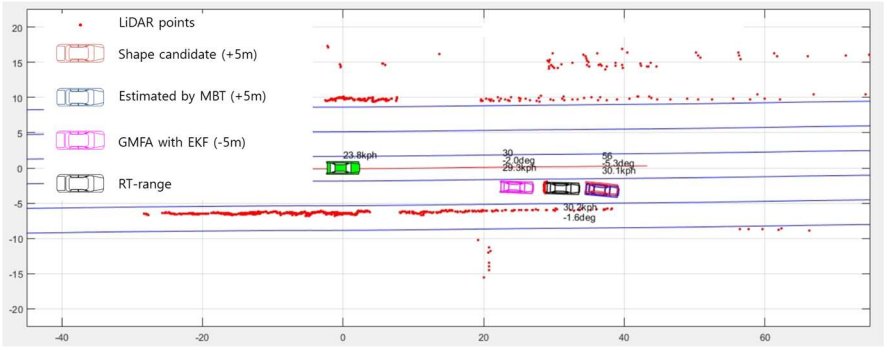


Figure 5.11. Reference data at Labview/MATLAB.

RT-range is equipment to measures the relative position, speed, yaw, and acceleration with high accuracy. RT-range collects data from RT, Oxts' high-performance GPS, equipped in ego and target vehicle, respectively, through communication. Using RT-range, the target's reference data can be obtained, and Figure 5.11 shows the reference data and the estimated result by each algorithm. Red dots and blue lines represent LiDAR point clouds and lanes, respectively. Black, red, blue, and magenta vehicles represent reference data, shape candidates, MBT results, and GMFA with EKF results.

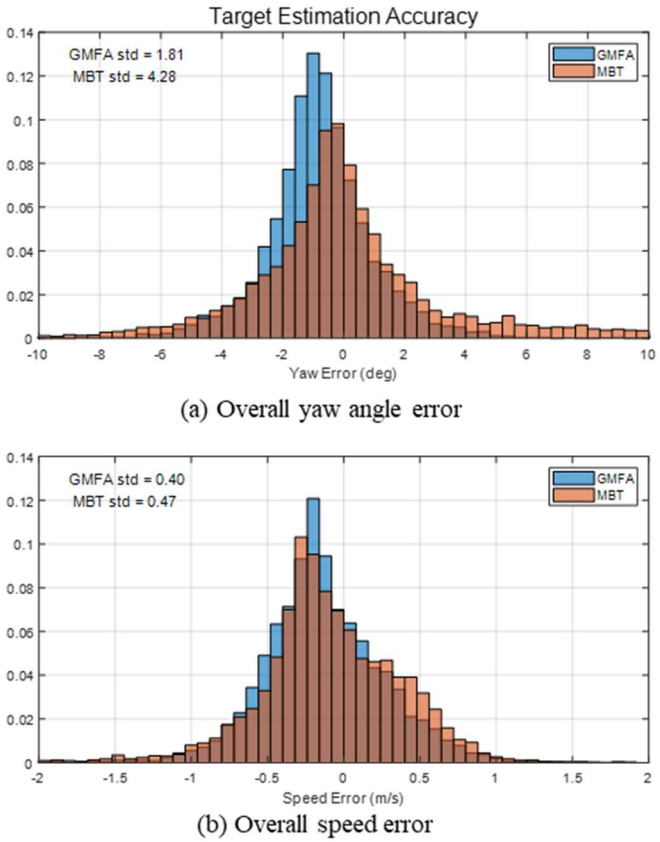


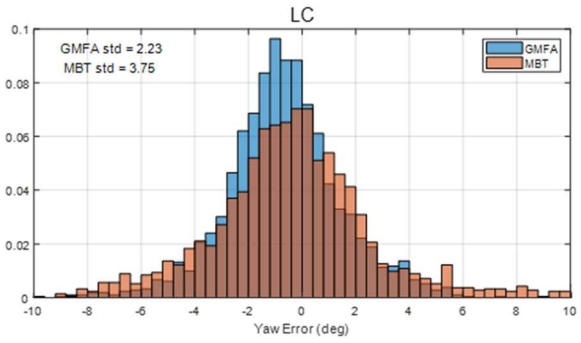
Figure 5.12. Estimation error distribution about GMFA with EKF and MBT.

Figure 5.12 depicts the error distribution of yaw angle and speed estimated by GMFA with EKF and MBT about whole reference date. The blue and red represent the error distributions of the proposed approach and MBT, respectively. GMFA with EKF makes more precise estimates than MBT for both yaw and speed errors. In a lane change scenarios, GMFA with EKF estimates more precisely than MBT as Figure 5.13. The bias is due to calibration errors in the RT range and LiDAR, and asymmetry of the target

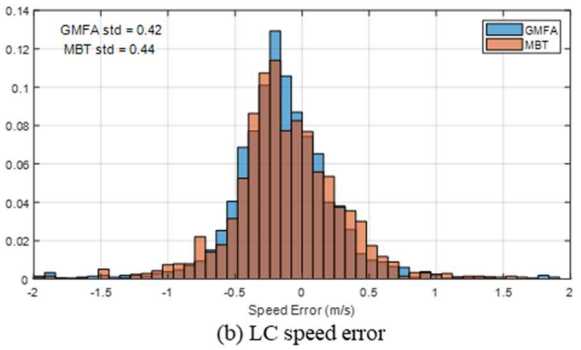
position and the road shape during data acquisition. Therefore, the standard deviation indicating the distribution of errors is recommended for the validation of estimation accuracy, and the values of each approach are shown in Table 5.3.

TABLE 5.3. ESTIMATION RESULTS OF GMFA WITH EKF AND MBT

	Standard deviation of	Lane keeping	Lane changing	Total
GMFA	Yaw angle[deg]	1.64	2.23	1.81
	Speed[kph]	0.40	0.42	0.40
MBT	Yaw angle[deg]	4.46	3.75	4.28
	Speed[kph]	0.48	0.44	0.47



(a) LC yaw angle error



(b) LC speed error

Figure 5.13. Estimation error distribution about lane changing scenarios.

As shown in Table 5.3, GMFA with EKF enhances the estimation accuracy compared with MBT under all scenarios. The estimation accuracy of yaw has been improved by 58% compared with MBT. Also, GMFA with EKF estimates speed 15% more accurately. We will explain why the GMFA made a more accurate estimate through a specific scene.

Figure 5.14 depicts the reference states and the estimated states at (b and c), the estimation error (d and e), and scene from bird's eye view (a) during forward lane changes over time. Figure 5.14 (a) represents reference, extracted shape, and estimated states via black, red, blue and magenta car. The yaw angle, speed, and each error at the instant are represented by points in Figure 5.14 (b), Figure 5.14 (c), Figure 5.14 (d), and Figure 5.14 (e). For the visualization, the estimation results via GMFA with EKF and MBT are plotted with + 5 m and - 5 m offset, respectively, in Figure 5.14-(a).

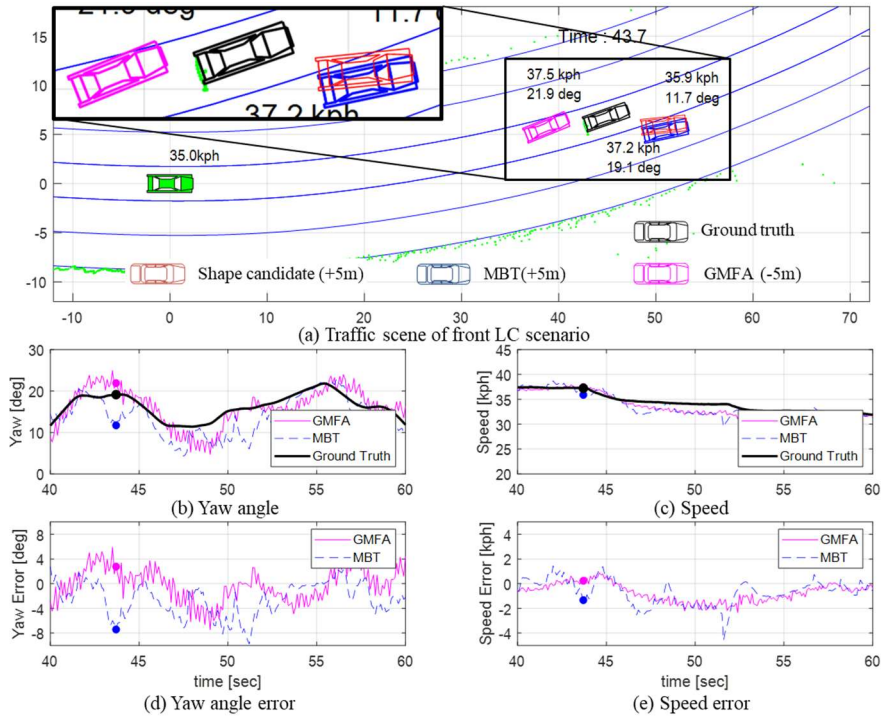


Figure 5.14. Yaw and speed estimated by GMFA with EKF and MBT in front LC scenarios.

The differences in the estimated accuracy of yaw angle are shown in Figure 5.14 (b) and Figure 5.14 (d). The standard deviation of MBT is larger than the proposed approach because the error in yaw angle greater as 44 s and 51 s in Figure 5.14 (b) and Figure 5.14 (d), respectively. The error has been caused the results of the shape extraction as indicated by the red vehicle in Figure 5.14 (a). In this sense, the shape extraction using current step's point cloud limits the accuracy of the yaw angle of the extracted shape, as shown Figure 2.2. The extracted shape varied from the actual yaw by more than 10°, as shown in Figure 5.14 (a). A significant deviation of shape extraction from the true value

results in inaccurate estimation of MBT. However, GMFA with EKF estimates the yaw angle via the direction of cluster movement as compared with the consecutive cluster, which explains the higher accuracy as compared with MBT.

So far, the estimated performance of MBT and GMFA with EKF implemented in Labview/MATLAB environment has been compared. Since the environment is specialized for rapid prototyping and is unsuitable for optimization of computation speed, we implemented the same algorithms in ROS/c++ environment and developed GMFA with PF to improve estimation performance. From now on, we will analyze the estimation performance of the algorithm implemented in ROS/c++ environment.

To obtain estimation performance in ROS/c++ environment, the driving data is collected from FMTC using the White Ioniq described in Section 1.2 and RT/range. The measured data has been collected at a speed of up to 40kph, and it is made up of more severe lane changes and overtaking than Labview/MATLAB environmental data. At ROS/c++ environment, all measured points is used without ROI and STOM grid size is set to 0.1m. Measuring frequency of Velodyne and Ibeo is set to 20Hz and 25Hz, respectively. Figure 5.15 depicts driving data collected by White Ioniq.

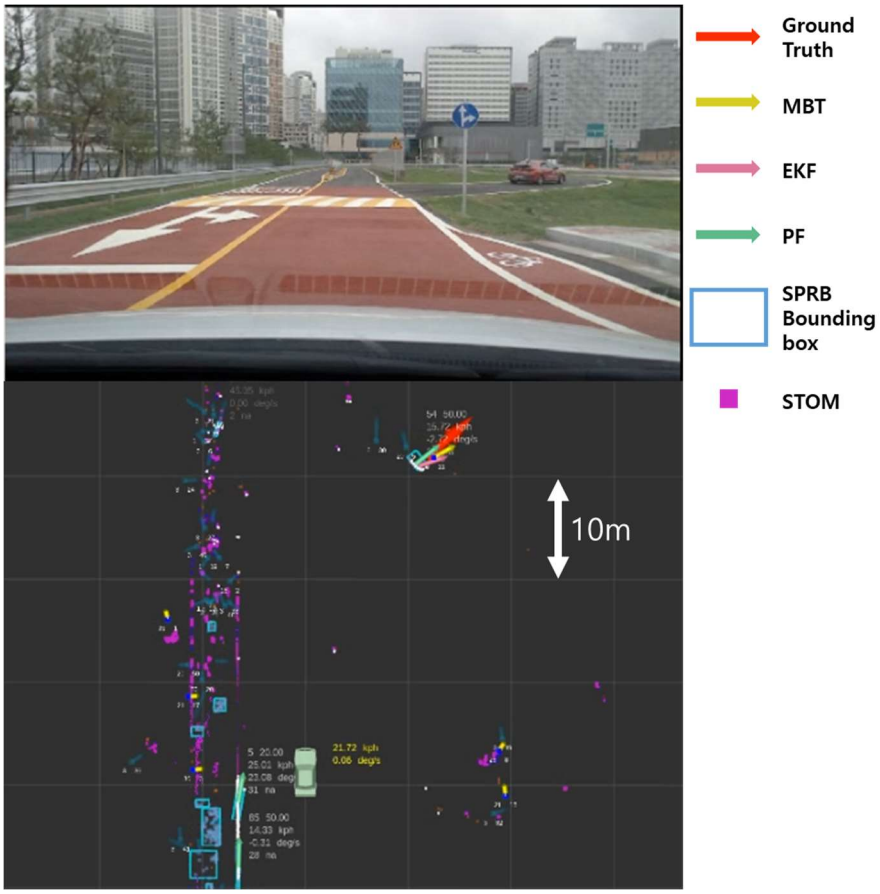


Figure 5.15. FMTC driving data collected using White Ioniq and RT-range from bird's eye view.

Red, yellow, pink, and green arrows represent ground truth through RT-range, MBT, GMFA with EKF, and GMFA with PF, respectively, and light blue boxes and dots represent SPRB tracked through PF. Boxes are presented to distinguish the tracked SPRB not estimated.

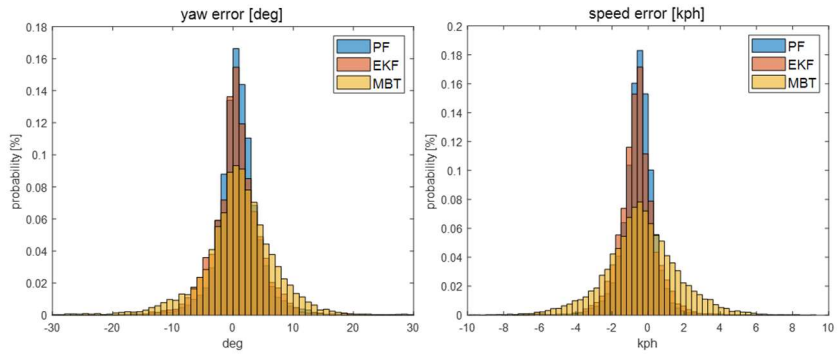


Figure 5.16. Estimation error distribution at ROS/c++ environment.

Figure 5.16 shows the distribution of the estimated error of yaw and velocity in the entire reference data. EKF refers to GMFA with EKF and PF refers to GMFA with PF. All estimators estimate without bias, and it can be seen that the estimation of PF is the most accurate. This fact can be compared quantitatively through Table 5.4.

TABLE 5.4. ESTIMATION RESULTS ACCORDING TO DISTANCE AT ROS

Standard Deviation of Error	Range [m]	[-30 , 30]	[30 , 60]	[60 , 90]	Total
Yaw [deg]	MBT	6.11	6.35	-	6.21
	EKF	4.66	4.96	5.30	4.84
	PF	**3.60**	**3.24**	**4.77**	**3.61**
Speed[kph]	MBT	1.97	2.11	-	2.02
	EKF	1.17	0.77	**0.76**	1.03
	PF	**0.98**	**0.69**	0.95	**0.92**

Table 5.4 represents the estimation performance of each estimator according to the relative distance of the target. In MBT, the variance is blank about the

area far then 60m because tracking failed in the area. PF's estimation performance is the best in all regions except for the speed estimation result in the region of over 60m. Based on the total accumulated data, the yaw and speed estimation performance of PF improved by 25% and 10%, respectively, compared to the estimation performance of EKF. The reason for this improvement in performance can be confirmed through analysis of each data.

Figure 5.17 and Figure 5.18 present the estimation performance of yaw and speed in a lane change situation. The black, blue, orange, and yellow lines represent the results of RT-range, PF, EKF, and MBT, respectively. It is clear that PF and EKF have a much more stable performance than MBT. When comparing the performance of PF and EKF, the most noticeable thing is the reduction in estimation delay for yaw. In both figures, we can see reducing the delay, as indicated by the black arrow. It can be seen that there is a delay reduction of about 1.3 seconds every lane change. Since the speed is estimated through the distance traveled in the yaw direction, the yaw delay reduction improves not only the yaw but also the speed estimation performance.

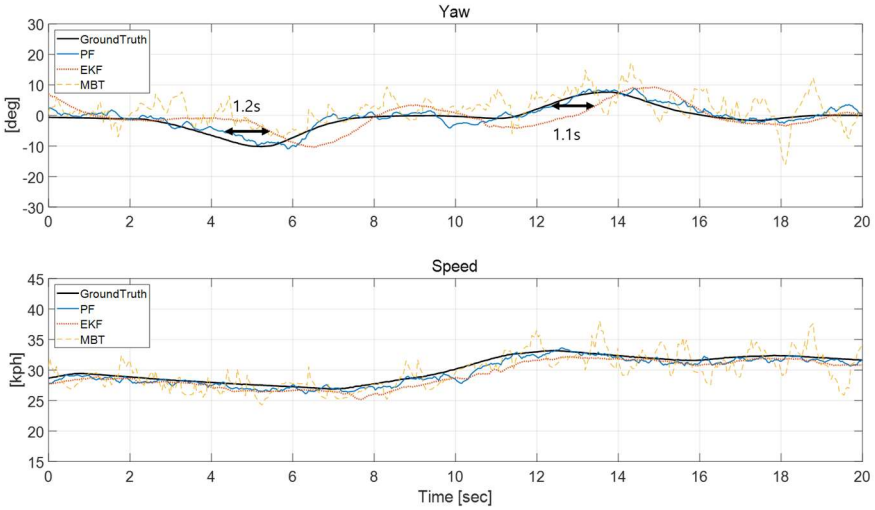


Figure 5.17. Estimation results according to time about LC scenario 1.

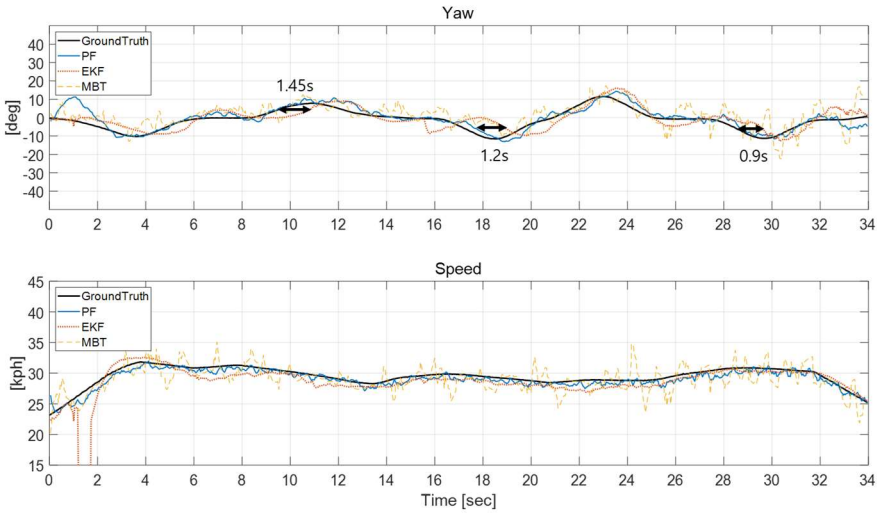


Figure 5.18. Estimation results according to time about LC scenario 2.

Improved estimation accuracy and reduced delay have a significant effect on autonomous driving. Figure 5.19 depicts the estimated states in LC Scenario

3 and the lane of the target vehicle estimated from the states. When the Time to Lane Crossing (TLC) is less than 3.0 seconds or target is in the ego's lane, the lane number is 0. Lane number 1 means that the target is in the lane next to the ego vehicle. In the case of MBT, it can be seen that the lane number vibrates due to the noisy estimation of the yaw angle. When false alarms are frequent, a target in the next lane is misrecognized as a target in the ego lane, then a false deceleration occurs.

When the target changes lane from the ego lane to the next lane, the target's lane is determined using the position for safety. From 4 and 15 seconds in Figure 5.19, both PF and EKF determine the target is in the next lane at a similar time. However, when the lane number changes from 1 to 0, when the target is cut in from the next lane to the ego lane, it is necessary to preemptively determine through TLC to enable comfortable autonomous driving. The TLC estimation is affected by the yaw angle of the target. The cut-in detection using PF is faster than EKF about 1 second by reducing the yaw angle estimation delay at 11 and 23 seconds of Figure 5.19. If quick cut-in detection is possible, it is possible to maintain a safe distance with a small acceleration to make a more comfortable and safe autonomous driving.

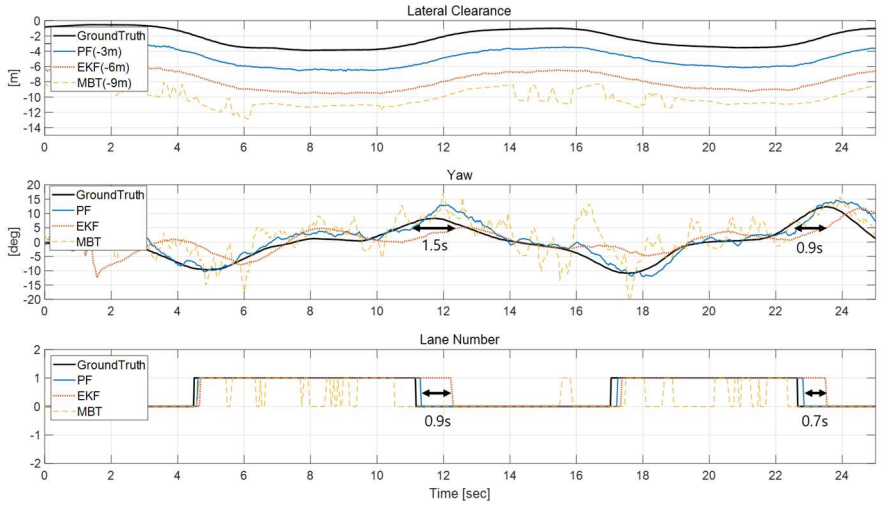


Figure 5.19. Target's estimated states and lane number at LC scenario 3.

Figure 5.20 depicts clearance and track ID in a scenario of overtaking a target. For clearance, the longitudinal position of the nearest point on the surface of the target is expressed in the ego vehicle coordinate system. The reference data is kept constant at 0m for a while because the nearest point on the target surface changes while passing through the target. From this data, it can be seen that GMFA tracks in a wider area than MBT. This is because when the distance increases, the number of points to the target becomes three or less, and it is impossible to extract the shape of the vehicle using these points.

Another thing to note is the phenomenon that the EKF loses its target around 26 seconds. At this time, the track ID is identical before and after; it means that the target estimation accuracy becomes poor beyond as much as the detection result is not accepted for the target. The location where this phenomenon

occurred is a distance of 20 ~ 30m based on the ego vehicle. The shape of the target changes from I-shape to L-shape as the ego vehicle changes lanes to pass in the range. Therefore, it can be seen that the ICP registration failure due to this shape change, and as a result, the estimation performance deteriorates. However, in the case of PF, the most suitable shape is estimated among the physically possible shapes, so even if the shape changes rapidly, there is no error like ICP.

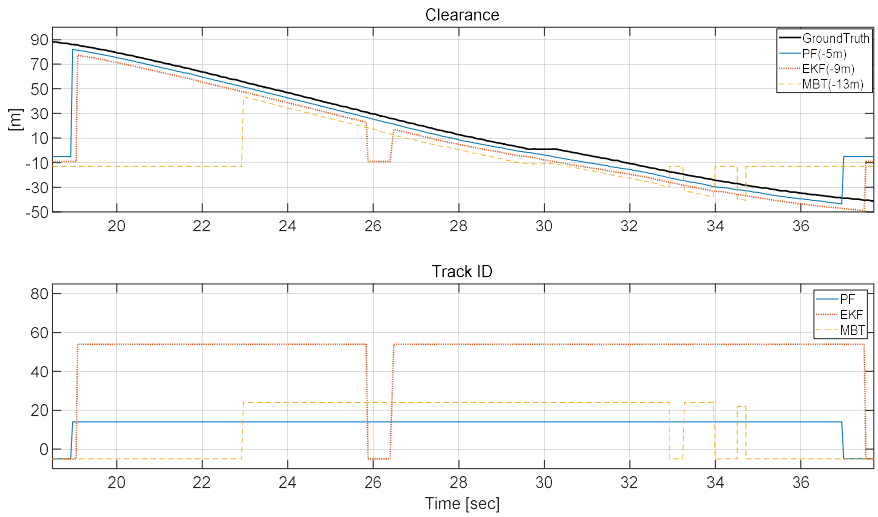


Figure 5.20. Clearance and Track ID at overtaking scenario.

On the other hand, it can be seen in the overtaken scenario that failure due to a yaw angle error does not occur in the same area in Figure 5.21. The displacement due to the shape change is dominant in the overtaking scenario because the speed of the target is as low as 15kph, but in the overtaken scenario, the displacement due to movement is more dominant than the displacement due

to the shape change, as the speed of the target is as fast as 40kph.

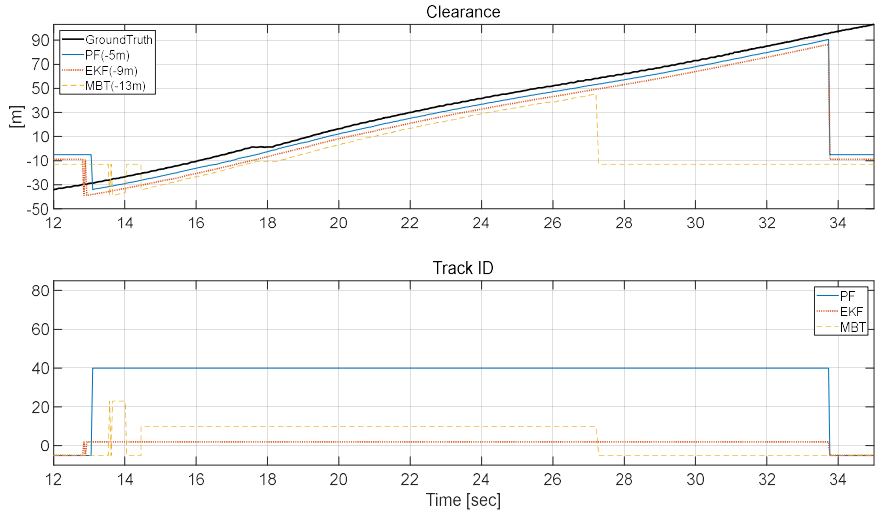


Figure 5.21. Clearance and Track ID at overtaken scenario.

From now on, we compare GMFA-PF with an algorithm using Deep Learning, which has achieved outstanding achievements in detection. Deep Neural Network (DNN) used as a control in this study was developed by Yu['20] using the KITTI benchmark.

First of all, we check the detection range. Figure 5.22 and Figure 5.23 depict that DNN only detected the target from -10m to 35m. Since the white Ioniq sensor configuration is different from KITTI, DNN only detects in such a short-range. Since the KITTI benchmark is acquired through the HDL-64E, 3D features can be measured in a wider area. Therefore, DNN trained using the data set detects using 3D features. However, a low-cost sensor installed in the white Ioniq can measure 3D features in the short-range. Therefore, DNN detects

in a short-range where 3D features are measured.

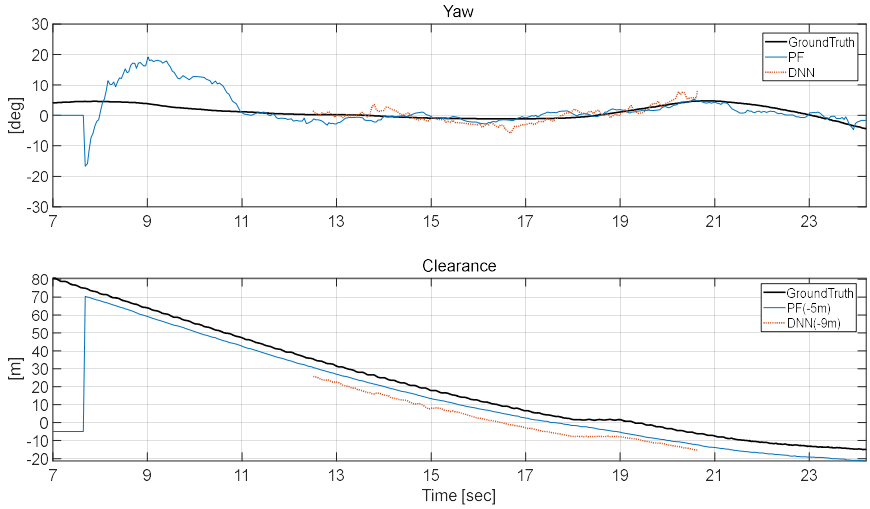


Figure 5.22. DNN's clearance and yaw estimation performance at overtaking scenario.

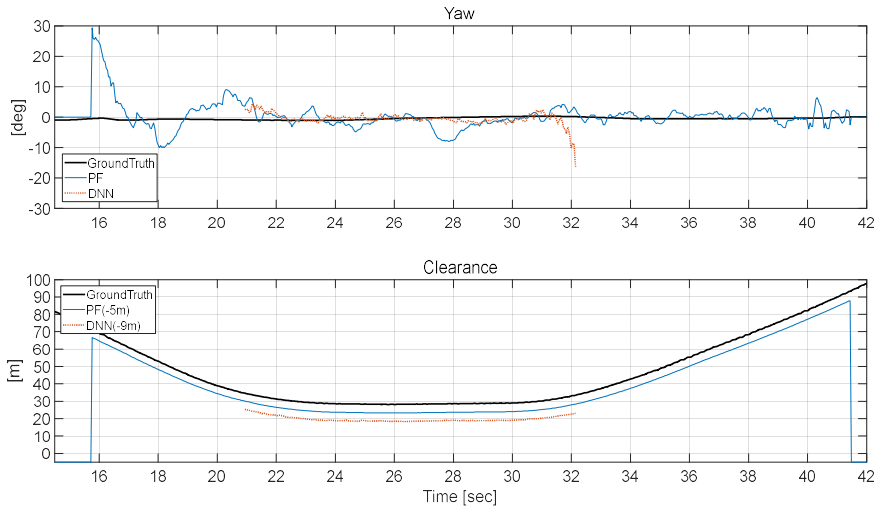


Figure 5.23. DNN's clearance and yaw estimation performance at approaching scenario.

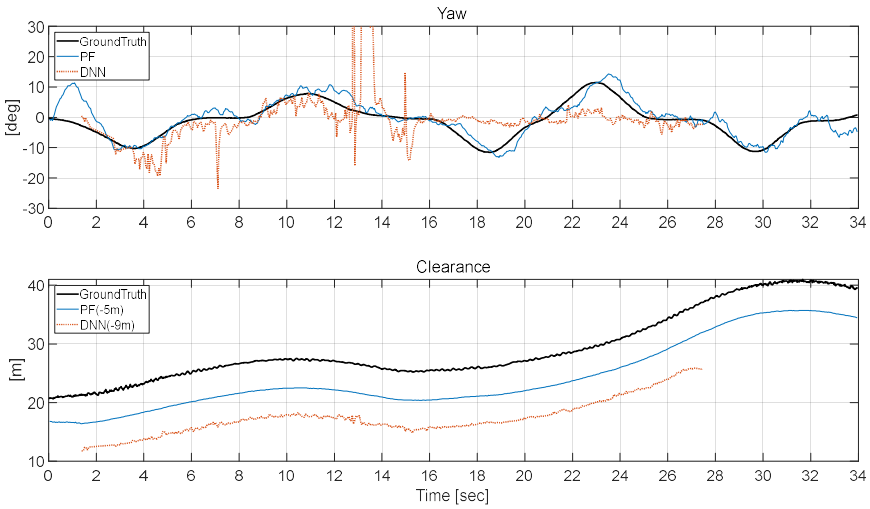


Figure 5.24. DNN's clearance and yaw estimation performance about forward lane changing target.

Figure 5.24 shows the estimation performance of GMFA with PF and DNN for lane changing target at 20-40m ahead. At 2 to 6 seconds, for lane changing target close to 25m, the yaw angle is fairly estimated by DNN. However, data from 16 seconds to 26 seconds indicate that the yaw angle of lane change occurring at a distance greater than 25 m is not estimated. In addition, it can be seen that a 20 degrees or more yaw angle estimation error has occurred through the peak data of 7 seconds, 13 seconds, and 15 seconds. The cause of this error shows the limitation of estimating the yaw angle of the target through the instantaneous shape in the sparse point cloud. In fact, it is difficult for even humans to estimate the yaw angle with sufficient accuracy using a moment scan as Figure 5.25. From this result, it can be confirmed that GMFA using the track before detect framework has strengths in sparse point cloud.

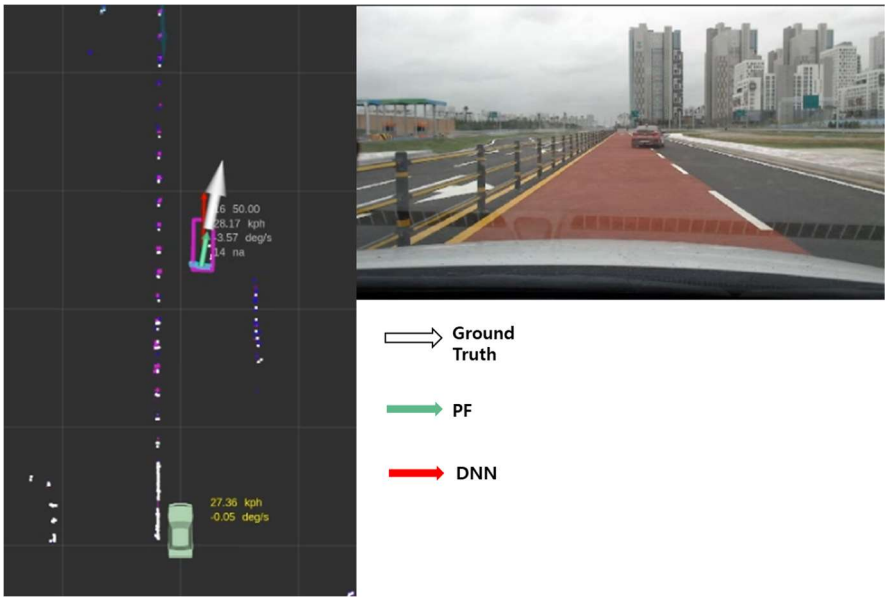


Figure 5.25. DNN and PF for front lane changing target.

5.3 Calculation Time

TABLE 5.5. CALCULATION TIME AT LABVIEW/MATLAB

Time/ Frame	Data Parsing	STOM predict	Cluster-ing	GMFA w EKF	STOM update	Total
Mean [ms]	6.3	7.3	5.6	30.6	4.3	54.1
Max [ms]	11.3	15.6	13.3	80.2	9.2	118.2

All the experiments and simulation tests at Labview/MATLAB have been conducted with an Intel Core i7-4790 3.20GHz CPU at MATLAB R2018a. As shown in Table 5.5, the Calculation time per frame for Nambu-Beltway and

SNU campus datasets confirmed the results of detection. The first frame of each dataset was excluded from the calculation time because of the need for additional time for initialization. Therefore, the computation time was analyzed for a total of 494 frames. The average and maximum values are shown. Table 5.5 displays the calculation time of the sub-functions in a sequential order. Since the scan frequency of Ibeo 2010 Lux used at Labview/MATLAB is 12.5Hz (80ms), the feasibility of real-time application of the proposed approach for each scan without additional optimization, parallel computation, or ROI selection in the above environment is proved.

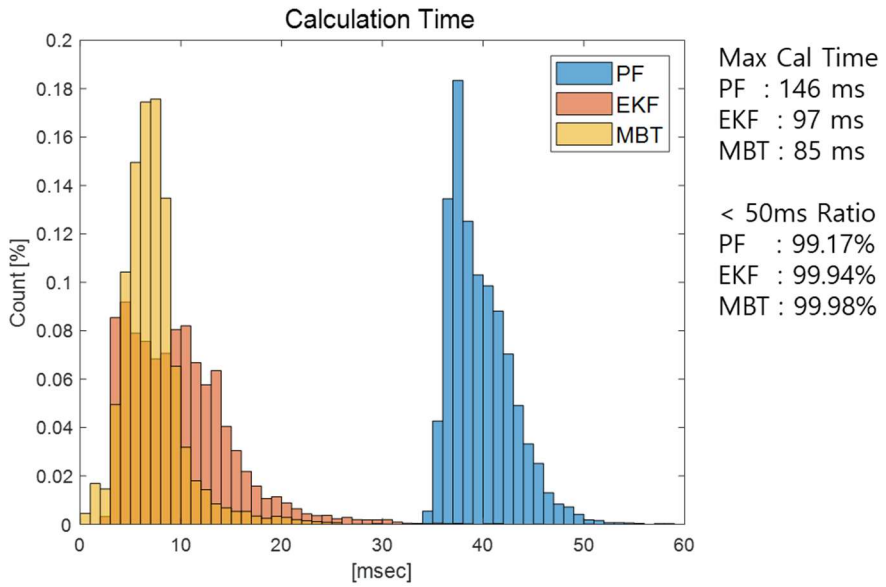


Figure 5.26. Calculation Time at ROS/c++.

The calculation times of the three algorithms have been compared in the ROS/c++ environment. Figure 5.26 shows the calculation time as a histogram

when calculating urban data on the i7-9700K 3.60GHz CPU. It can be seen that the MBT with a small amount of calculation is the fastest and the PF requires the most time. However, since both EKF and PF complete within 50ms more than 99% processing, which is the update period of velodyne, it can be confirmed that the real-time requirements are satisfied. Next to Figure 5.26, the maximum computation time and the percentage of computations completed within 50ms are depicted. Since the proposed module is implemented as a user process in the General Purpose Operating System, it can be interrupted at any time by other processes or operating system. Therefore, if additional optimization is achieved in the Real Time Operating System, the 0.9% timeout could be sufficiently handled.

Chapter 6. Conclusion & Future Works

6.1 Conclusion

In order to improve the efficiency of moving object detection and tracking in the urban environment using LiDAR, a novel method based on the interaction between STOM and GMFA has been developed. All points are efficiently handled in the urban environment without ROI selection, HD map, and other sensors. STOM is constructed rapidly and accurately in the local coordinate system by eliminating the interference of the moving objects using interaction with GMFA. The points of moving object are classified via STOM to suppress the detection and tracking failure due to static obstacles, and tracked using the correspondence between consecutive scans via GMFA. In addition, since tracking has been achieved without assuming the shape of the moving object, detection and tracking for various moving objects encountered in the urban environment is successfully performed. The yaw and speed are estimated with high accuracy via correspondence of points in consecutive scans.

In fact, the F_1score is improved by 25% and the standard deviation of the yaw angle error is reduced by 58%, as compared with the MBT using shape assumption. GMFA shows robust tracking even in sparse point clouds far than 60m as compared with the MBT and DNN. In addition, EKF's shortcomings are supplemented via PF, which improves the yaw angle and speed estimation

performance by 25% and 10%, and significantly reduced the delay to estimate yaw. The improvement enables the autonomous vehicle to cope with cut-in vehicles preemptively. At the same time, since the computation time per frame is shorter than the LiDAR scan period with 99%, the proposed approach complied with real-time requirements. In the problem definition, we summarized three requirements: real-time, modularity, and sparse point cloud. The proposed perception module satisfies all of these.

6.2 Future Works

The proposed perception module imitates pointwise tracking by a efficient calculation. Despite the approximations, it can be seen that the PF uses the maximum amount of computation with real-time performance. Thanks to the development of parallel computing, STOM and GMFA have become parallel, and as a result, there is room to upgrade at STOM aspect computation. Therefore, if simultaneous localization and mapping is applied to the prediction (currently using dead reckoning) by utilizing this margin, it is expected that both the estimation performance of the STOM and the ego vehicle's state will be improved while maintaining real-time characteristics. In addition, if other sensors such as radar and front cam are also fused in parallel, it will be possible to detect and estimate using various sensor information while maintaining real-time requirements. Finally, considering the proposed perception module as

pointwise tracking, the module adds speed to each LiDAR point. Therefore, the LiDAR with the proposed module can be thought of as a 5D (3D position and 2D velocity) LiDAR, and a learning-based detection algorithm using the module's output could be developed. The detection network using points and point's velocity is expected to improve performance than the existing network using only points.

Bibliography

ADAMS, M. D. (2001). "On-line gradient based surface discontinuity detection for outdoor scanning range sensors." *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, IEEE.

AINSLIE, M. A. (2010). "Principles of sonar performance modelling." Springer.

AUTONET2030 (2014). "Co-operative Systems in Support of Networked Automated Driving by 2030." 2016.

AZIM, A. and O. AYCARD (2012). "Detection, classification and tracking of moving objects in a 3D environment." *2012 IEEE Intelligent Vehicles Symposium*, IEEE.

BERGMAN, N. (1999). "Recursive Bayesian estimation: Navigation and tracking applications." Linköping University.

BERRIO, J. S., J. WARD, S. WORRALL, W. ZHOU and E. NEBOT (2017). "Fusing lidar and semantic image information in octree maps." *ACRA Australasian Conference on Robotics and Automation 2017*.

BHALLA, K., M. SHOTTEN, A. COHEN, M. BRAUER, S. SHAHRAZ, R. BURNETT, K. LEACH-KEMON, G. FREEDMAN and C. J. MURRAY (2014). "Transport for health: the global burden of disease from motorized road transport." The World Bank.

BHATTACHARYYA, P. and K. CZARNECKI (2020). "Deformable PV-RCNN: Improving 3D Object Detection with Learned Deformations." *arXiv preprint arXiv:2008.08766*.

BLACKMAN, S. and R. POPOLI (1999). "Design and analysis of modern tracking systems(Book)." *Norwood, MA: Artech House, 1999.*

BOEHME, A. and H. BALD (2013). "Motor vehicle having a parking

assistance system." Google Patents.

BøRCS, A., B. NAGY, M. BATICZ and C. BENEDEK (2014). "A model-based approach for fast vehicle detection in continuously streamed urban LIDAR point clouds." *Asian Conference on Computer Vision*, Springer.

BøRCS, A., B. NAGY and C. BENEDEK (2017). "Instant object detection in LiDAR point clouds." *IEEE Geoscience and Remote Sensing Letters* 14(7): 992-996.

BOUZOURAA, M. E. and U. HOFMANN (2010). "Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors." *2010 IEEE Intelligent Vehicles Symposium*, IEEE.

BRESENHAM, J. E. (1965). "Algorithm for computer control of a digital plotter." *IBM Systems journal* 4(1): 25-30.

BUEHLER, M., K. IAGNEMMA and S. SINGH (2009). "The DARPA urban challenge: autonomous vehicles in city traffic." springer.

CHEN, X., H. MA, J. WAN, B. LI and T. XIA (2017). "Multi-view 3d object detection network for autonomous driving." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

CHENG, J., Z. XIANG, T. CAO and J. LIU (2014). "Robust vehicle detection using 3d lidar under complex urban environment." *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE.

CHO, H., Y.-W. SEO, B. V. KUMAR and R. R. RAJKUMAR (2014). "A multi-sensor fusion system for moving object detection and tracking in urban driving environments." *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE.

CHUM, O., J. MATAS and J. KITTLER (2003). "Locally optimized RANSAC." *Joint Pattern Recognition Symposium*, Springer.

COLLINS, T. and J. COLLINS (2007). "Occupancy grid mapping: An

empirical evaluation." *2007 Mediterranean Conference on Control & Automation*, IEEE.

DICKMANN, J., J. KLAPPSTEIN, M. HAHN, N. APPENRODT, H.-L. BLOECHER, K. WERBER and A. SAILER (2016). "Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding." *2016 IEEE Radar Conference (RadarConf)*, IEEE.

DOMINGUEZ, R., E. ONIEVA, J. ALONSO, J. VILLAGRA and C. GONZALEZ (2011). "LIDAR based perception solution for autonomous vehicles." *2011 11th International Conference on Intelligent Systems Design and Applications*, IEEE.

DOUILLARD, B., J. UNDERWOOD, N. KUNTZ, V. VLASKINE, A. QUADROS, P. MORTON and A. FRENKEL (2011). "On the segmentation of 3D LIDAR point clouds." *2011 IEEE International Conference on Robotics and Automation*, IEEE.

ESTER, M., H.-P. KRIEGEL, J. SANDER and X. XU (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." *Kdd*.

FERRI, F., M. GIANNI, M. MENNA and F. PIRRI (2015). "Dynamic obstacles detection and 3d map updating." *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

GAUS, H., G. FRANKE and W. STAHL (1996). "Short range ultrasonic distance warning system for motor vehicle." Google Patents.

GEIGER, A., P. LENZ and R. URTASUN (2012). "Are we ready for autonomous driving? the kitti vision benchmark suite." *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE.

GILHOLM, K., S. GODSILL, S. MASKELL and D. SALMOND (2005). "Poisson models for extended target and group tracking." *Signal and Data Processing of Small Targets 2005*, International Society for

Optics and Photonics.

GRANSTROM, K., M. BAUM and S. REUTER (2016). "Extended object tracking: Introduction, overview and applications." *arXiv preprint arXiv:1604.00970*.

GRANSTRøM, K., C. LUNDQUIST and U. ORGUNER (2011). "Tracking rectangular and elliptical extended targets using laser measurements." *14th International Conference on Information Fusion*, IEEE.

GRANSTRøM, K., S. REUTER, D. MEISSNER and A. SCHEEL (2014). "A multiple model PHD approach to tracking of cars under an assumed rectangular shape." *17th International Conference on Information Fusion (FUSION)*, IEEE.

GRISETTI, G., C. STACHNISS and W. BURGARD (2007). "Improved techniques for grid mapping with rao-blackwellized particle filters." *IEEE transactions on Robotics* 23(1): 34-46.

GU, X., Y. WANG, C. WU, Y. J. LEE and P. WANG (2019). "Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

HE, C., H. ZENG, J. HUANG, X.-S. HUA and L. ZHANG (2020). "Structure Aware Single-stage 3D Object Detection from Point Cloud." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

HE, M., Q. ZHU, Z. DU, H. HU, Y. DING and M. CHEN (2016). "A 3D shape descriptor based on contour clusters for damaged roof detection using airborne LiDAR point clouds." *Remote Sensing* 8(3): 189.

HERSHBERGER, J. and J. SNOEYINK (1994). "An O (n log n) implementation of the Douglas-Peucker algorithm for line simplification." *Proceedings of the tenth annual symposium on Computational geometry*.

HO, B.-J., P. SODHI, P. TEIXEIRA, M. HSIAO, T. KUSNUR and M. KAESS (2018). "Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3D environments." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

JESSUP, J., S. N. GIVIGI and A. BEAULIEU (2014). "Merging of octree based 3d occupancy grid maps." *2014 IEEE International Systems Conference Proceedings*, IEEE.

JO, K., S. LEE, C. KIM and M. SUNWOO (2019). "Rapid Motion Segmentation of LiDAR Point Cloud Based on a Combination of Probabilistic and Evidential Approaches for Intelligent Vehicles." *Sensors* 19(19): 4116.

KAESTNER, R., J. MAYE, Y. PILAT and R. SIEGWART (2012). "Generative object detection and tracking in 3d range data." *2012 IEEE International Conference on Robotics and Automation*, IEEE.

KOHLBRECHER, S., O. VON STRYK, J. MEYER and U. KLINGAUF (2011). "A flexible and scalable slam system with full 3d motion estimation." *2011 IEEE international symposium on safety, security, and rescue robotics*, IEEE.

KONOLIGE, K., M. AGRAWAL, M. R. BLAS, R. C. BOLLES, B. GERKEY, J. SOLA and A. SUNDARESAN (2009). "Mapping, navigation, and learning for off-road traversal." *Journal of Field Robotics* 26(1): 88-113.

KOSIS, S. K. (2018). "Causes of road traffic accidents (driver's aspect, population over 19 years old)." KOSIS.

KULKARNI, A. S. (2020). "Motion Segmentation for Autonomous Robots Using 3D Point Cloud Data."

KUTILA, M., P. PYYKøNEN, W. RITTER, O. SAWADE and B. SCHæUFELE (2016). "Automotive LIDAR sensor development

scenarios for harsh weather conditions." *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE.

LAM, J., K. KUSEVIC, P. MRSTIK, R. HARRAP and M. GREENSPAN (2010). "Urban scene extraction from mobile ground based lidar data." *Proceedings of 3DPVT*.

LEE, H., H. CHAE and K. J. I.-P. YI (2019). "A Geometric Model based 2D LiDAR/Radar Sensor Fusion for Tracking Surrounding Vehicles." 52(8): 130-135.

LEE, J.-S., C. KIM and W. K. CHUNG (2010). "Robust RBPF-SLAM using sonar sensors in non-static environments." *2010 IEEE International Conference on Robotics and Automation*, IEEE.

LEE, S., C. KIM, S. CHO, M. SUNWOO and K. JO (2020). "Robust 3-Dimension Point Cloud Mapping in Dynamic Environment using Point-wise Static Probability-based NDT Scan-matching." *IEEE Access*.

LI, B. (2019). "On Enhancing Ground Surface Detection from Sparse Lidar Point Cloud." *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

LI, Y., B. YONG, H. WU, R. AN and H. XU (2015). "Road detection from airborne LiDAR point clouds adaptive for variability of intensity data." *Optik* 126(23): 4292-4298.

LIKAS, A., N. VLASSIS and J. J. VERBEEK (2003). "The global k-means clustering algorithm." *Pattern Recognition* 36(2): 451-461.

LIU, J. S. and R. CHEN (1998). "Sequential Monte Carlo methods for dynamic systems." *Journal of the American statistical association* 93(443): 1032-1044.

LIU, K., W. WANG, R. THARMARASA and J. WANG (2018). "Dynamic Vehicle Detection With Sparse Point Clouds Based on PE-CPD." *IEEE*

Transactions on Intelligent Transportation Systems(99): 1-14.

MAGNIER, V., D. GRUYER and J. GODELLE (2017). "Automotive LIDAR objects detection and classification algorithm using the belief theory." *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE.

MEN, H., B. GEBRE and K. POCHIRAJU (2011). "Color point cloud registration with 4D ICP algorithm." *2011 IEEE International Conference on Robotics and Automation*, IEEE.

MERDRIGNAC, P., E. POLLARD and F. NASHASHIBI (2015). "2D laser based road obstacle classification for road safety improvement." *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, IEEE.

MERTZ, C., L. E. NAVARRO-SERMENT, R. MACLACHLAN, P. RYBSKI, A. STEINFELD, A. SUPPE, C. URMSON, N. VANDAPEL, M. HEBERT and C. THORPE (2013). "Moving object detection with laser scanners." *Journal of Field Robotics* 30(1): 17-43.

MEYER-DELIUS, D., M. BEINHOFER and W. BURGARD (2012). "Occupancy Grid Models for Robot Mapping in Changing Environments." *AAAI*.

MOOSMANN, F. and T. FRAICHARD (2010). "Motion estimation from range images in dynamic outdoor scenes." *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE.

MORAS, J., V. CHERFAOUI and P. BONNIFAIT (2011). "Credibilist occupancy grids for vehicle perception in dynamic environments." *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE.

NA, S., L. XUMIN and G. YONG (2010). "Research on k-means clustering algorithm: An improved k-means clustering algorithm." *2010 Third International Symposium on intelligent information technology and security informatics*, IEEE.

PANG, S., D. MORRIS and H. RADHA (2020). "CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection." *arXiv preprint arXiv:2009.00784*.

PETROVSKAYA, A. and S. THRUN (2008). "Model based vehicle tracking for autonomous driving in urban environments." *Proceedings of robotics: science and systems IV, Zurich, Switzerland* 34.

PETROVSKAYA, A. and S. THRUN (2009). "Model based vehicle detection and tracking for autonomous urban driving." *Autonomous Robots* 26(2-3): 123-139.

POMARES, A., J. L. MARTíNEZ, A. MANDOW, M. A. MARTíNEZ, M. MORãN and J. MORALES (2018). "Ground extraction from 3D lidar point clouds with the classification learner app." *2018 26th Mediterranean Conference on Control and Automation (MED)*, IEEE.

RAMER-DOUGLAS-PEUCKER, N.-P. (1972). "Ramer-Douglas-Peucker algorithm."

RHODES, A., E. KIM, J. A. CHRISTIAN and T. EVANS (2016). "LIDAR-based relative navigation of non-cooperative objects using point Cloud Descriptors." *AIAA/AAS Astrodynamics Specialist Conference*.

ROS, G., S. RAMOS, M. GRANADOS, A. BAKHTIARY, D. VAZQUEZ and A. M. LOPEZ (2015). "Vision-based offline-online perception paradigm for autonomous driving." *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE.

ROUMELIOTIS, S. I. and G. A. BEKEY (2000). "Segments: A layered, dual-kalman filter algorithm for indoor feature extraction." *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, IEEE.

SABATINI, S., M. CARNO, S. FIORENTI and S. M. SAVARESI (2018). "Improving Occupancy Grid Mapping via Dithering for a Mobile Robot Equipped with Solid-State LiDAR Sensors." *2018 IEEE*

Conference on Control Technology and Applications (CCTA), IEEE.

SALTI, S., F. TOMBARI and L. DI STEFANO (2014). "SHOT: Unique signatures of histograms for surface and texture description." *Computer Vision and Image Understanding* 125: 251-264.

SCHEEL, A., S. REUTER and K. DIETMAYER (2016). "Using separable likelihoods for laser-based vehicle tracking with a labeled multi-Bernoulli filter." *2016 19th International Conference on Information Fusion (FUSION)*, IEEE.

SCHMID, M. R., M. MAEHLISCH, J. DICKMANN and H.-J. WUENSCHE (2010). "Dynamic level of detail 3d occupancy grids for automotive use." *2010 IEEE Intelligent Vehicles Symposium*, IEEE.

SCHNABEL, R., R. WAHL and R. KLEIN (2007). "Efficient RANSAC for point-cloud shape detection." *Computer graphics forum*, Wiley Online Library.

SCHUELER, K., T. WEIHERER, E. BOUZOURAA and U. HOFMANN (2012). "360 degree multi sensor fusion for static and dynamic obstacles." *Intelligent Vehicles Symposium (IV), 2012 IEEE*, IEEE.

SHAN, J. and S. APARAJITHAN (2005). "Urban DEM generation from raw LiDAR data." *Photogrammetric Engineering & Remote Sensing* 71(2): 217-226.

SHAN, T. and B. ENGLOT (2018). "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

SHAN, T., B. ENGLOT, D. MEYERS, W. WANG, C. RATTI and D. RUS (2020). "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping." *arXiv preprint arXiv:2007.00258*.

SHI, S., C. GUO, L. JIANG, Z. WANG, J. SHI, X. WANG and H. LI (2020). "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection."

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

SIMON, D. (2006). "Optimal state estimation: Kalman, H infinity, and nonlinear approaches." John Wiley & Sons.

SINGH, S. (2015). "Critical reasons for crashes investigated in the national motor vehicle crash causation survey."

TONGTONG, C., D. BIN, L. DAXUE, Z. BO and L. QIXU (2011). "3D LIDAR-based ground segmentation." *The First Asian Conference on Pattern Recognition*, IEEE.

URMSON, C., J. A. BAGNELL, C. BAKER, M. HEBERT, A. KELLY, R. RAJKUMAR, P. E. RYBSKI, S. SCHERER, R. SIMMONS and S. SINGH (2007). "Tartan racing: A multi-modal approach to the darpa urban challenge."

USELMANN, D. J. and L. M. USELMANN (2004). "Sonic blind spot monitoring system." Google Patents.

VAN DE VEN, J., F. RAMOS and G. D. TIPALDI (2010). "An integrated probabilistic model for scan-matching, moving object detection and motion estimation." *2010 IEEE International Conference on Robotics and Automation*, IEEE.

VANDORPE, J., H. VAN BRUSSEL and H. XU (1996). "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder." *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE.

VAQUERO, V., E. REPISO and A. SANFELIU (2019). "Robust and real-time detection and tracking of moving objects with minimum 2D LIDAR information to advance autonomous cargo handling in ports." 19(1): 107.

VELAS, M., M. SPANEL, M. HRADIS and A. HEROUT (2018). "Cnn for very fast ground segmentation in velodyne lidar data." *2018 IEEE*

International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE.

VU, T.-D., O. AYCARD and N. APPENRODT (2007). "Online localization and mapping with moving object tracking in dynamic outdoor environments." *2007 IEEE Intelligent Vehicles Symposium*, IEEE.

WAHLSTRøM, N. and E. ÖZKAN (2015). "Extended target tracking using Gaussian processes." *IEEE Transactions on Signal Processing* 63(16): 4165-4178.

WALDSCHMIDT, C. and H. MEINEL (2014). "Future trends and directions in radar concerning the application for autonomous driving." *2014 11th European Radar Conference*, IEEE.

WANG, C.-C., C. THORPE, S. THRUN, M. HEBERT and H. DURRANT-WHYTE (2007). "Simultaneous localization, mapping and moving object tracking." *The International Journal of Robotics Research* 26(9): 889-916.

WANG, D. Z., I. POSNER and P. NEWMAN (2015). "Model-free detection and tracking of dynamic objects with 2D lidar." 34(7): 1039-1063.

WARREN, M. E. (2019). "Automotive LIDAR technology." *2019 Symposium on VLSI Circuits*, IEEE.

WHO, W. H. O. (2011). "Global launch: decade of action for road safety 2011-2020." World Health Organization.

WU, J., Y. TIAN, H. XU, R. YUE, A. WANG and X. SONG (2019). "Automatic ground points filtering of roadside LiDAR data using a channel-based filtering algorithm." *Optics & Laser Technology* 115: 374-383.

XU, L., C. FENG, V. R. KAMAT and C. C. J. A. I. C. MENASSA (2019). "An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments." 104: 230-245.

YE, Y., L. FU and B. LI (2016). "Object detection and tracking using multi-

layer laser for autonomous urban driving." *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE.

YOO, J. H., Y. KIM, J. S. KIM and J. W. CHOI (2020). "3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection." *arXiv preprint arXiv:2004.12636*.

YU, J. (2020). "Integration of Clustering-based Unlearned Object Detection and Deep Neural Network for Real-time Perception in Autonomous Driving System." Department of Mechanical Engineering. Seoul, Seoul National University.

YUE, Y., D. WANG, P. SENARATHNE and D. MORATUWAGE (2016). "A hybrid probabilistic and point set registration approach for fusion of 3D occupancy grid maps." *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE.

YUNFEI, B., L. GUOPING, C. CHUNXIANG, L. XIAOWEN, Z. HAO, H. QISHENG, B. LINYAN and C. CHAOYI (2008). "Classification of LIDAR point cloud and generation of DTM from LIDAR height and intensity data in forested area." *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 37(7): 313-318.

ZHANG, X., W. XU, C. DONG and J. M. DOLAN (2017). "Efficient L-shape fitting for vehicle detection using laser scanners." *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE.

ZHAO, H., J. SHA, Y. ZHAO, J. XI, J. CUI, H. ZHA and R. J. I. T. O. I. T. S. SHIBASAKI (2011). "Detection and tracking of moving objects at intersections using a network of laser scanners." 13(2): 655-670.

ZHENG, Z., Y. LI and W. JUN (2015). "LiDAR point cloud registration based on improved ICP method and SIFT feature." *2015 IEEE International Conference on Progress in Informatics and Computing*

(PIC), IEEE.

ZHONG, Q., Y. LIU, X. GUO and L. REN (2018). "Dynamic Obstacle Detection and Tracking Based on 3D Lidar." 22(5): 602-610.

ZHOU, B., Z. TANG, K. QIAN, F. FANG and X. MA (2017). "A LiDAR odometry for outdoor mobile robots using NDT based scan matching in GPS-denied environments." *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, IEEE.

초 록

차량용 2차원 라이다를 이용한 정지 장애물 맵과 기하학적 모델 프리 접근 방식 기반 실시간 주행 환경 인식

본 논문은 도심 자율주행을 위한 차량용 2차원 라이다와 샤시센서를 이용한 실시간 인식 모듈을 제안한다. 제안된 인식 모듈은 주행 환경을 정지 장애물과 이동 장애물로 구분하며, 이는 각각 정지 장애물 맵 (STOM) 과 트랙으로 표현한다. 본 모듈은 V2X나 HD map 과 같은 기반 시설의 정보를 활용하지 않기 때문에 어떤 도로에서나 활용 가능하다. 또한 이 모듈의 연산 시간은 자율주행 차량에 탑재된 다목적 컴퓨터에서 라이다의 측정 주기보다 짧기 때문에 실시간으로 활용이 가능하다.

제안된 모듈은 두 개의 병렬적 하위 모듈로 구성된다. 첫 번째는 정지 장애물 맵 구성 부분으로, 이 하위 모듈은 라이다와 자차량 정보, 이전 스텝의 트랙을 활용하여 정지 장애물을 2차원 확률 분포로 묘사한다. 정지 장애물을 각각에 대한 검출 없이 2차원 그리드 맵으로 표현하기 때문에, 정지 장애물 맵은 다양한 형상의 정지 장애물을 효과적으로 묘사한다. 움직이는 장애물을 추적 모듈과의 상호작용을 통해 구분하여 반복 연산을 통한 방식보다 효율적으로 동적 장애물이 있는 환경에서 정지 장애물 맵을 추정한다. 두 번째 하위 모듈은 형상 모델 프리 접근법 (GMFA)로, 이전 스탭의 정지 장애물 맵을 이용하여 이동 물체를 구분하고, 구분된 이동 물체를 추적

을 통해 검출 및 그 상태를 추정한다. 거리 기반 클러스터링 후에 연속된 스캔에서 형상을 고려한 4차원 공간에서 인접한 두 클러스터를 이용해서 트랙을 생성한다. 형상 가정 없이 클러스터의 형상을 직접 활용하여 클러스터를 비교하기 때문에 오토바이, 버스, 보행자, 승용차 등의 다양한 도로 이용자를 하나의 모델로 효율적으로 추적한다.

추적된 트랙에서 진행 방향, 속도 등의 상태를 추정하기 위해서 두 가지 방법이 제안되었다. 첫 번째 방식은 확장 칼만 필터를 이용하였다. 이 방식은 추적된 클러스터와 매칭된 클러스터를 반복적 최근접 점쌍 방식으로 가장 유사해지는 강체 이동 모션을 추출하고, 이를 측정으로 하는 확장 칼만 필터를 통해서 이동 장애물의 동적 상태를 추정한다. 두 번째 방식은 파티클 필터를 이용해 점구름을 직접 측정값으로 활용하였고, 그 결과 물체가 밀집된 상황 또는 형상 변화가 큰 상황에서 더 나은 성능을 보인다.

제안된 인식 모듈은 Labview/MATLAB 과 ROS/c++ 환경에서 각각 구현되었다. 각 환경에서 실제 자율주행 차량을 이용하여 취득한 데이터를 이용하여 정량적 그리고 정성적으로 평가되었다. 검출 성능은 전방 카메라를 이용한 라벨링 데이터를 활용하여 평가되었으며, 추정 성능은 RT-range를 이용한 데이터를 통해서 평가했다. 같은 연산량의 이전 알고리즘과 비교하여 도심환경에서 두 성능 모두 향상되었다.

주요어: 자율 주행 자동차, 움직이는 물체에 대한 검출 및 추적, 인식, 희박한 점구름, 모델 프리 추적, 라이다

학 번: 2016-23468