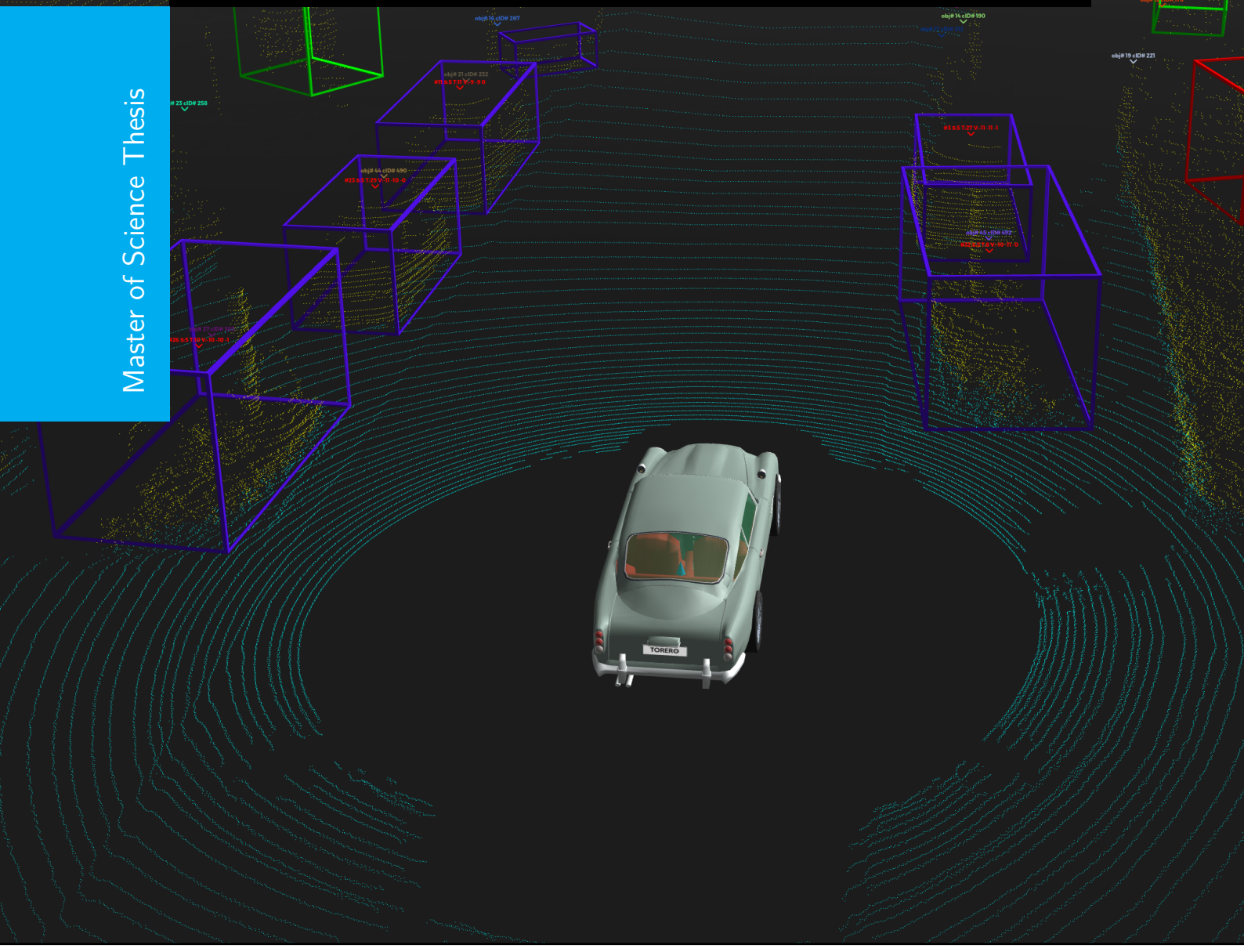
3D-LIDAR Multi Object Tracking for Autonomous Driving

Multi-target Detection and Tracking under Urban Road Uncertainties

A.S. Abdul Rachman

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

3D-LIDAR Multi Object Tracking for Autonomous Driving

Multi-target Detection and Tracking under Urban Road Uncertainties

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering, track Control Engineering at Delft University of Technology

A.S. Abdul Rachman

November 9, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this report was done in cooperation with Production Engineering of E-Mobility Components (PEM) Institute of RWTH Aachen University. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

3D-LIDAR Multi Object Tracking for Autonomous Driving

by

A.S. Abdul Rachman

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering, track Control Engineering

Dated: November 9, 2017

Supervisor(s):

Prof. Pascual Campoy-Cervera Supervisor

Mohsen Sefati, Msc. Second Supervisor

Reader(s):

dr.ir. Manuel Mazo-Espinosa, Jr.

Prof.dr.ir. Pieter P. Jonker

Abstract

The recent advancement of the autonomous vehicle has raised the need for reliable environmental perception. This is evident, as an autonomous vehicle has to perceive and interpret its local environment in order to execute reactive and predictive control action. Object Tracking is an integral part of vehicle perception, as it enables the vehicle to estimate surrounding objects trajectories to achieve dynamic motion planning. The 3D LIDAR has been widely used in object tracking research since the mechanically compact sensor provides rich, far-reaching and real-time data of spatial information around the vehicle. On the other hand, the development of autonomous driving is heading toward its use in the urban-driving situation. In an urban situation, a robust detection and tracking algorithm is required due to increasing number of Vulnerable Road User (e.g. pedestrian and cyclist), heterogeneous terrain, inherent measurement uncertainties and limited sensor reach.

This thesis presents an integrated framework of multi-target object detection and tracking using 3D LIDAR geared toward urban use. The framework combines occlusion-aware detection methods, probabilistic adaptive filtering and computationally efficient heuristics logic-based filtering to handle uncertainties arising from sensing limitation of 3D LIDAR and complexity of the target object movement. The implemented framework takes a raw 3D LIDAR data as input to perform multi-target object detection while simultaneously maintaining track of the detected objects' kinematic states and dimension in robust, causal, and real-time manner.

Robust detection is enabled by slope-based ground removal and L-shape fitting to reliably enclose object of interest into bounding box in the presence of sensor occlusion. The tracker utilises three combined Bayesian filters (IMM-UK-JPDAF) which simultaneously tackle association uncertainties, motion uncertainties and estimate non-linear stochastic motion model in real-time. Logic-based rule filters are also designed to augment the rest of detection and tracking based on the understanding of LIDAR sensor limitation and occlusion characteristic.

The evaluation results using real-world pre-recorded 3D LIDAR data show the proposed framework can achieve promising real-time tracking performance in urban situations. Diverse datasets are deliberately chosen to evaluate if the MOT system is capable of working in a varying driving scenario. The benchmark results highlight that the designed and implemented MOT system is performing on par with the state-of-art works and yield satisfying accuracy and precision in most given urban settings.

Table of Contents

List of Figures

List of Tables

Preface

This report rounds off the master thesis work carried out as partial fulfilment for the MSc. Control Engineering program at TU Delft. The master thesis research is conducted with cooperation with Autonomous Driving Group at PEM Institute RWTH Aachen.

The project hosted by PEM deals with the research of emerging state-of-art sector which intersects with the author's personal and professional interest: intelligent vehicle. It is an unmissable opportunity for the author to combine the prior academic background obtained at TU Delft with the chance of applying it at RWTH Aachen, another equally distinguished university. New environment, new challenges, new people and new findings: the experiences are simply invaluable.

It is a tremendous pleasure to work under the daily supervision of Mr. Mohsen Sefati whose guidance, support and feedback have helped me to shape my work and motivated me to be constantly improving. I also would like to give the highest appreciation to the advisory of Prof. Pascual Campoy, whose key remarks and observations throughout the thesis work have enabled me to add a concrete value and contributions to the end results. Their dedication and patience which go beyond the time and location barrier are hereby gratefully acknowledged.

I also would like to extend my gratitude to all other students and colleagues at Autonomous Driving Group PEM, for their support on technical implementation and helpful general guidance during my stay at Aachen. Last but not least, to all of my friends in Delft who have been keeping me good company through every moment; also to my families with their unwavering support and encouragement, I owe every and each of you my special thanks.

November 9, 2017

"You must let go of the illusion of control."

— *Master Oogway*

Chapter 1

Introduction

1-1 Defining Vehicle Autonomy

Autonomous driving can cover a wide gamut of definition based on its use case and degree of autonomy, Society of Automotive Engineer (SAE) Internationals provides a useful classification for the autonomous vehicle on its International Standard no. J3016[1], this classification notably has the equivalent comparison with other pre-existing local standards such as the one made by national transport regulatory bodies such as Germanys BASt and US NHTSA. Table 1-1 summarizes the classification and the equivalent level according to BaSt and NHTSA.

Level	Name	Narrative definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of *dynamic driving task*	System capability (*driving modes*)	BASt level	NHTSA level
***Human driver* monitors the driving environment**								
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a	Driver only	0
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes	Assisted	1
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes	Partially automated	2
***Automated driving system* ("system") monitors the driving environment**								
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes	Highly automated	3
4	**High Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes	Fully automated	3/4
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**	-	

Table 1-1: summary of levels of driving automation for on-road vehicles. Source:[2, 1]

According to SAE classifications, existing commercially available Advanced Driving Assists

(ADAs) such as Adaptive Cruise Control and Parking Assists fall on Level 3, in which human intervention is still needed as a fallback during the dynamic driving task. Moreover, these ADAs are designed for relatively low traffic and well-structured environment such as a highway. Limited autonomous driving in the more complex environment (mainly urban situation) is also commercially available, such as intersection assist, construction site guidance, and pedestrian collision prevention. The limited degree of autonomy of contemporary car demonstrates that up to this point urban autonomous driving is research in motion. Note that in this report we define an autonomous vehicle as the equivalent of SAE Level 5, which is in line with the common goal of the autonomous driving technology: fully remove human involvement in the traditional sense of driving task.

Due to the inherent high complexity of autonomous driving tasks, it is useful to compartmentalize the technologies into modular building blocks separated by function. Matthaei and Maurer[3] provide a comprehensive functional architecture based on the top-down approach. Notably, to model to the driving task in relation to human driving behaviour, the architectural design is divided into 3 abstraction levels[4, 5, 6]: navigation (strategical), guidance (tactical), and stabilization (operational). The three levels also correspond to different system scale, in term of resolution, horizon and accuracy. The architecture boils down to the utilization of surrounding and self-information from sensors coupled with external data to achieve the desired driving task. These tasks are defined granularly based on function and scale. The schematic of the top-down architecture can be seen in Figure 1-1.

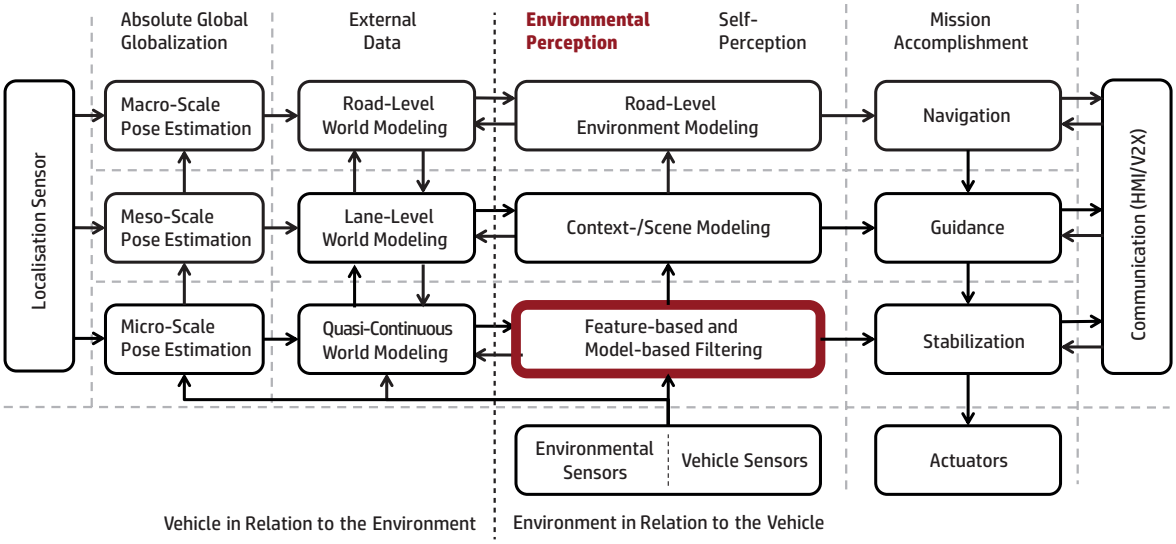


Figure 1-1: Functional system architecture for an autonomous vehicle (adapted from[3]). Low-level perception module is marked in red.

In the architectural point of view, the lowest level of the perception module (see Figure 1-2) is directly fed with environmental sensor measurement and is responsible for conveying environmental states to the upper-level perception module and orthogonally to the stabilization module. Therefore, it is clear that perception in the lowest level is crucial in the sense that its output becomes the reference of two higher perception modules as well as low-level vehicle stabilization. Referring to the convention used in functional system architecture, the following section thus will concentrate the discussion in the scope of low-level vehicle perception; in

particular, the object tracking of dynamic road participant.

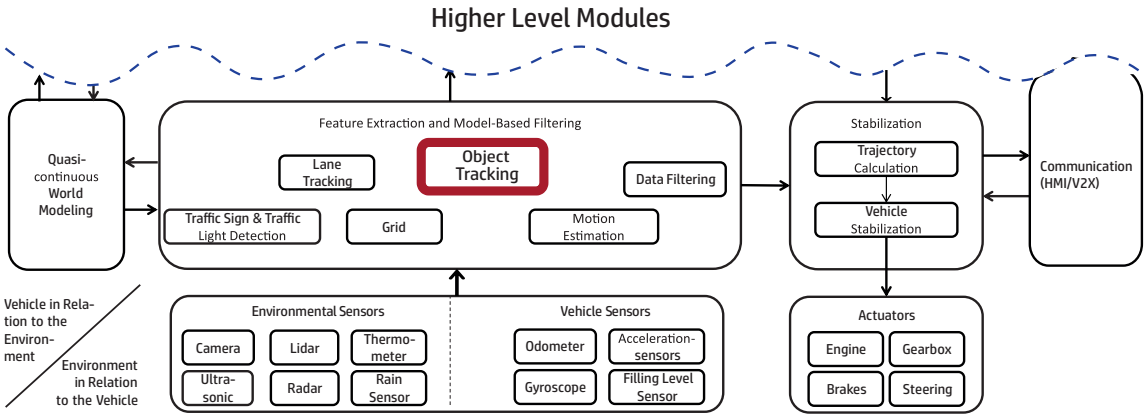


Figure 1-2: A low-Level overview of the functional autonomous Driving architecture. (adapted from[3]) The focus of this thesis in the overall architecture is marked in red.

1-2 Urban Vehicle Perception: Related Works and Challenges

The role of perception ability in autonomous driving technology has been briefly mentioned in the previous section. Here, we define it further and also discuss the state-of-art of urban autonomous driving along with its main challenges involved in employing vehicle perception.

The commercial implementation of lateral and longitudinal Advanced Driving Assists such as "Autopilot" in Tesla car[7] demonstrates that modern vehicle is able to utilise environmental understanding to gain partial automation in the highway-like situation. It is evident that the case of Tesla Autopilot may not be trivially applicable in an urban situation. In the inner-city driving scenario, a substansial variation of traffic and structures on the vehicle own and adjacent lanes essentially make the perception of relevant road users, static obstacles and the road course even more challenging task.

Urban-oriented autonomous vehicles themselves have been extensively researched in body of literature and even tested in (controlled) urban scenario, the latter part is particularly true for the vehicles which have competed in and cleared the DARPA Urban Challenges[8], such as Boss[9], Stanford's Junior[10] and MIT's Talos[11]. A public demonstration of Project Stadtpilot[12] and Project BRAiVE[13] further reinforces the feasibility of autonomous cars being ubiquitous in the future road.

There are important challenges which need to be addressed in urban autonomous driving: first. the significant variation of traffic and structures on the vehicle own and adjacent lanes essentially makes the separation between of relevant road object from the terrain even more challenging task[11]. Second, advancing from typical highway scenario into urban domain also require additional focus on the detection of a different class of traffic object, and especially the addition of Vulnerable Road User (VRU)[14]. Third, the need to classify the traffic object and predict its future trajectory (i.e. intent) also arises. Traffic objects also sport differing movement pattern which may change within a short timeframe. Since the vehicle is expected

to recognize the intention of surrounding objects, the task of vehicle perception is expected to become substantially more complex.

Vehicle perception is modelled in term of position and motion of dynamic objects relative to the vehicle position; it is the process of extraction of geometrical values (e.g. dimension, position, velocity, colour, etc.) from surrounding situation[3] to model their semantic context up to a sufficient extent[15]. While a human driver can assign a semantic meaning to the visual perceptions quickly and nearly without error, this is still a comparatively challenging task for a machine with the current state of the technology[16]. For instance, a machine can give out precise measurement and location of arbitrary traffic sign from a camera image. However, it cannot readily tell what that sign reads and what to do with it without a priori knowledge. Thus modelling the semantic context (i.e. object classification) is one of the primary tasks in machine perception. Recent works such as semantic labelling using stixel[17] demonstrate that such contextual information can be integrated during detection and tracking process to derive object class from geometrical and dynamics behaviour clue, for instance separating static object to dynamic object by looking at object shape and past motions.

Rieken et al.[15] consider there are two possible approach of vehicle perception: using a priori knowledge (i.e. localization) and perception-driven approach that relies onboard sensors to model all perceptive aspects of autonomous driving, furthermore the author also asserted that perception driven approach is more suitable for urban autonomous driving due to the risk associated with the changing nature of urban structures (i.e. a priori information cannot accommodate real-time change). Perception-driven approach generally relies on vision-based environmental sensors to obtain measurement data. The recently introduced compact 3D rotating LIDAR scanner[18] is especially suitable for this purpose, since it enables far-reaching high fidelity 3D acquisition of surrounding spatial information which is difficult to achieve with conventional sensing technologies, such as camera and RADAR.

It is also beneficial to describe the challenges in perception task in a quantitative manner. In essence, the difficulties faced in perception problem can be attributed to the incomplete quantities from the sensor to derive precise measurement and/or the lack of knowledge to assign a semantic context to the measured object, therefore using term *uncertainty* is appropriate. According to Dietmayer[16], the following three uncertainty domains exist for the machine perception:

1. **State uncertainty**: uncertainty in the measured physical variables, such as size, position and speed. This directly related to measurement errors in the sensors and signal processing module that cannot always be avoided
2. **Existence uncertainty**: the uncertainty as to whether an object detected by the sensors and transferred to the representation of the surroundings actually correspond to real object of interest.
3. **Class uncertainty**: uncertainty with regard to the correct semantic assignment, which can be caused by insufficiently accurate measured data or shortcoming in the classification procedure

These uncertainties need to be addressed for a reliable vehicle perception. For this purpose, the Bayesian probabilistic[19] filtering is widely used due to its heuristic-free approach noted as an advantage[16, 19], making it applicable generically. On the other hand, the computational

constraint has to be taken into account during application of optimal filtering in automotive purpose[20], and in state-of-practice, sensor-specific heuristic criterion is still effectively put to use to handle these three uncertainties.

Last but not least, in perception task not only the vehicle has to detect and derive a physical measurement of surrounding objects, but it also has to keep track of dynamic objects in a continuous manner for the reference point of vehicle closed-loop control. Going by this notion, the concept of object tracking becomes a core essence in urban vehicle perception.

1-3 Multi-Object-Tracking: Overview and State-of-Art

It is useful to understand the high abstraction of object tracking, especially the intended objective, how the uncertainties are being handled, followed by the state-of-art procedure. Additionally, since it is given that in urban scenario there are almost always multiple objects of interest surrounding the ego-vehicle, we generalize the tracking scheme in the urban situation as Multi Object Tracking (MOT). The schematic of MOT can be seen in Figure 1-3.

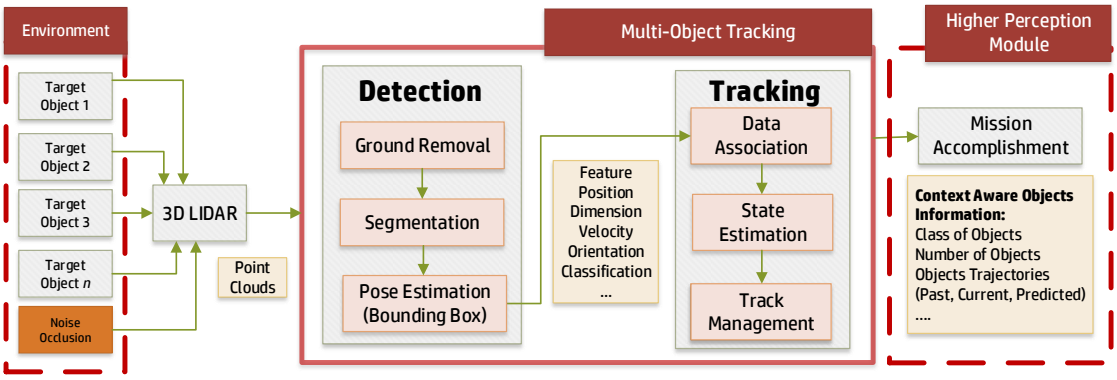


Figure 1-3: Schematic of object tracking

To model the perceived environment, a vehicle needs to be *continuously* aware of the kinematic states (e.g. position and velocity) of surrounding objects. The knowledge of the objects states is required to execute appropriate control action within reasonable time-frame, this is crucial as the vehicles control strategy relies heavily on the known environmental states to determine time-constrained control actions, for instance the ability to track the movement of pedestrians which is about to cross a road is used by the vehicle to apply braking in timely fashion, this reasoning is made possible by object tracking done in a causal manner (i.e. online) with some degree of guarantee of the execution time (i.e. real-time).

The object tracking procedure given in Figure 1-3 follows widely used[21, 22, 23, 24, 25, 26, 27] tracking-by-detection (i.e. object detection and tracking are done in successive fashion). On the detection stage, segmentation of the raw data is done to build basic feature of the scanned objects and to distinguish between dynamic and static objects. Subsequently, the segmented objects pose is then estimated based on either its outlier feature or fitting the measurement into known model[28]. Segmentation also introduces reduction of dimensionality, since tracking

each individual sensor's scan hits is often computationally infeasible depending the type of the sensor, and redundant as the sensor does not distinguish between object, terrain and noise.

At this stage, the raw measurement has already been translated into refined measurement with meaningful attributes (e.g. the static/dynamic classification and the objects pose). In another word, the object is successfully detected. Note that the detected object may not be static, in that case the attributes (i.e. the dynamic model states) will evolve over time.

Subsequently, the detected object is given to state estimation filter so that object kinematic states can be predicted according to a dynamic motion model. The purpose of the tracker is to effectively estimate the possible evolution of the detected object states even in the presence of measurement uncertainties, an optimal Bayesian filter such as Kalman Filter and Particle Filter are extensively used to address state uncertainties due to unmodelled dynamics of target evolution or noise acting on measurement.

Next, a data association (DA) procedure is done by assigning detected object into a track with established filter state or newly started trajectory. In multi-target object tracking scenario, there exist multiple objects with multiple different trajectories. The purpose of DA is to ensure the detected objects are localized while simultaneously maintaining their unique identity. At this stage uncertainties of the measurement due to finite sensor resolution (e.g. sensor may not be able to resolve multiple cluttered objects and return a single measurement instead) and/or detection step imperfection may arise In order to address this issue, Bayesian probabilistic approaches are commonly used to handle DA process, such as are Joint Probabilistic Data Association Filter (JPDAF)[29] or Multiple Hypothesis Tracking (MHT)[30].

Finally, in practice a track management is required to cancel out spurious track based on specific criteria. Track management is responsible for handling existence and classification uncertainties based on a sensor-specific heuristic threshold. For example, track existence is defined to be true if 90% of its trajectory of track hypothesis is associated with a highly correlated measurement. The same also applies with class uncertainties; object class can be determined based thresholding of dimension, past motion, or visual cue (i.e. colour). Alternatively, these two uncertainties can also be assumed to be a random variable evolving according to Markovian process. The Integrated Probabilistic Data Association[31] specifically model the existence of track as binary Markov process.

Note that in tracking-by-detection paradigm, the detection result is a pre-requisite of object tracking process and thus it must be seen as integrated part of object tracking process. on the other hand, the imperfection of detections steps will negatively impact the final tracking result. This imperfection can be caused by general uncertainty from raw sensor data and/or occlusion which can be caused either by vision blockage of other objects (e.g. a road sign blocking pedestrian) or orientation of object that simply does not allow LIDAR sensor to obtain complete geometric measurement

In a typical urban scenario, sensor occlusions are unavoidable occurrences, to address this several works try to explicitly model occlusion[32, 33, 34] during detection and incorporated occlusion handling in their object tracking method. In the inter-process level, one notable approach as proposed by Himmelsbach[25] is the bottom-up/top-down approach which lets the tracker and detector exchange stored geometrical information to refine the detection parameter. Alternatively, some other works have proposed a track-before-detect method that avoids explicit detection that takes sensor raw data as an input for tracking process[35].

1-4 Objectives

In the preceding section, the general requirement and uncertainties of vehicle perception in the urban situation have been established. Furthermore, as a state-of-art sensor, 3D LIDAR is widely discussed in research of perception tasks geared toward the autonomous vehicle. Among others, Chen, et. al[28] introduce model-based detection for surrounding vehicles, Schreier et. al[36] suggest a compact grid-based representation to reliably track uncertain objects, and Himmelsbach[25] proposed a top-down bottom-up approach to enhance detection and tracking result with simultaneous classification. **Notwithstanding, there are comparably less literature which addresses the integration of each stand-alone approach for vehicle perception MOT Task**.

Integration-wise, works of Zhang[37], Wojke[27], and Choi[38] notably propose the complete scheme of MOT using 3D LIDAR. **However, the use-case of urban driving is not a focus, and limitation of vehicle embedded computer is not necessarily taken into account.**

Finally, **a vast amount of MOT evaluations are concerned only with camera-based tracking[39], and there are comparatively fewer works that explicitly evaluate the 3D LIDAR-based MOT, especially so for MOT in a wide variation of urban situation.**

In this thesis, the above gaps are treated as research questions. Accordingly, this thesis seeks to address the questions by achieving the following global objective:

Research Objectives

Design, implement and evaluate an integrated framework of 3D LIDAR-based multi-target object detection and tracking tailored for autonomous urban driving; the framework takes a raw, 3D LIDAR data as raw input and provide objects kinematic pose as end output in robust, causal, and real-time manner.

Followed with more granular sub-objectives in which their fulfilment reflects the research contributions:

1. Identify the requirements of object tracking and detection in term of robustness against tracking target uncertainties and 3D LIDAR sensor limitation in urban situation.
2. Design algorithms which fulfil the defined requirements to achieve the precise and accurate task of both detection and tracking by a combination of probabilistic adaptive filtering and logic-based rule.
3. Perform real-time implementation and tuning of object detector and tracker in C++ with an interface to real-world, publicly available LIDAR datasets.
4. Design and perform evaluation based on established metrics against publicly accessible real-world LIDAR data for validation purpose
5. Perform comparison with state-of-art works which also use comparable MOT metrics.

At the last chapter (Section 6-2), the report shall recapitulate how the proposed approach fulfils each and every sub-objective.

1-5 Document Structures

The thesis report will be structured as follows: in this chapter, a brief overview of perception as part of autonomous driving functional architecture has been given, along with the state-of-art on-road perception technologies. Next, we have put our focus on the urban autonomous driving current research and its challenges. Finally, the general framework of MOT task has also been explained to bridge the pursuing chapter.

The thesis shall cover underlying theories and assumption to be used in the design and implementation of MOT. Accordingly, in **Chapter 2** the report introduces the reader to the rationale of choosing LIDAR sensor, along with the explanation of its data acquisition procedure. Fundamentals of LIDAR object detection along with the state-of-the-art works are to be presented. The applicability of these approaches for urban scenario object tracking will be explored. Subsequently, **Chapter 3** shall cover the formulation of object tracking as state estimation (filtering) problem. Several classes of tracking filter also will be introduced along with the use case geared toward addressing challenges in MOT problems.

The last 3 chapters deal with the realization of the MOT framework: **Chapter 4** will discuss the design and implementation of MOT as a hierarchical system: detector, tracker and all of its accompanying subsystem. Practical assumptions and approaches will be discussed to bridge the theoretical proposal discussed in preceding chapter to real implementation. Evaluation in **Chapter 5** will provide the results of quantitative and qualitative performance evaluation of the tracker, formulate comparison to state-of-art works and suggest possible improvement. Finally, **Chapter 6** will summarize the system design and results of the evaluation, identify the limitation and draw up conclusion based on discussion brought in the preceding chapters.

Chapter 2

Detection Fundamentals

2-1 Overview

This chapter aims to give an overview of object detection process. The detection problem can be defined as a process of recognizing the presence of target object with unambiguous identity and state information. This thesis utilises the tracking-by-detection scheme. Therefore the tracking process relies on detection results[32, 39] in order to be able to maintain the knowledge of the current and evolution of the objects state. Object detection using 3D LIDAR can be sequentially divided into two major steps: pre-processing of raw sensor data (segmentation), and estimating the pose of the target object. Thus, in this thesis, the object detection is seen as an integral part of the MOT system. The generalized flows of object detection found in the literature[37, 40, 28, 41, 42, 43, 44] can be seen in Figure 2-1.

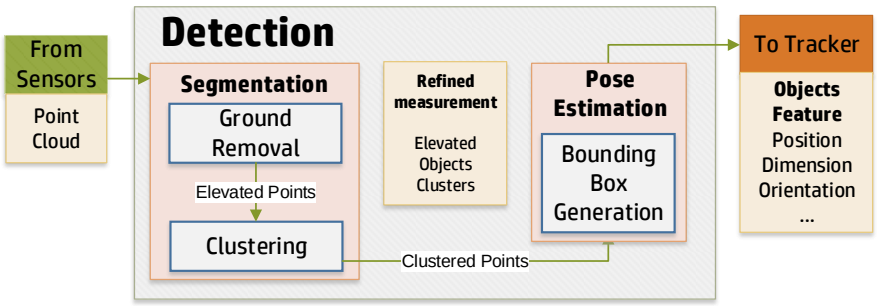


Figure 2-1: Schematics of object detection

The main challenge in object detection includes the uncertainty of sensor measurements and inevitable occlusion of scanned objects. This chapter will be structured as follows: first, the rationale of sensor choice is to be elaborated along with its technical detail and limitation, as it is useful to understand the characteristic of its raw measurement and the entailing obstacle. Second, we briefly discuss the advantages and limitation of state-of-the-art segmentation and pose estimation methods. Finally, we identify the most appropriate methods for urban-oriented MOT, which then will serve as the basis for the design and implementation chapter.

2-2 Spatial Data Acquisition using 3D LIDAR

2-2-1 Rationale of 3D LIDAR as Modal Sensor

Acquisition of surrounding objects states is enabled by various type of vision sensor, such as the conventional camera, stereo-camera and more recently, the LIDAR (Light Detection And Ranging). Compared to LIDAR, camera offers a relatively cheaper, widely available sensors which provide high-resolution and field-of-view (particularly the surround camera). Moreover, object tracking using camera as a modal sensor is a well-researched topic in computer vision field, Luo et. al[45] provides an extensive review of available MOT literature. Notwithstanding, object tracking for autonomous vehicle introduces an additional challenge: unstructured on-road environment featuring variations in illumination, background, and scene complexity[46]. It is also important to note that majority, if not all camera-based vehicle object tracking does not utilise surround camera, and deals only with frontal or rear objects (see[47, 48, 49, 50, 51]).

Likewise, in order to be capable of accurate tracking in full 3D space, the stereo-camera is needed. Note that we seek to implement the tracking in real-time and vehicle is often embedded with an embedded computer which prioritises robustness and cost-effectiveness over speed. The use of surround camera implies the processing of at least 4 combined images, on the other hand if stereo-camera setup is considered, disparity maps need to be generated from each camera, and thus, at minimum effectively doubles the image processing demand. Additionally, the camera has to provide high-resolution to be able to track objects located far away and cluttered with other objects. Moreover, to the best of author knowledge, no of-the-shelves solution offers stereo-camera in a surround setup, yet. It is also interesting to note that large number of literature focuses on the tracking of frontal, rear view or side view, not both. These combined prerequisites may not be particularly suitable for the goal of urban object tracking,

On the other hand, the recently introduced 3D LIDAR offers the ability to acquire massive-scale 3D geometrical information of the surrounding scene using single mounted sensor system. In addition to real-time friendly acquisition speed (10Hz), 3D LIDAR also provides additional information such as reflectance and compactness of the surfaces. Compared to conventional camera systems, 3D LIDAR provides lower resolution but is highly robust against unstable illumination and generally provides a larger horizontal field of view[52, 53]. More importantly, the dense point information that covers long distances makes 3D LIDAR especially suitable for the perception of arbitrary dynamic object[25].

However, 3D LIDAR is also known to have some limitations in regard to adverse weather condition, and limited angular resolution[53, 54]. Colour information and fine texture also cannot be derived from LIDAR data. Note that these attributes can be used to classify objects more accurately compared to the singular use of dimension and geometric shape, especially for incomplete measurement due to occlusion.

At this point, a distinct conclusion can be drawn based on the initial problem definition: object tracking for an autonomous vehicle in the urban situation. To achieve this, we mainly concern ourselves with accurate, long-range and surround 3D spatial data. Camera technology, as have been discussed above is capable of producing such result, but requires relatively more complex setup, compared to off-the-shelves 3D LIDAR scanner. Computation power also limits our use of camera for this purpose, as image processing for high-resolution stereoscopic data simply

results in unavoidable high demand of computing power, which may violate the real-time constraint.

3D LIDAR is not without shortcoming, particularly regarding low angular resolution and reconstruction of occluded data is comparatively harder due to no other attributes other than shape and reflectance can be derived from raw measurement. However, 3D LIDAR demands lower computational power relative to camera with comparable results. Generally, the choice of LIDAR as modal sensor is derived from the need to find the most suitable technology for its intended purpose: surround 3D spatial scanning

Camera on the other hand, would be more beneficial in the other tasks of model-based filtering, particularly lane tracking and traffic sign detection (refer to the Functional Architecture in Figure 1-2). Finally, one increasingly used approach to address limitation brought by single sensor modality is simply to use sensor fusion. Sensor fusion of Camera and LIDAR can be found in quite a number of recent literature[54, 55, 56, 57, 58, 59] with beneficial results.

However still, we need to take into account the capability of underlying computing platform to incorporate sensor fusion while still obeying the real-time constraint. The summary of comparison between two sensors can be seen in Table 2-1.

Table 2-1: Comparison of vision sensor technologies

Modal Sensors	Principle of Operation	Computational Complexity	Characteristics
Camera	Measure ambient visible light intensity	Moderate	High resolution, high vertical FOV, surround view needs multiple camera in a different location, susceptible to illumination change.
Stereo-camera	Uses multiple cameras to generate binocular vision	High	Same with camera but with added depth information, practical depth accuracy up to 50 m.
3D LIDAR	Measure distance & reflectance from 600-100 nm laser signal	Low (depends on number of points)	Low resolution, very long range (up to 200 m), not susceptible to illumination change, surround view using single mounted sensor system.

2-2-2 3D LIDAR Scanner: Velodyne HDL-64

This section shall briefly introduce the reader to the modal sensor used throughout this thesis, with its advantage and limitation in the context of urban vehicle perception.

Fundamentally, LIDAR is an optical measurement principle to localize and measure the distance of objects in space. LIDAR works under the same principle of the well-known RADAR technology, however instead of using radio wave, LIDAR uses ultraviolet, infrared or beams in the visible electromagnetic spectrum.[60]. LIDAR is used to measure the distance of between a host measurement system and the measured object. Automotive LIDAR typically uses *time-of-flight distance measurement*[60]. Principally, the time elapsed between transmitting and

receiving of the light (laser) pulse is directly proportional between these two objects, such that:

$$d = \frac{c_0 \cdot t}{2} \tag{2-1}$$

where d is the distance in m, c_o is the speed of light and t is time of travel. In addition, LIDAR is also capable of returning light intensity to represent the reflectance and contour of the object of interest. LIDAR works by powering a diode that emits laser pulse collimated by a transmitter lens. The emitted laser beam hits a target, reflected and a part of the reflected light hits a photodiode after passing through a receiver lens. A Digital Signal Processor with a precise clock is used to measure the time between transmitted and received signal which in turn is used to compute the target distance from the device. The intensity of the received signal is also used to measure target characteristics such as reflectivity. These sequence of operations is summarized in Figure 2-2(a). Naturally, a unidirectional sensor can only provide limited field of view, to compensate, modern 3D LIDAR scanner uses rotation mechanisms (see Figure 2-2 (b)) and includes more than one pair of emitter-receiver.

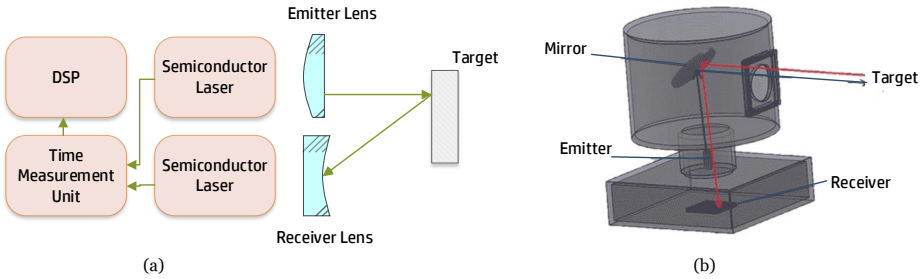


Figure 2-2: LIDAR Schematics (a) General Principle of Measurement (b) Rotating 3D LASER scanner

3D LIDAR is fairly recently introduced sensor, and compared to conventional camera, there is not yet equivalent cost-effective mass-produced solution tailored for automotive deployment. Velodyne series[61] is effectively the only available product in the market that can be readily integrated into a lab-grade land autonomous vehicle. Accordingly, most works (among others: [38, 62, 55, 27, 54, 63, 10, 64]) in the autonomous driving field that uses 3D LIDAR, also use Velodyne HDL-64 (see Figure 2-3) as a reference sensor. Consequently, the assessment of this technology will be based mostly on Velodyne HDL-64 implementation.

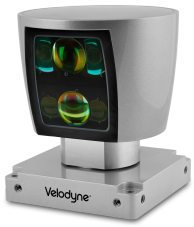


Figure 2-3: Velodyne HDL 64-E (source: Velodyne[61])

Velodyne HDL 64E scanner is a 3D LIDAR scanner consisted of 64 arrays of laser scanner with rotating head (i.e. as opposed to stand-alone LIDAR sensors commonly used solely for distance measurement). Unlike conventional 2D LIDAR scanner such as Riegl VQ-250 and Optech Lyn

which only provide single layer of scan, the HDL-64 the is capable of providing 64 layers of LIDAR measurement which conjointly construct a 3D spatial representation of surrounding environment. Each emitter-detector pairs single rotation scan generates the "layer" of points. The summation of these 2-D point creates the 3-D image. For instance, the collection of points from a single emitter/detector pair over flat ground appears as a continuous circle (see Figure 2-4 (c))

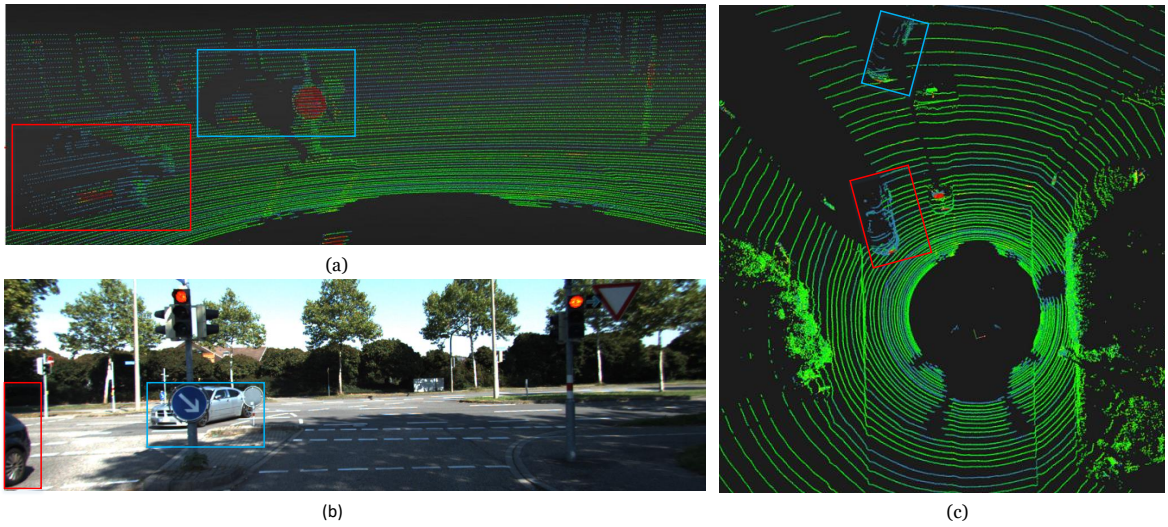


Figure 2-4: 3D LIDAR measurement and instance of occlusion (a) LIDAR frontal view (b) camera frontal view (c) LIDAR Top-view. Car 1 LIDAR reading (cyan box) is occluded by street sign and Car 2 (red box) is self-occluded due to its orientation against ego car (i.e. the rear part geometry cannot be completely measured)

Velodyne rotates at 10 Hz by default, a sensor update frequency which is well within real-time constraint for automotive use[65]. 3D LIDAR measurement has comparatively lower resolution than camera and being a collection of points and reflectance information, it leaves plenty of CPU time for higher level logic (e.g. object classification and tracking). Compared with camera however, 3D LIDAR lacks colour and texture information and thus will alter the approach used for classical machine vision classification.

Note that large number of points transfer per seconds may potentially bottleneck the bandwidth of embedded systems. In reality, reduction in dimensionality such as the use of occupancy grid and clustering[66] is still necessary. It is important to note, that the 3D Laser scanner has the limited angular resolution[27] (0.05 deg[61]) and low vertical field of view (27 deg[61]). This makes only a few usable reading in very long distances[27], and emphasizes the occlusion on a certain blind-spot angle.

Another important limitation that should be taken into account is that in case of occlusion, no data can be derived from the target object (refer to Figure 2-4 (a)-(c)). The areas of blanking, where a shadowing occurs are the blind spots of the sensor. Here, the occlusion phenomena highlight the limitation of LIDAR measurement which directly impacts the detection and tracking performance.

Regardless, we see that although not without several shortcomings, 3D LIDAR is shown to be more aligned with the use case of urban vehicle object tracking. Namely, due to mechanical

compactness of the sensors, directly available spatial data and immunity against common environmental attenuation in the urban situation.

2-2-3 Data Structure: 3D Point Cloud

Velodyne HDL 64E provides 360-degree spatial information which results in up to 1 millions 3D points per measurement. These points can be encapsulated using Point Cloud Representation[67] as the raw input of object detection process. Point Cloud data Representation is a convenient container to store, process and visualize 3D scanner raw measurement. Formally, it can be defined as a collection of multidimensional points representing the measured or generated counterpart of physical surfaces. Point Cloud is naturally characterized by spatial XYZ-coordinates and may optionally be assigned additional attributes (in case of LIDAR, the reflectivity information).

A point is simply a tupple incorporated with a number of various attribute:

$$p_i = \{x_i, x_y, x_z, I_i, ...\} \tag{2-2}$$

and a Point Cloud is represented as a collection of points:

$$P = \{p_1, p_2, p_3, p_i, ...p_i\} \tag{2-3}$$

The measurement of LIDAR scanner in point cloud representation can trivially be derived directly from the raw data and redistributed for multiple uses. Numerous object tracking literature body[62, 68, 69, 55, 52, 70, 20, 71, 33, 72] make use of publicly available LIDAR dataset. Not only this approach enable research to be done without sensor physical presence, but it also provides the opportunity of validating different algorithm with identical data sets.

KITTI datasets[73] is one notable example, it provides the recording of LIDAR sensors, among other sensors in a diverse urban driving scenario. The recording capture real-world traffic situations and range from highways over rural areas to inner-cityscenes with hand-labelled static and dynamic object. The sensor used by KITTI team is also Velodyne HDL-64E, and the authors also provide a synchronized camera raw image sequences in addition LIDAR recording. Furthermore, for validation purpose, object labels in the form of 3D tracklets can be used as ground truth. KITTI datasets point cloud data is used throughout this thesis as raw input, exemplary material source and evaluation dataset of the MOT framework.

2-3 Segmentation

A large amount of point clouds data demand a high computational power to process, in addition, due to discontinuous nature of point cloud, it is useful to combine the geometric points into a semantically meaningful group. Therefore, the raw measurement needs to be pre-processed to eliminate unnecessary element and reduce the dimensionality of possible target object before it passed to the detection process. The segmentation process mainly deals with differentiating non-trackable objects such as terrain and kerb from unique objects of interest such as cars, cyclists and pedestrians.

Segmentation method can be divided into two groups based on the underlying tracking scheme[28]: the grid-based and the object-based. The grid-based segmentation is mainly used for track-before-detect scheme, and the object-based is used mostly for the track-by detect scheme. Although it is important to note the relation is not exclusive. For example, Himmelsbach et al.[25] used grid-based approach in the pre-processing stage (specifically clustering) but later used object-based approach to perform tracking.

2-3-1 Grid-based

The grid cell-based methods chiefly rely on the global occupancy grid maps to indicate the probability of an object existing in a specific grid (i.e. occupied). One common approach to update the probabilities is to compare the occupancy in current time frame k with occupancy in time frame $k-1$ with the Bayesian Occupancy Filter[74]. In some implementation, Particle Filter is also used in used derive velocity in each grid[75, 76]. The particles with positions and velocities in a particular cell represent its velocity distribution and occupancy likelihood. When the neighbouring cells with similar speeds are clustered, then each cluster can be represented in the form of an oriented cuboid. Grid-based tracking results in simpler detection process and less complicated data association process, however, the disadvantage of grid-based representations is if a moving object cannot be detected (e.g. due to occlusion), the area will be mapped as a static grid by default if no occlusion handling is applied. Additionally, grid-based representations contain a lot of irrelevant details such as unreachable free space areas and only have limited ability to represent dynamic objects[36]. The latter part suggests grid-based detection is insufficient for the purpose of urban autonomous vehicle perception which require detailed representation of dynamic objects to model the complex environment. Therefore, we shall focus on the track-by-detect scheme, and the object-based detection will be explored more in-depth.

2-3-2 Object-based

Object based segmentation on the other hand, uses the point model (or rather collections of bounded points) to describe objects. Unlike grid-based method, a separate pose estimation and tracking filter are required to derive dynamic attributes of the segmented object. The segmentation is chiefly done with ground extraction to separate non-trackable object with objects of interest, followed by clustering to reduce the dimensionality of tracked objects. The two steps will be discussed in the following subsections. Note that the step-wise results of object-based segmentation processes can be seen in Figure. 2-5.

Ground Extraction

Due to non-planar nature of roads, a point cloud coming from 3D Laser scanner also includes terrain information which is considered as non-obstacle (i.e. navigable) by the vehicle. It is useful to semantically label the navigable terrain (hereby called ground) from the elevated point that might pose as an obstacle. Ground extraction is an important preliminary process in object detection process. As we are going to deal with a large number of raw LIDAR measurement, computation load must be factored during implementation. Chen et al.[77]

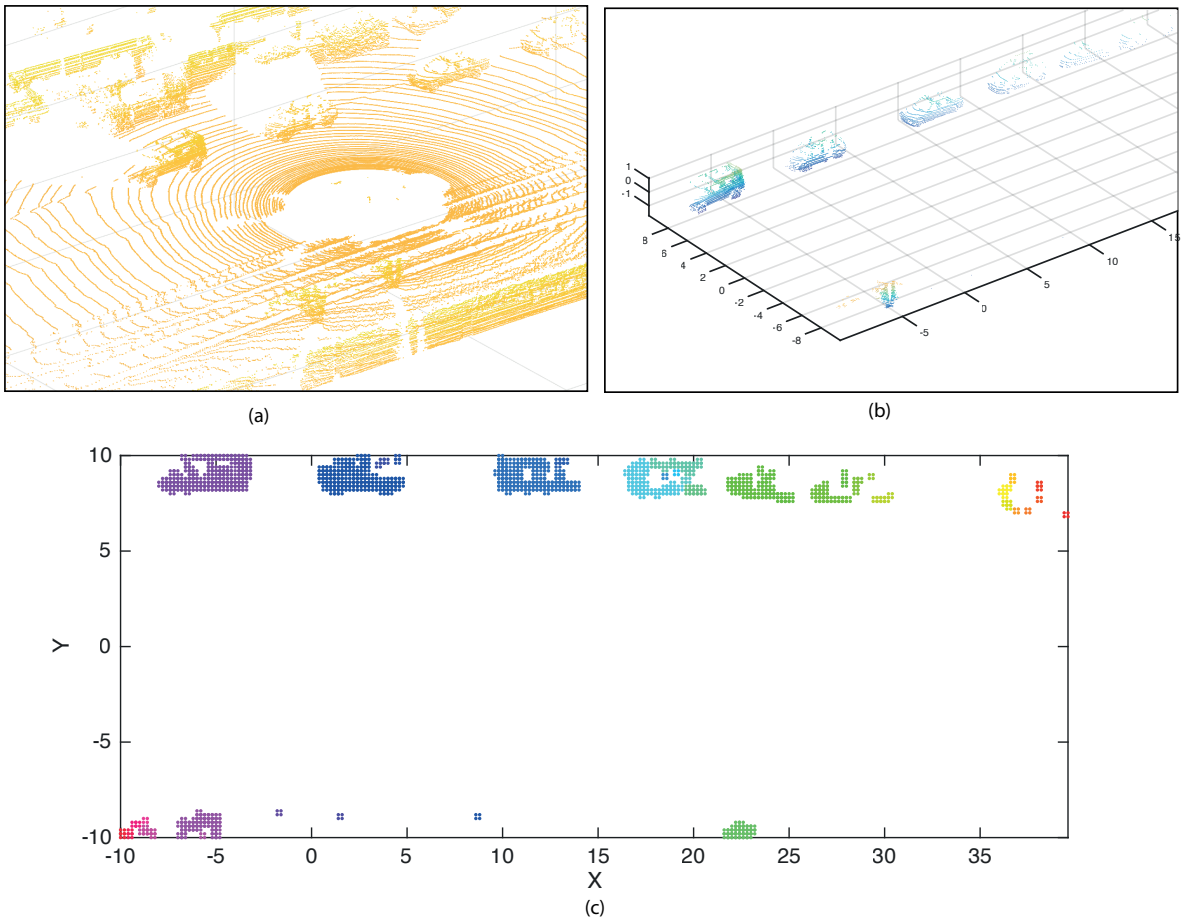


Figure 2-5: Segmentation process results (a) raw measurement (b) post-ground extraction (c) post 2D Clustering (different color per cluster)

divide ground extraction process into three subgroups: grid/cell-based methods, line-based methods and surface-based methods. Meanwhile, Rieken et al.[15] consider line-based and surface-based method as one big group called scan-based method. Grid-cell based method divides the LIDAR data into polar coordinate cells and channel. The method uses information of height and the radial distance between adjacent grid to deduct existence of obstacle when slope between cell cross certain threshold, i.e. slope-based method (see[12, 10]).

On the other hand, scan-based method extracts a planar ground (i.e. flat world assumption) derived from specific criteria, one of the approaches is to take the lowest z value and applying Random sample consensus (RANSAC) fitting to determine possible ground[78]. The advantage of grid cell-based method is that the ground contour is preserved and flat terrain is represented better. However, compared to the scan-based method it does not consider the value of neighbourhood channels, and thus may lead to inconsistency of elevation due to over-sensitivity to slope change. Ground measurement may also be incomplete due to occlusion by a large object. One notable approach which factored the occlusion is by Rieken et all.[14], they combine of channel-wise ground classification with a grid-based representation of the ground height to cope with false spatial measurements and also use inter-channel dependency to compensate for missing data.

Clustering

A large number of point clouds are computationally prohibitive if the detection and tracking are to be done over individual hits. Therefore, these massive point cloud is to be reduced into smaller clusters in which each of them is simply a combination of multiple, spatially close-range samples; the process is called *clustering*. Clustering can be either done in 3D, 2D (taking the top-view or XY plane) or 2.5D[79] which retain some z-axis information such as elevation in occupancy grids.

2D clustering offers computationally simpler operation. Rubio et al.[80] presented a 2D clustering based on Connected Component Clustering which has shown to be implementable in real-time due to its low complexity, this method is also used in 3D object tracking method by Rieken et al.[14].

Some literature in 2D object tracking[25, 45, 81] have shown that this approach is often sufficient in the application of object tracking. However, care should be taken as vertically stacked objects (e.g. pedestrian under a tree) will be merged into one single cluster, which might be undesirable depending on the vehicle navigation strategy.

3D clustering offers high fidelity object cluster that incorporates the vertical (z-axis) features. Still, the resulting data and the computational effort required is some magnitude larger than its 2D counterpart. Compared to 2D clustering, there are fewer works which explicitly deal with 3D clustering for object tracking with LIDAR data. Klasing et. al.[63] proposed a 3D clustering method based on Radially Bounded Nearest Neighbor (RNN), and more recently Hwang, et. al[82] using DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to employ full 3D clustering. Considering real-time requirement and significant number of points involved, 2 or 2.5D clustering is more preferred[83] owing to the fact vehicle onboard computer likely to have limited computational power.

2-4 Pose Estimation

Recognizing the existence of objects of interest has been done in the segmentation. Subsequently, in order to extract usable information in term of the object trajectory and orientation, the pose estimation needs to be done. Object *pose* is a broad term that may include the dimension, orientation (heading), velocity and acceleration of such objects. Pose estimation generally can be grouped into model-based or feature-based method. Model-based pose estimation aims to match raw measurement into a known geometric model, while feature-based pose estimation deduces object shape from a sets of feature.

2-4-1 Model-based

Model-based pose estimation uses optimization-based iteration to fit vehicle into cuboid or rectangle representation. The cuboid object is parametrized and the most probable vehicle pose from the segment points are iterated. In order to fit clusters of points into a model, edge-like features are to be extracted and "best-fit" method is utilised to fit it into a known model. A notable example is in Barrois[84] where the optimization problem is formulated as

the minimization of the polar distance between the scene points obtained and the visible sides to compute the best vehicle pose.

Petrovskaya and Thurn[21] use importance sampling scoring based on the fitting of measurement to a predetermined geometric model. Another interesting approach is by Morris et. al.[85] whose matched filter takes view-dependent self-occlusion effects into account, and utilise 4 rectangles to represent the inner and outer sides of the vehicle.

Another the major challenge in bounding box generation is the orientation estimation, common approaches are by calculating the minimum area of clustered points[86], however in the presence of partial occlusion the results can be spurious in term of dimension and orientation accuracy, to tackle this problem Rieken, et al.[43] uses an L-, U- or I-like simple set of geometric classifier to derive most appropriate orientation. An alternative approach is to use convex-hull method to generate bounding box[87, 52]; the idea is to minimize the average distance between the convex-hull points and fit a rectangle. (see Figure 2-6).

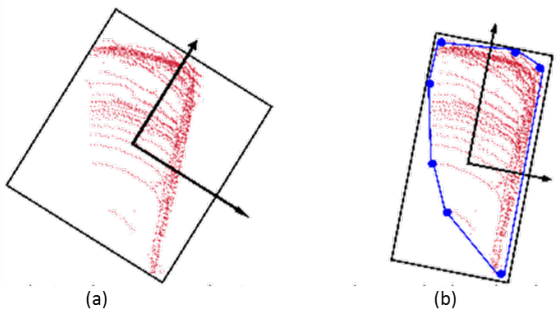


Figure 2-6: Example of model-based filtering: Bounding box fitting for a partially occluded object (a) with minimum area rectangle (b) with model-based convex-hull fitting. Adapted from[52]

Model-based pose estimation can also be combined with Bayesian probabilistic likelihood such as in the works of Vu and Aycard[88], Liu[64], and Nashashibi[89]. They explicitly modelled the possible occlusion area to estimate the vehicle dimension based on scan-line distance. Liu[64] in particular uses "transitional region" between the inner-outer bounding box model object aiming to accommodate more measurement errors.

Although model-based method offers optimal pose estimation, the major disadvantage of this method is the high computational time required, and this may not be suitable for real-time application. Moreover, the optimisation problem may reach a solution at local minima depending on the initialization and results in a sub-optimal pose. As a consequence, the feature-based pose estimation shall be preferred.

2-4-2 Feature-based

Feature-based pose estimation, on the other hand, deduces the object dimension based on the edge features. For instance, Darms et al.[90] extracted edge targets from raw measurement as a part of the box model to represent the vehicle's pose. Himmelsbach et. al.[25] used Random sample consensus (RANSAC) algorithm to fit the dominant line of the segment points with the orientations of the objects. Luo et al.[45] uses a graph-based method to fit clustered scans

into an arbitrary shape, although this approach does not provide orientation information as is.

Another approach, as done by Ye et. al.[91] is to extract a sequence of points facing the sensor with the smallest radius among the others in similar observation azimuth. then these points were fitted into L-shaped polyline using iteration endpoint algorithm. Mertz et al.[92] use corner-fitting method to iterate through set of edge points to deduce the possible corner points. Similarly, Tay et al.[93] use edge filtering to deduce a bounding box vertex by iterating the edge lines to the nearest end point. (Refer to Figure 2-7)

Machine-learning AdaBoost based detection methods can be found in the work of Zhang[94] et al., they use positive training samples obtained from multiple viewpoints of the object to train the detector to find 3D Harr-like features from clustered points. The trained detector is then used to generate a voxelized box for detection result. More recently, Braun, et. al.[95] utilises sensor-fusion approach using Regional Convolutional Neural Network (R-CNN) to estimate object orientation based on the joint proposal of stereo camera as well as LIDAR data

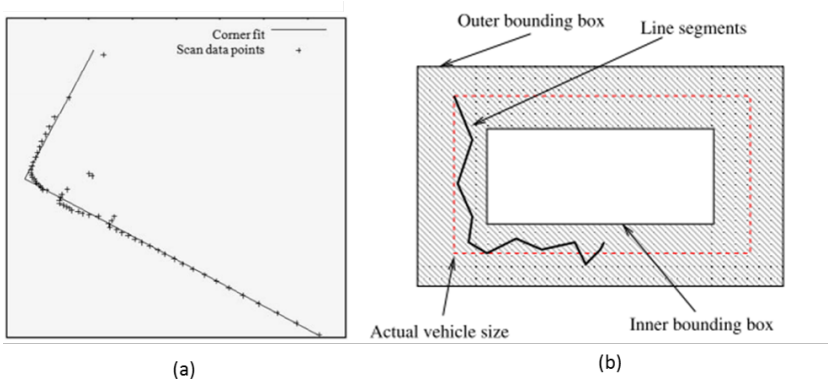


Figure 2-7: Example of feature-based detection (a) corner fitting (adapted from Mertz et al.[92]) (b) edge filtering (Adapted from[93])

Feature-based detection offers good trade-off between accurate pose estimation and computational-time. However, Liu[28] also asserted that these approaches are notably sensitive to unstable measurement.

To summarise this chapter, the 3D LIDAR sensor is selected due to its ability to acquire surround spatial information with feasible computational cost. However, occlusion-aware detection method has to be used to utilise the full potential of LIDAR. In addition, a real-time requirement calls for efficient methods. Therefore, based on these two criteria, the Slope-based channel classification and 2D Connected Component Clustering are to be used for segmentation process with embedded height. Meanwhile, the pose estimation shall utilise minimum area rectangle augmented with L-shape fitting and cluster height information to form a 3D Box. Both methods have been shown to yield fast but reasonably accurate detection result under urban environment[96, 15], which in turn is essential for mission-critical urban MOT task.

Chapter 3

Tracking Fundamentals

3-1 Overview

Object tracking can be defined as the process of using sensor measurements to determine the location, path and characteristics of arbitrary objects[97]. Typically, object tracking seeks to enumerate the location, (unique) identities, and various types of states such as velocity, orientation and in the context of autonomous driving, the classification of such object. In the framework of vehicle perception, the task of object tracking is essential, as the environmental measurement is only useful if the vehicle is capable of making use of the said measurement into actionable information.

The necessary component of object tracking has been introduced in Section 1-3 (Multi-Object-Tracking: Overview and State-of-Art). To further elucidate the object tracking task, it is useful to see the interprocess flow between component as depicted in Figure 3-1. The result of detection is used to start a new track, or if existing track exists, the measurement is checked if it statistically likely to be correct measurement through gating. Passing the gating is the prerequisite of association between measurement and tracks before being sent forward to state estimator. This chapter shall cover the assumed modelling of sensor and target dynamics, used not only in state estimatio and prediction, but also Data Association. Accordingly, several classes of Bayesian filter will be introduced.

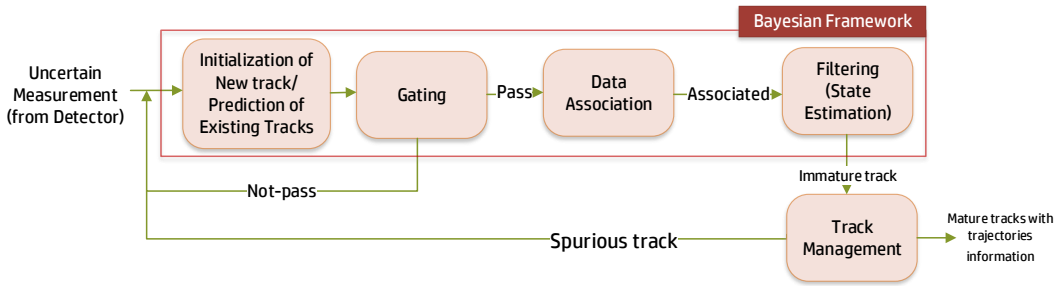


Figure 3-1: Object Tracking flow. Bayesian filtering is to be used to handle the uncertainties.

3-2 Sensor and Target Modelling

The LIDAR sensor is placed on moving ego-car, which is considered as the origin. Due to the measuring principle of rotating LIDAR sensor the measurement originally comes in polar coordinates (in the form of distance and bearing). However, Velodyne digital processing unit (and by extension the KITTI dataset) readily provides the sensor measurement on the Cartesian coordinate system, In this thesis, the latter coordinate system will be used to conform better with ego-vehicle navigation frame. The relation between the ego-car navigation frame and sensor measurement frame can be seen in Figure 3-2.

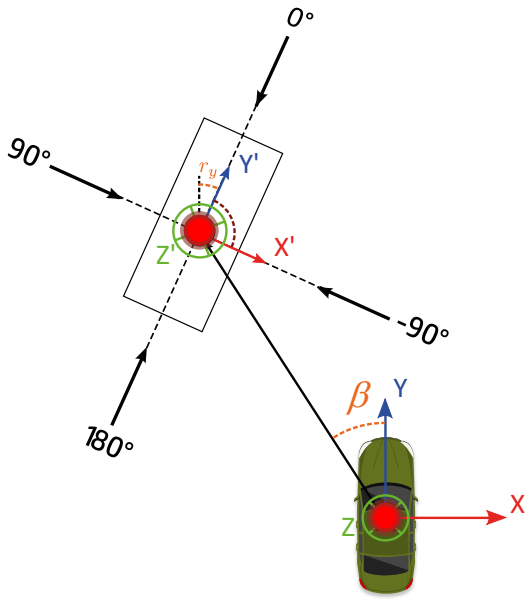


Figure 3-2: Sensor measurement frame relative to ego-car

As mentioned in Section 2-2, Velodyne HDL-64 raw measurement consists of point cloud in 3D Cartesian coordinates combined with intensity, or *reflectance*. Although intensity can be used to enhance the detection process (for example using threshold-based noise removal[98]) its value highly varies with the material, colour and geometric shape of the detected object[99]. In the urban scenario where the object appearance is highly non-homogeneous, the intensity information is thus not used due to the requirement of variable calibration and overly complex modelling.

At this point, the detector already bounded all likely clusters belonging to objects into boxes. Note that in this thesis a *point tracking* is used as opposed to extended object tracking[100] (i.e. tracking with geometrical states embedded). One chief reason is that LIDAR sensor reflects highly inconsistent size of measurement generating points depending on distance, reflectance and ego vehicle point of view angle, incorporating geometric shapes is guaranteed to undesirably enlarge measurement uncertainties. In addition, a minimum number of filtered state is preferred to limit the requirement of computational power. Since it is given multiple filter instances are needed to perform MOT, a scalable design choice is preferred. It is important to note that the box representation is still embedded in each track to represent each tracked

object spatial extent (will be discussed in details in the next chapter).

Furthermore, it is reasonable to assume that road objects does not have vertical motion (in Z-axis) and thus we are left with measurement in the position vector of z in x and y-axes:

$$\mathbf{z} = \begin{bmatrix} x_{pos} & z_{pos} \end{bmatrix}' \tag{3-1}$$

Note that the measurement is obtained at discrete time instance.

The tracking target dynamic motion model can be described mathematically by a discrete time, stochastic state-space model in the form of:

$$\begin{aligned} \mathbf{x}_{k+1} &= f_k(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= H(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_k \end{aligned} \tag{3-2}$$

with state vector $\mathbf{x}_k \in \mathbb{R}^p$, measurement vector $\mathbf{z}_k \in \mathbb{R}^n$ system function F, and measurement function H at each time step k. The system is disturbed with zero-mean, white, Gaussian noise sequences $\mathbf{w}_k \in \mathbb{R}^n$ and $\mathbf{v}_k \in \mathbb{R}^n$ which are mutually independent with covariance matrices $\mathbf{Q}_k$ (motion noise) and $\mathbf{R}_k$ (measurement noise).

In most traditional models, the state vector includes the position, the velocity, and the acceleration in both dimensions, here we also incorporate the velocity magnitude and heading angle since these states are usually a pre-requisite for autonomous vehicle trajectory planning[101], the state vectors $\mathbf{x}_k$ thus becomes:

$$\mathbf{x}_k = \begin{bmatrix} x_{pos}k \\ z_{pos}k \\ k \\ v_k \\ \dot{\psi}_k \end{bmatrix} \tag{3-3}$$

with $x_{pos}k$ is the position in the x-axis, $z_{pos}k$ is the position in the y-axis, ψ_k is the yaw (heading), v_k is the magnitude of velocity and $\dot{\psi}_k$ is the yaw rate, all computed at time instance k. The evolution of tracking target's system states over time is depicted in Figure 3-3.

The measured output at time step k (i.e. observation) $\mathbf{z}_k$ thus can be obtained by multiplying the corresponding (linear) measurement function h_k with the states vector:

$$\mathbf{z}_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{h_k} \mathbf{x}_k + \mathbf{v}_k \tag{3-4}$$

Finally, we need to consider that road objects are often manoeuvring objects: they do not follow a single motion model consistently over all time steps. For instance, the dynamics of a straight cruising vehicle would differ significantly when compared that of a turning vehicle. To address this issue, multiple motion models are used to capture the dynamics of the manouevring object as much as possible without the need of expanding the motion and measurement noise to address the uncertainties. The motion models system function and its general description can be found in Appendix D.

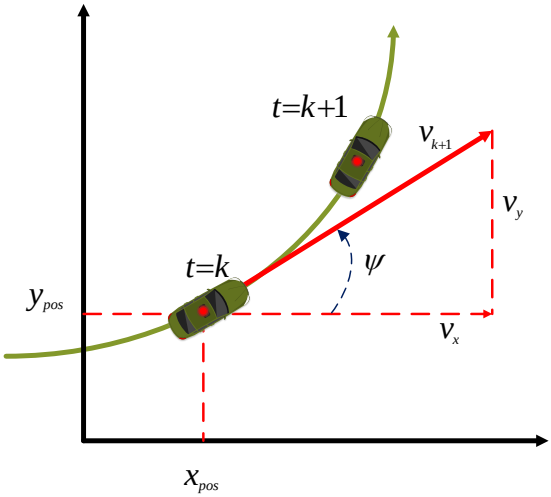


Figure 3-3: Evolution of Tracking Target

3-3 Object Tracking as A Filtering Problem

In this thesis, object tracking problem is modelled as a filtering problem in which the object states are to be estimated from noisy, corrupted or otherwise false measurements. The estimated states and assumed system dynamics are given in the previous section. The basic concept of Bayes filtering is introduced along with the filters which will be used during implementation.

3-3-1 Bayes Theorem in Object Tracking

In object tracking problem, quantities such as sensor measurements, and the states of the tracked objects along with its surrounding environment are modelled as random variables. Since random variables can take on multiple values, the probabilistic inference is needed to compute the law governing the evolution of said variables. Bayes filtering is one of the most used and well developed probabilistic and statistical theories which can be applied directly to model and solve issues in object tracking[19]. This approach is the fundamental of more complex algorithms used in solving object tracking problem.

Let $\mathbf{x}_k$ be the random variables corresponding to the tracker object, $\mathbf{Z}_k = (z_1, z_2, z_i..., z_k)$ be a measurement at time i related with $\mathbf{x}_k$. Furthermore, given an assumption that random variable take continuum of values, the knowledge of object x is represented by a probability density function (pdf) $p(\mathbf{x}_k)$. Bayesian theorem (in which the filter is based on) allows updating existing knowledge about $\mathbf{x}_k$ given knowledge of $\mathbf{Z}_k$, we can infer[97]

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \frac{p(\mathbf{Z}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{Z}_k)} \tag{3-5}$$

However, since the measurements are received in a sequence over discrete time step, a recursive form needs to be used. Assuming the system has Markov property (a process in the future is stochastically independent of its behaviour in the past).

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \frac{p(\mathbf{Z}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{Z}_k|\mathbf{z}_{k-1})} p(\mathbf{x}_k|\mathbf{x}_{k-1})p(S^{k-1}|\mathbf{z}_{k-1}) \tag{3-6}$$

Furthermore, it is also essential to know the sequence of objects and their states and also the number of objects and their states at a particular time k. This knowledge can be derived from $p(\mathbf{x}_k|\mathbf{Z}_k)$ by integrating out objects and their states as follows:

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \frac{1}{p(z_k|\mathbf{z}_{k-1})} \int_{Sk-1} p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})d\mathbf{x}_{k-1} \tag{3-7}$$

The integral in above equation is called Chapman-Kolmogorov equation. The solution of the equation gives the predicted state of the elements of $\mathbf{x}_k$, given all the measurements up to time $k-1$ and the state at time $k-1$, the $p(z_k|\mathbf{x}_k)$ term will update the predicted state on the update of measurement at time k before being re-normalized.

Solving the recursive relation in (3-7) is the main idea object tracking as filtering problems. Object tracking filter generally differs only in problem requirements and the use of different forms of the likelihood function $p(z_k|\mathbf{x}_k, \mathbf{z}_{k-1})$ and the transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. In the context of object tracking the prior density is derived by the object dynamic equation and the likelihood is derived from the measurement (sensor observation) equation. Therefore for the MOT problem, Bayesian estimation can be applied as follow[102]:

1. **State prediction:** the current state of tracked object the last time step is synchronized to the measurement time of the most recent sensor observations. A motion model is used to describes the expected evolution of the system over time, however note that synchronization based on a model typically adds uncertainty to the estimate, since no model is perfect (in fact, all models are wrong, but some are useful[103]). If the state of the object and the provided sensor measurement are not defined in the same coordinate system, the prediction step also includes a coordinate transformation process object state to the measurement domain. In order to achieve this transformation, a sensor-specific measurement model which account sensor noise is utilised.

2. **State update** State prediction results in a probabilistic representation of the synchronized system state (referred as *state hypothesis*) in the measurement domain. In addition, the corresponding sensor observation sample in the same time step is available for comparison. Next, the likelihood of the sensor model is evaluated at the position of the observed measurement, given the assumption of the state hypothesis. Based on the likelihood, the predicted state estimate is adjusted to represent the best knowledge of the system at the current time step. The final result of update step is a probabilistic combination of the information from the latest sensor measurement with the previously accumulated system state.

The following subsections shall go through relevant Bayesian filters relevant to MOT framework proposed in this thesis.

Remark: unless otherwise stated, the following subsections are to use state variables, system function, measurement equation and noise characteristics introduced in Subsection 3-2 (Sensor and Target Modelling).

3-3-2 Unscented Kalman Filter

Kalman Filter (KF)[104] is the analytical implementation of the Bayesian method that seeks to recursively compute the optimal parameter estimates (i.e. the object states) from its posterior density[19]. Specifically, KF assumes the object dynamic equation and posterior density at previous time step follow Gaussian distribution and the system and measurement function are linear. This assumption may not work well with object tracking problem as the tracked object dynamics cannot be perfectly captured by linear motion model. To realize tracking by means of a nonlinear model, the Extended Kalman filter (EKF)[105] as well as the Unscented Kalman filter (UKF)[106] are used. The EKF linearizes the system function or measurement function using a Taylor series approximation. Interested reader may refer to[97] for algorithmic and mathematical derivation of classical KF and EKF for object tracking,

The UKF on the other hand, avoids computationally expensive linearization altogether and opt to use an approximation based on so-called sigma points from a Gaussian distribution. Instead of linearizing around the mean and using the Jacobian as the system function, the UKF propagate the chosen sigma points through the original non-linear function. The Gaussian then can be recovered from the newly transformed points. Note that the resulting probability density function (pdf.) is only an approximation of Gaussian distribution, so it is not an optimal filter like classic KF per se. Nevertheless, UKF is shown to have more accurate estimate than EKF in the presence of strong non-linearity[106]. The comparison between classic KF, EKF and UKF can be observed in Figure 3-4.

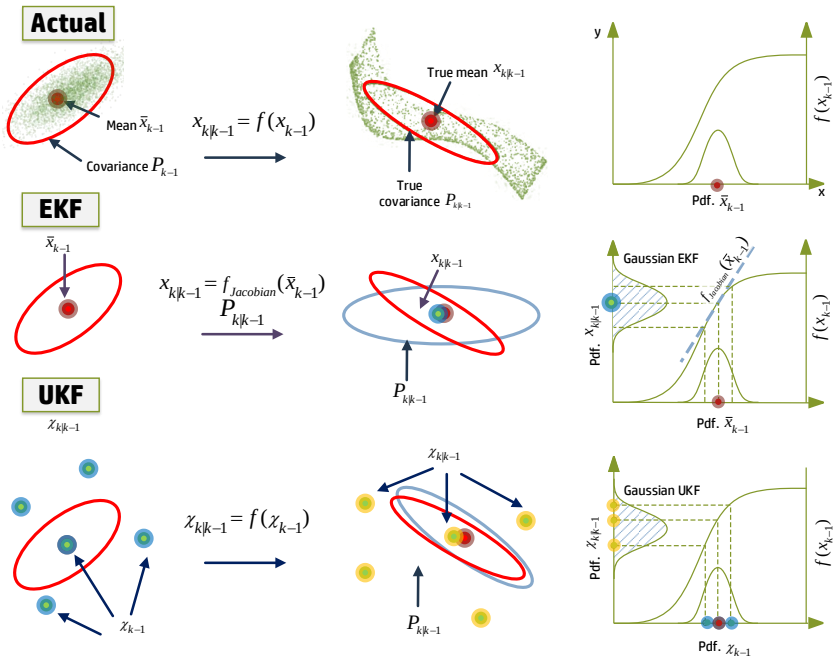


Figure 3-4: Comparison between UKF and EKF on the propagation of mean and variance in non-linear system. Note that the UKF probability density function (pdf) is only an approximation of Gaussian. Inspired by[19, 107]

.

UKF is in widely used in LIDAR Multi-object tracking as presented in[72, 69, 81, 25] due to its comparable computational complexity to standard KF and noted to be less demanding than EKF.[97]. The characteristic is the rationale of UKF use in this work.

The iteration steps of UKF is summarized as follows:

Sigma Point Sampling

Form set of sigma point χ:

$$\begin{aligned}
\chi^0_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} \\
\chi^i_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_i, \qquad i = 1,\dots,L \\
\chi^i_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_{i-L}, \qquad i = L+1,\dots,2L
\end{aligned} \tag{3-8}$$

where $\left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_i$ is the "i"th column of the matrix square root of $(L+\lambda)\mathbf{P}^*_{k-1|k-1}$.

Prediction

The sigma points are then propagated through the transition function f.

$$\chi^{*,i}_{k|k-1} = f(\chi^i_{k-1|k-1}) \quad i = 0,\dots,2L \text{ where } f\ R^L \rightarrow R^{|\mathbf{x}|}. \tag{3-9}$$

The weighted sigma points are recombined to produce the predicted state and covariance.

$$\begin{aligned}
\hat{\mathbf{x}}^-_{k|k-1} &= \sum_{i=0}^{2L} W^i_s \chi^{*,i}_{k|k-1} \\
\mathbf{P}^-_{k|k-1} &= \sum_{i=0}^{2L} W^i_c\ [\chi^{*,i}_{k|k-1} - \hat{\mathbf{x}}^-_{k|k-1}][\chi^{*,i}_{k|k-1} - \hat{\mathbf{x}}^-_{k|k-1}]^{\mathrm{T}} + \mathbf{Q}_{k-1|k-1}
\end{aligned} \tag{3-10}$$

where the weights for the state and covariance are given by

$$\begin{aligned}
W^0_s &= \frac{\lambda}{L+\lambda} \\
W^0_c &= \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) \\
W^i_s &= W^i_c = \frac{1}{2(L+\lambda)} \\
\lambda &= \alpha^2(L+\kappa) - L
\end{aligned} \tag{3-11}$$

where α and κ control the spread of the sigma points. β is related to the distribution of x.

Update

Again, form a set of $2L + 1$ sigma points

$$\begin{aligned}
\chi^0_{k|k-1} &= \mathbf{x}^-_{k|k-1} \\
\chi^i_{k|k-1} &= \mathbf{x}^-_{k|k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}^-_{k|k-1}}\right)_i, \qquad i = 1, \ldots, L \\
\chi^i_{k|k-1} &= \mathbf{x}^-_{k|k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}^-_{k|k-1}}\right)_{i-L}, \qquad i = L+1, \ldots, 2L
\end{aligned} \tag{3-12}$$

Propagate sigma points through the observation function h

$$Z^i_k = h(\chi^i_{k|k-1}) \quad i = 0, \ldots, 2L \tag{3-13}$$

Produce the predicted measurement and predicted measurement covariance.

$$\hat{\mathbf{z}}^-_k = \sum_{i=0}^{2L} W^i_s Z^i_k \tag{3-14}$$

$$\mathbf{x}_k = \sum_{i=0}^{2L} W^i_c \; [Z^i_k - \hat{\mathbf{z}}^-_k][Z^i_k - \hat{\mathbf{z}}^-_k]^{\mathrm{T}} + R_{k|k-1} \tag{3-15}$$

The state-measurement cross-covariance matrix becomes

$$\mathbf{C}_{\mathbf{x}_k\mathbf{z}_k} = \sum_{i=0}^{2L} W^i_c \; [\chi^i_{k|k-1} - \hat{\mathbf{x}}^-_{k|k-1}][Z^i_k - \hat{\mathbf{z}}^-_k]^{\mathrm{T}} \tag{3-16}$$

and the UKF gain is given by

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{z}_k}\mathbf{P}^{-1}_{\mathbf{z}_k\mathbf{z}_k} \tag{3-17}$$

The updated state and covariance then can be computed as

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k-1} + K_k(\mathbf{z}_k - \hat{\mathbf{z}}^-_{k|k-1}) \\
\mathbf{P}_k &= \mathbf{P}_k = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{x}_k\mathbf{K}^T_k
\end{aligned} \tag{3-18}$$

An alternative to KF is Particle Filter[108] (PF), also called Sequential Monte Carlo filter. PF is another implementation of the Bayesian recursive filter which aims to deal with a more general case where the Gaussian-linear assumption does not necessarily hold. The idea behind PF is to approximate the posterior of a set of weighted hypothesis states (called "particle"). PF actually uses the same principle with UKF as both use Monte Carlo sampling; the differences is UKF use deterministically few sampling points while PF uses a much larger sample points in an attempt to propagate an accurate (but computationally expensive) probability distribution of the state.

PF notably does not require linearization and thus algorithmically simpler to implement. However, the performance of filter is dependent on the number of samples and the required

number of generated particle is exponentially related to the model dimensionality. Thus PF is often limited low dimensional spaces[108].

A notable use of PF for LIDAR tracking can be found in the work of Petrovskaya and Thrun[21] for their autonomous driving car "Junior" where Rao-Blackwellized Particle Filter is used as an alternative to EKF. Special care was taken to limit the complexity of the PF formulation such as limiting the estimated parameters and storing only vehicle pose and velocity. Fortin, et. al[109] propose a method for joint detection and tracking of vehicles in scanning laser range data. Danescu et. al.[76] applied occupancy-grid to reduce the dimensionality of the particles and aim to achieve real-time performance, similar compact representation approach using stixel and PF can also be found at[110].

PF is an attractive solution or tracking problem, nevertheless its use is relatively less common than KF particularly in real-time MOT problems for efficiency reason. As discussed above extra steps need to be taken adapt PF for real-time object tracking. Ultimately, we have to account the sharing of the computational load with the processing of LIDAR raw data (i.e. detection) and simultaneous multi-target tracking.

3-3-3 Interacting Multiple Model

The state estimation problem relies on the object motion model to predict and update the object states. However, in an urban situation tracked objects do not necessarily move in well-defined pattern. Even if a perfect motion model representing the object trajectories is available, there is no guarantee that the object will follow a specific motion model all the time. For instance, this scenario can be commonly found in road junction with a traffic light. Assuming at least moderate traffic, road users may follow different motion model depending on its position and navigation intent (refer to Figure 3-5). This phenomenon means that manoeuvring targets filtering need to be used.

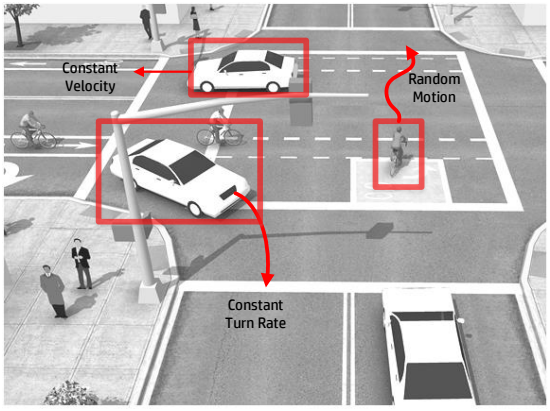


Figure 3-5: Manoeuvring targets in typical road junction. Here we consider 3 different motion models as defined in[111, 112], Constant Velocity, Constant Turn Rate and Random Motion. All models are listed in Appendix D.

The use of object-manoeuvring tracker also implicitly enables tracker to distinguish between statics and dynamic objects. Typically, dynamic objects move in predictable pattern while

noises can be seen to move in a more random pattern. Finally, the use of non-linear motion model requires the uses of non-linear KF as state estimator such as UKF. IMM implementation itself is commonly used for tracking of vehicle-like objects as can be found in[113, 66, 114, 88, 115].

IMM filter aims to generate better estimates for anobject with ambiguous dynamic behaviour. This approach used by IMM is as follows: by using multiple models representing different potential target manoeuvres to run state estimation filters and subsequently rank the most probable estimation results. Typically, the output of IMM is either a probability-weighted combination of the individual filters or the output of the filter with the highest probability. IMM consists of j filters running in parallel and each filter use different motion model to represent state evolution and measurement. It means for a single track, j state filter(s) is to be maintained.

First, the predicted state vectors and covariance matrices are generated by each individual filter estimates with respect to the predicted model probabilities. Next, each of the estimates is compared to current measurement to update the model-match probability. During the update step, the computed probabilities of the dynamical target model are added to the updated state vector and error covariance matrix information produced by the corresponding filter.

The mode probabilities are then used in the so-called merge (or combination) and mixing step. During the merge stage, the state vectors and covariance matrices of each motion model are merged into a single state vector and covariance matrix. Finally, in the mixing process, new filtered state estimates, error covariance matrices and corresponding model probabilities are computed for each model using weighted state estimates.

The derivation is as follows, we assume that the system with motion model uncertainty is evolving as Jumping Markov Linear Systems (JLMS), the stochastic state space system dynamics and measurement equation representing each model j are then defined as

$$\begin{aligned} \mathbf{x}_{k+1} &= f_j(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_{j,k} \\ \mathbf{z}_k &= h_j(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_{j,k} \end{aligned} \tag{3-19}$$

where $j = 1, 2...r$ is part of model set $\mathbb{M} = \{\mathbf{M}_j\}_{j=1}^{r}$ and state variables and noise assumption is identical with that of (3-2). The IMM is one variant of JMLS which employ r amount of motion models run in parallel for state estimation. Since IMM is in practice typically uses implementation of Kalman Filter[97], it can be seen as adaptive switching implementation of Kalman filter.

In the Bayesian framework, we can infer the posterior pdf. of IMM as

$$p(\mathbf{x}_k, M_k|\mathbf{z}_k) \tag{3-20}$$

that is given all measurements $\mathbf{z}$ up until time k, with discrete state variable $\mathbf{x}_k$, and M_k for the motion mode we can infer the estimate of joint pdf. We can further decompose (3-20) using conditional probability lemma and further rewrite it to becomes

$$\begin{aligned} p(\mathbf{x}_k, M_k|\mathbf{z}_k) &= p(\mathbf{x}_k, M_k|\mathbf{z}_k)p(M_k|\mathbf{z}_k) \\ p(\mathbf{x}_k, M_k|\mathbf{z}_k) &= \sum_{j=1}^{r} p(\mathbf{x}_k, M_{j,k}|\mathbf{z}_k) \underbrace{p(M_{j,k}|\mathbf{z}_k)}_{\mu_{j,k}} \end{aligned} \tag{3-21}$$

where $\mu_{j,k}$ is the posterior mode probability that the object motion matches the dynamics of motion model ran by filter j at time k. In recursive form, the pdf. becomes[97, 116]:

$$p(\mathbf{x}_{k-1}, M_k|\mathbf{z}_{k-1}) = \sum_{j=1}^{r} p(\mathbf{x}_{k-1}, M_{j,k}|\mathbf{z}_{k-1})\mu_{i|j,k-1} \tag{3-22}$$

where

$$\mu_{i|j,k-1} = \frac{p_{ij}\mu_{i,k-1}}{\underbrace{\sum_{i=1}^{r} p_{ij}\mu_{i,k-1}}_{\bar{\mu_{j,k}}}} \tag{3-23}$$

p_{ij} is predefined transition probability from model index i to index j (a filter design parameter set in Matrix Π) and $\mu_{j,k}^{-}$ is the mode match probability of filter j at current time step k.

$$\Pi = \begin{bmatrix} p_{1,1} & \cdots & p_{r,1} \\ \vdots & \ddots & \vdots \\ p_{1,r} & \cdots & p_{r,r} \end{bmatrix} \tag{3-24}$$

The IMM complete cycle is given as follows

Mixing

IMM incorporates weighted average of each j-th model filter state $\hat{\mathbf{x}}_{i,k-1}$ to determine a single combined estimates state $\hat{\mathbf{x}}^{*}_{i,k-1}$ and its corresponding variance $P^{*}_{j,k-1}$. This is done by summing up each model's joint posterior density, and the process is called *mixing*. Therefore, $x^{*}_{j,k-1}$ and $P^{*}_{j,k-1}$ are given as

$$\begin{aligned} \hat{\mathbf{x}}^{*}_{j,k-1} &= \sum_{i=1}^{r} \mu_{(i|j),k-1}\hat{\mathbf{x}}_{i,k-1} \\ P^{*}_{j,k-1} &= \sum_{i=1}^{r} \mu_{(i|j),k-1}\hat{\mathbf{x}}_{i,k-1}[\mathbf{P}_{i,k-1} + (\hat{\mathbf{x}}_{j,k-1} - \hat{\mathbf{x}}^{*}_{j,k-1})(\hat{\mathbf{x}}_{j,k-1} - \hat{\mathbf{x}}^{*}_{j,k-1})^{T}] \end{aligned} \tag{3-25}$$

Prediction

The prediction is made based on the mixed initial states $\mathbf{x}^{*}_{j,k-1}$ and $\mathbf{P}^{*}_{j,k-1}$ by the individual j-th Kalman filter employing its own motion model. The procedure follows ordinary Kalman Filter prediction (refer to Subsection 3-3-2).

The resulting model-specific predicted states are to be denoted as $\mathbf{x}^{-}_{j,k}$ and $\mathbf{P}^{-}_{j,k}$. In addition, we also have model-specific predicted measurement $\hat{\mathbf{z}}^{-}_{j,k}$ and innovation covariance $\mathbf{S}_{j,k}$.

Updates

Likewise, the filter update procedure is to be done by model specific Kalman filter (see Subsection 3-3-2). The resulting posterior states and innovation covariances are denoted by $\hat{\mathbf{x}}_{j,k}$ and $\mathbf{P}_{j,k}$

The mode probability reflects the degree of fitness of current measurement to active model j, it is updated by

$$\mu_{j,k} = \frac{\lambda_{ij}\mu^{-}_{j,k}}{\sum_{i=1}^{r}\lambda_{ij}\mu^{-}_{i,k}} \tag{3-26}$$

where $\lambda_{j,k}$ is the Gaussian likelihood of the measurement given by

$$\lambda_{j,k} = \frac{1}{\sqrt{|2\pi\mathbf{S}_{j,k}|}} e^{0.5(\mathbf{z}_k - \hat{\mathbf{z}}^{-}1_{j,k})^T \mathbf{S}^{-1}_{j,k}(\mathbf{z}_k - \hat{\mathbf{z}}^{-}_{j,k})} \tag{3-27}$$

Combination

The filter output of each j filter is then (re-)combined as a single states $\hat{\mathbf{x}}_k$ and $\mathbf{P}_k$ by

$$\begin{aligned} \hat{\mathbf{x}}_k &= \sum_{i=1}^{r} \mu_{j,k}\hat{\mathbf{x}}_{j,k} \\ \mathbf{P}_k &= \sum_{j=1}^{r}(\mathbf{P}_{j,k} + (\hat{\mathbf{x}}_{j,k} - \hat{\mathbf{x}})(\hat{\mathbf{x}}_{j,k} - \hat{\mathbf{x}}_k)^T) \end{aligned} \tag{3-28}$$

Schematically, the IMM cycle can be observed in Figure 3-6.

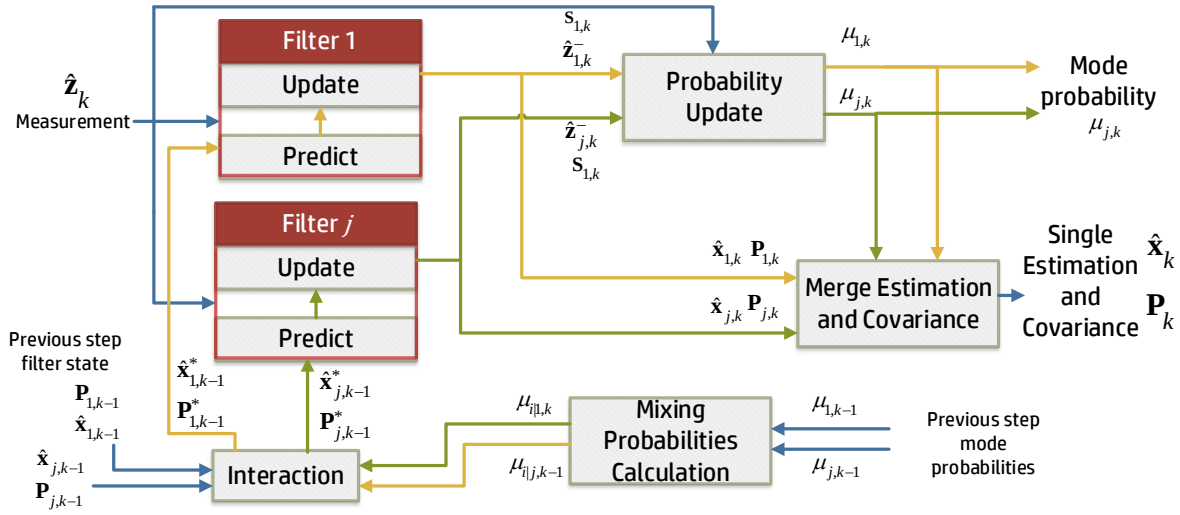


Figure 3-6: Schematics of IMM Filter. The IMM can be extended to $j-th$ filter with r amount of different motion model.

3-3-4 Data Association Filter

KF (or its variant) offers the ability to provide optimal estimates of an object track. Notwithstanding, due to measurement (i.e. detections result) uncertainty there is *no* guarantee that the tracked object is actually a relevant object, or even if it is an existing object in the first place. Therefore, a subsequent classification and validation of the estimated track are simply necessary.

Data Association (DA) is a process of associating the detection result into a tracking filter. There are two classes of DA filter: the deterministic filter and the probabilistic filter. Representative of deterministic DA filter is Nearest Neighborhood Filter (NNF) algorithm which updates each object with the closest measurement relative to the state. NNF associates object with known track based on the shortest Euclidean or the Mahalanobis distance between the measurement and track.

The probabilistic DA filter that is very well-known in object tracking literature body is the eponymous Probabilistic Data Association Filter (PDAF)[29]. The PDAF perform a weighted update of the object state using all association hypotheses in order to avoid hard, possibly erroneous association decisions commonly encountered in the use of NNF algorithm. The erroneous association is often found during the scenario in which multiple measurements is located close to each other (i.e. clutter) and results in single measurement being used to incorrectly update all other nearby objects.

The summarized differences between NN and PDA can be seen in Table 3-1. PDA is also one of the most computationally efficient tracking algorithms among clutter-aware tracker[97], for instance when compared to MHT[117]. Due to its attribute, the PDA filter will be especially relevant in the urban MOT, and it will be discussed with details in the next section.

Table 3-1: Summary of DA filter classification

Data Assocation	Single Target		Multi-Target	
Association schema	n-1-association (local): Detections are associated to single track		n-m-association (global): Detections are associable to any track	
Deterministic vs Probabilistic	Nearest Neighbor	Probabilistic Data Association Filter (PDA)	Global Nearest Neighbour	Joint Probabilistic Data Association Filter (JPDA)
Assignment decision schema (hard vs. soft)	pick closest detection for each track	use probabilistic weighting based on distance	enumerates overall association hypotheses and chooses the one with the smallest sum of distances	use marginalized joint probabilistic weighting based on distance

3-4 Probabilistic Data Association Filter

It is useful to start with PDAF formulation to understand how the joint probability is incorporated to PDAF. For simplicity, we assume a conventional KF with a linear system function and measurement for the following derivation.

PDAF and its derivative all use the incoming measurement inside a validation gate to approximate the probability distribution function of the tracked object after each update. This is done by assuming the measurement follows a Gaussian probability distribution. Following[29], the PDAF itself is working based on the following assumptions:

1. Only one target of interest is present, whose state $\mathbf{x} \in \mathbb{R}^{n_x}$ is assumed to evolve in time according to the equation

$$\mathbf{x}_{k-1} = f_{k-1}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1} \tag{3-29}$$

   with the true measurement $\mathbf{z}_k \in \mathbb{R}^{n_z}$ given by:

$$\mathbf{z}_k = h_k \mathbf{x}_k + \mathbf{w}_k \tag{3-30}$$

   where $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are zero-mean white Gaussian noise sequences with covariances matrices of $\mathbf{Q}_{k-1}$ and $\mathbf{R}_k$.

2. The tracks are initialized.

3. The past information through time $_{k-1}$ about the target is summarized approximately by a sufficient statistic in the form of the Gaussian posterior

$$p[\mathbf{x}_{k-1}|\mathbf{z}^{k-1}] = \mathbb{N}[\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}] \tag{3-31}$$

4. If the target was detected and the corresponding measurement fell into the validation region, then, according to $(3-30)$, only maximum of one of the validated measurements can be target originated.

5. All non-object originated measurements are assumed to be originated from a clutter that is uniformly distributed in space and Poisson distributed in time.

The PDAF algorithm is divided into 4 main stages, Prediction, Measurement Validation, Data Association and State Estimation. The prediction and state update states are similar to conventional KF. However, the measurement validation and data association is a specific part of JPDAF. One cycle execution of PDAF algorithm can be observed in Figure 3-7.

3-4-1 Prediction

The PDAF predict the state and covariance matrices one step ahead of time just like KF, given by

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= f_{k-1}(\hat{\mathbf{x}}_{k-1|k-1}) \\ \hat{\mathbf{z}}_{k|k-1} &= h_k(\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_{k|k-1} &= f_{k-1}\mathbf{P}_{k-1|k-1}f'_{k-1} + \mathbf{Q}_{k-1} \end{aligned} \tag{3-32}$$

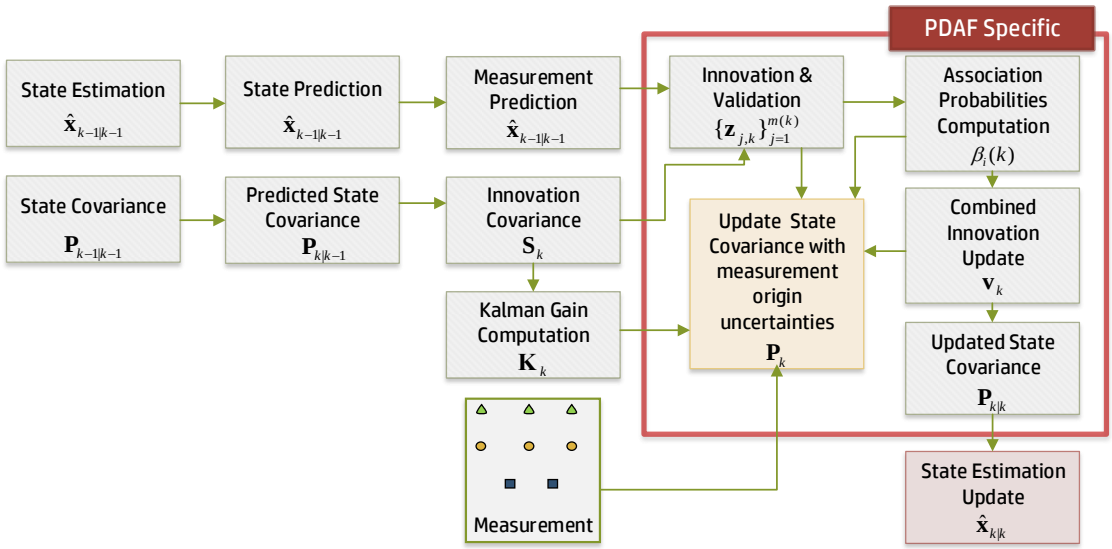


Figure 3-7: One cycle of the probabilistic data association filter. The part specific to JPDAF is highlighted in red box. The state-estimation and covariance calculations are coupled due to covariances dependences on the innovations. (adapted from Bar-Shalom[29])

where $\mathbf{P}_{k|k-1}$ is obtained from (3-31) and the innovation covariance matrix corresponding to correct measurement given as

$$\mathbf{x}_k = h_k \mathbf{P}_{k|k-1} h_k' + \mathbf{R}_k \tag{3-33}$$

3-4-2 Measurement Validation (Gating)

Prior to the measurement being passed to a DA filter, a "weeding" process of bad measurement called *validation gating* or *windowing* is to be done. In essence, detections that exceed a predefined distance threshold to the closest track are excluded. Those detections are considered as *non-associable*, (i.e. it is very unlikely that those detections represent objects which are already being tracked). Instead, those detections are considered to be potential objects that are not yet tracked and thus are used to create new object hypotheses. In addition to using those non-associable detections as a source for new tracks. The illustration of gating can be seen in Figure 3-8.

The gating is typically implemented as measurement selection problem[97]. Gating selects a subset of measurements which, given a priori knowledge that the object exists and is detected contains the object detection with high probability. This probability is termed the "gating probability". A common approach of measurement gating is to assume the measurements are distributed according to a Gaussian, so the hyper-ellipsoid gate is used as gating region and the Mahalanobis distance of the obtained measurement state vector is computed and compared against that of predicted state vector[115]. A threshold based on inverse value χ^2 square distribution is often used[97], this threshold is called gate level.

The measurement validation is done by the gating area given by elliptical region[29]

$$(V)(k,\gamma) = \{\mathbf{z} : [\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}]' \mathbf{x}_k^{-1} [z - \hat{\mathbf{z}}_{k|k-1}] \leq \gamma\} \tag{3-34}$$

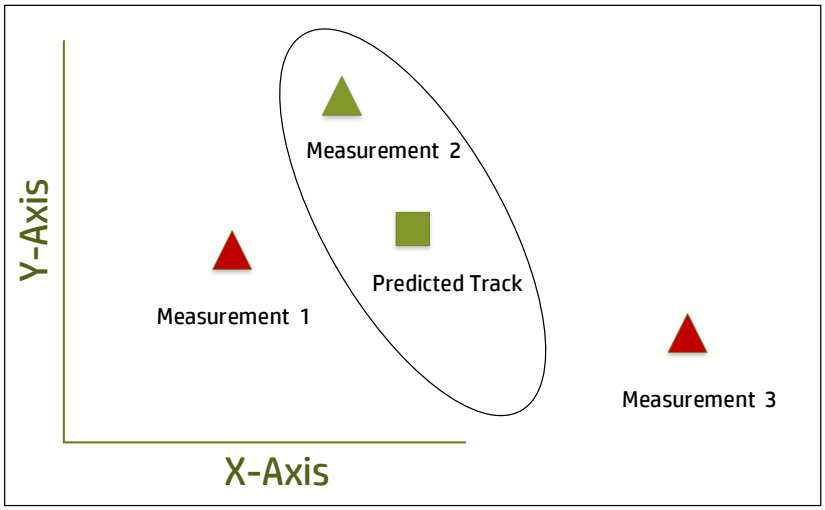


Figure 3-8: Gating process. Measurement 1 and Measurement 3 are considered to be unlikely to be sourced from predicted track, and thus will be discarded

where γ is the value of gate threshold equal to inv-$\chi^2(P_G)$, P_G is the probability that the gate contains the true measurement if detected, and $\mathbf{S}_k$ is the covariance of the innovation corresponding to the true measurement. The volume of the validation region for q-dimensional measurement is expressed as[118]

$$V_k = \frac{\pi^{\frac{q}{2}}}{\Gamma(\frac{q}{2}+1)}\sqrt{|\gamma \mathbf{S}_k|} \tag{3-35}$$

Finally, the set of validated measurements given as:

$$\mathbf{z}_k = \{z_{i,k}\}_{i=1}^{m_k} \tag{3-36}$$

3-4-3 Data Association

The non-parametric PDAF is used, we assume a diffuse prior clutter model[29] clutter density, which is suitable for heterogeneous clutter environment[119]. The association probabilities β of measurement i at time k is given as[118]

$$\beta_{i,k}(x) = \begin{cases} \dfrac{e^{\frac{1}{2}(\mathbf{z}_{m,k}-\hat{\mathbf{z}}_{m,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{m,k}-\hat{\mathbf{z}}_{m,k|k-1})}}{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D} + \sum_{j=1}^{m_k} e^{\frac{1}{2}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})}}, i = 1,\ldots,m_k \\ \dfrac{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D}}{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D} + \sum_{j=1}^{m_k} e^{\frac{1}{2}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})}}, i = 0 \end{cases} \tag{3-37}$$

P_D is the detection probability, P_G is the validation gate probability.

3-4-4 State Estimation

The state update equation is given by

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k v_k \tag{3-38}$$

with combined innovation $\hat{v}_k$ and filter gain $\mathbf{K}_k$ given as

$$\hat{v}_k = \Sigma_{j=1}^{m_k} \beta_{j,k}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}) \tag{3-39}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} h_k' \mathbf{x}_k^- 1 \tag{3-40}$$

and associated covariance of updated state given by

$$\mathbf{P}_{k|k} = \beta_{0,k}\mathbf{P}_{k|k-1} + [1-\beta_{0_k}](\mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T) + \mathbf{K}_k(\Sigma_{j=1}^{m_k}\beta_{j,k}v_{j,k}v_{j,k}^T - v_k v_k^T)\mathbf{K}_k^T \tag{3-41}$$

It can be seen that PDAF algorithm resembles conventional KF updates in term of state prediction and innovation. In addition, the probability weighting $\beta_{0,k}$ of correct measurement is accounted in (3-41) during update process. Note that $\mathbf{P}_k$ will get updated by probabilities weighting $1-\beta_{0_k}$ which correspond to the probabilities that correct measurement is received.

3-5 JPDA: Tracking Multiple Target in Clutter

The MOT problem can be treated as a single object tracking problem with multiple trackers run in parallel. However, this only work if the objects are moving independently. In urban areas, road users typically move in formation-like motion due to traffic conditions. Although the locations of tracked objects are different; velocity and acceleration may be nearly identical. (i.e. their motions are highly correlated). This situation introduces ambiguities in associating multiple measurements to multiple tracks. The situation is depicted in Figure 3-9.

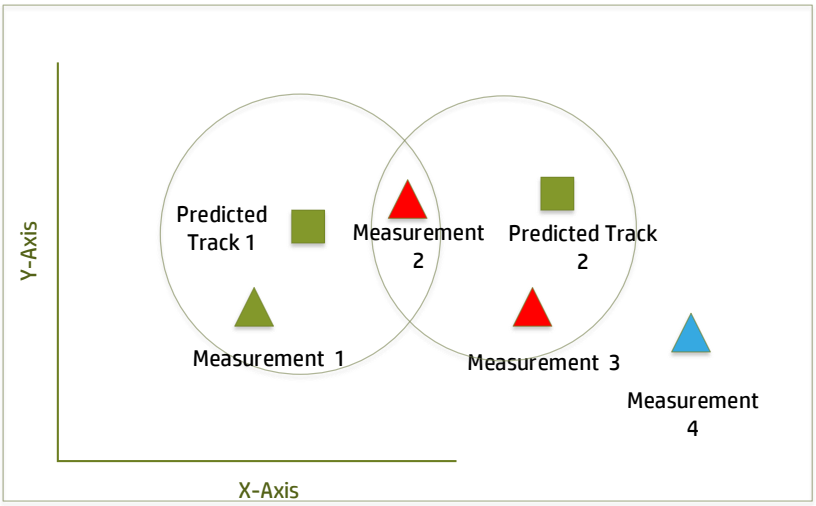


Figure 3-9: Clutter situation with gating. Measurement 2 could belong to Predicted Track 1 or 2, and a one-to-one association is expected. Wrong association will affect the tracker state estimation performance and might even induce track lost.

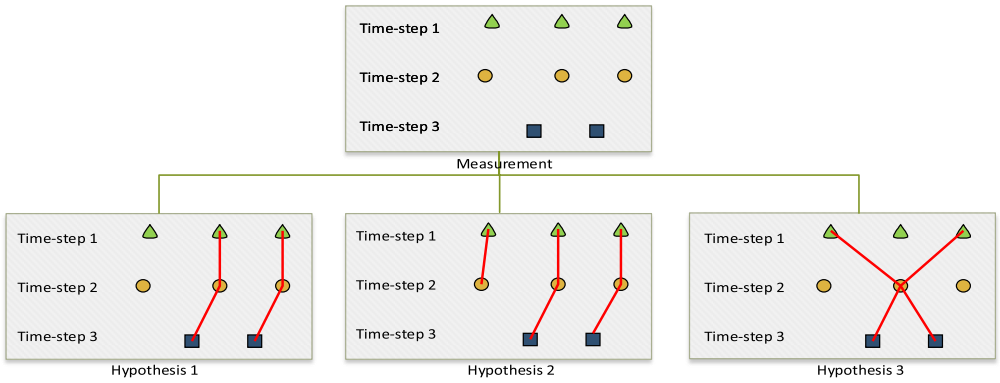


Figure 3-10: Data association possibilities tree for MOT. Note that different combination results in several different track evolution.

To address this issue, the clutter-aware version of PDAF, the Joint Probabilistic Data Association Filter (JPDAF) is the extension of PDAF that operates under assumption that object is tracked under clutter, the situation comes up when different tracks share similar measurements (c.f. Section 4-6-1). The 'joint' term refers to the inclusion global association hypotheses to calculate the weighting.

Probabilistically speaking, since the event that a measurement corresponds to track detection is mutually exclusive across tracks, but not necessarily mutually independent, the optimal filter update operations have to consider all possible tracks. Ditto, the allocation of measurements to possible tracks must be considered jointly or globally. Therefore the following assumptions are made, following[29]:

1. The number of established targets (i.e. tracks) N in the clutter is known a priori.
2. Measurement from one target can fall in the validation region of a neighbouring target and acts as a persistent interference.
3. The past of the system is summarized by an approximate of sufficient statistic consisting of state estimates, which are given by approximate conditional means, along with covariances for each target.
4. The states are assumed to be Gaussian distributed with means and covariances according to the above approximate sufficient statistic
5. Each target has a dynamic model (3-32) and measurement model (3-33).

Consider that we have some measurements j and also some predicted targets t. The Gaussian likelihood $A_{t,j}$ of associating j with t in a joint event θ is given by [118]

$$A_{j,t} = \frac{e^{-\frac{1}{2}(\mathbf{z}_{t,j|k} - \hat{\mathbf{z}}_{t,j|k-1})^T \mathbf{S_{t,j}}^{-1} (\mathbf{z}_{t,j|k} - \hat{\mathbf{z}}_{t,j|k-1})}}{\sqrt{\det(2\pi \mathbf{S}_{t,j})}} \tag{3-42}$$

Here we consider all numbers of clutter measurements are equally likely and the joint association $\gamma(\theta)$ probabilities become

$$\gamma(\theta) = \frac{\Phi!}{V^{\Phi}} \prod_{j} (A_{t,j})^{\tau_j} \prod_{t} (P_D^t)^{\delta_t} (1 - P_D)^{1-\delta_t} \tag{3-43}$$

where V is the volume of the surveillance region in which the measurements not associated with a target are to be assumed uniformly distributed, Φ is the number of false measurements in event A, τ is binary measurement indicator, δ is binary target detection indicator, and c is the normalization constant.

Consider Figure 3-9, if for example we have feasible joint association Measurement 1 to Track 1 (θ_{11}), Measurement 2 to clutter (θ_{02}) and Measurement 3 to Track 2 (θ_{23}), then $\delta_1 = 1$, $\delta_2 = 1$, $\tau_1 = 0$, $\tau_2 = 0$, $\tau_3 = 1$, and $\Phi = 1$, then $\gamma(\theta)$ becomes

$$\gamma(\theta) = \frac{1}{V} P_D^2 A_{11} A_{23}$$

and this also applies to other feasible joint events $(\theta_{tj}, \theta_{tj}, \theta_{tj})$, where $j = (1, \ldots, m_k)$ and $j = (1, \ldots, N)$.

Assuming that states conditioned on the past observations are mutually independent, the JPDA association probabilities β is simply the marginalization of all feasible association probability $P\{\theta_k | Z_k\}$,respectively given by

$$\beta_{jt}(l) \equiv= \sum_{\theta : \theta_{jt} \in \theta} P\{\theta_k | Z_k\} \tag{3-44}$$

$$P\{\theta_k | Z_k\} = \frac{\gamma(\theta)}{\sum\limits_{\theta} \gamma(\theta)} \tag{3-45}$$

Using (3-44) then the state estimation equations are to be decoupled for each target and executed the same way as in (3-38). Finally, the state estimation is done exactly the same with the PDA process.

Chapter 4

MOT System Design and Implementation

4-1 Overview

This chapter elaborates the design process and application practicality of the MOT framework based on the theoretical foundation and state-of-art given in preceding chapters. First, the system architecture will be given to give the reader a bird-view of the implementation. Subsequently, the development platform is presented to inform readers how the proposed framework is designed and implemented. Finally, each of the building blocks in the system will be visited to investigate the individual underlying reasons and how the theory is implemented in practice.

4-2 System Architecture

The MOT framework is divided into two major components based on the functional objective: (1) Detector which aims to produce meaningful, unambiguous segmented objects derived from raw LIDAR data and (2) Tracker, whose task is to assign and maintain dynamic attributes of detected objects across all time frame.

The component hierarchy and input-output flows can be seen in Figure 4-1. The input of the system is the raw data from 3D LIDAR scan in point cloud representation of environmental spatial data in Cartesian coordinate, while the output is a list of objects embedded with context aware attributes, namely the bounding boxes, trajectories and static/dynamic classification. Detector consists of 3 sub-components which similarly, represent the subtask of detection process: the ground removal, to eliminate object of non-interest and reduce dimensionality of raw data, clustering, which segment the point clouds into collection of coherently grouped points (i.e. the object), and bounding box fitting, which embed a uniform object dimension and general direction heading information to each cluster.

Tracker retrieves the list of bounding boxes and is responsible for keeping itself updated for the bounding box spatial and dimensional evolution on each time step change, handled by Position Tracker and Box Tracker sub-component, respectively. Note that the spatial evolution is expected to change according to motion model, while the box dimension should stays constant with changing heading. Both evolutions are perturbed by noise and uncertainties, therefore position tracker requires Bayesian filtering to reject the disturbance. Tracker also stores the output state of every track iteration in box history, in turn, the Box Tracker relies on past information from Box History to filter out noise before updating the bounding box.

The MOT tasks are largely sequential and inter-related, as each component relies on the output of preceding components to perform its task. To exploit this behaviour, the so-called bottom-up top-down approach[25] is used. The Detector ("top" component) is to exchange information in a feedback-loop fashion with Tracker ("bottom" component) to reduce false detection. Since track-by-detection paradigm is used, this approach also augments Tracker component task.

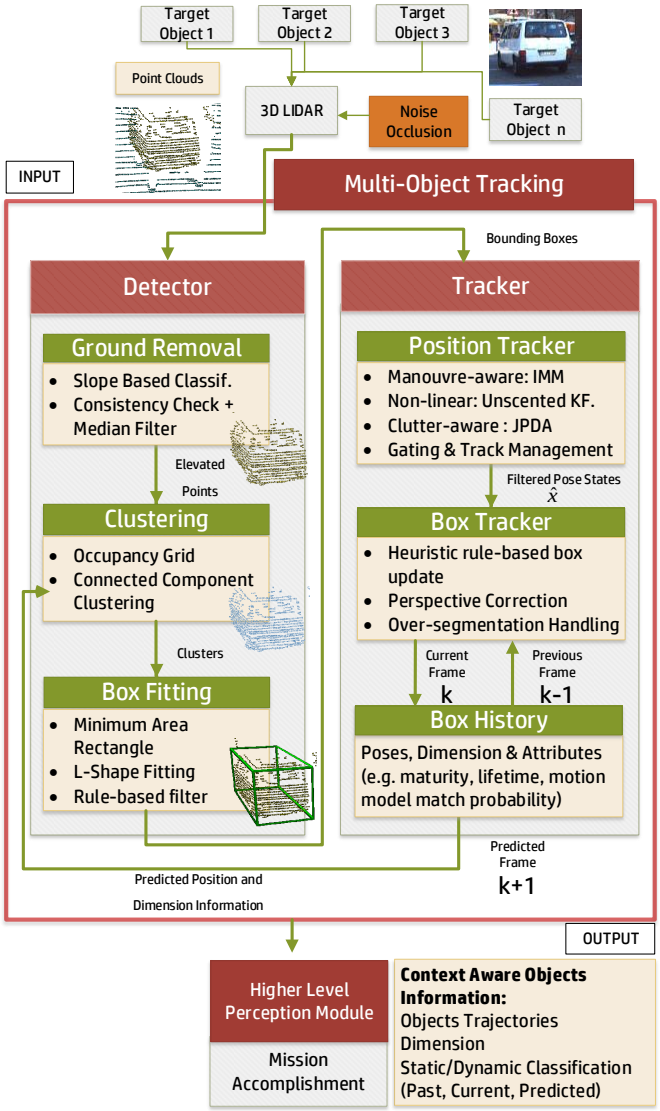


Figure 4-1: Multi Object Tracking system architecture

4-3 Development Framework and Methodology

The implementation is written chiefly in C++ programming language using QT toolkit[120] binding. The rationale of cross-platform QT use is to allow the implementation to be natively executed on other platforms in the future. In this case, the embedded platform typically used in car onboard computer such as Nvidia Drive PX[121]. The reference development and benchmark device is a standard desktop-grade developer PC (Intel Core i7-7700K with 8GB of RAM, running Ubuntu GNU/Linux 16.0.4.3 LTS). Again, we try to emulate car environment where in the industry most high performance vehicle embedded computer are running on Operating System based on Linux kernel (see [12, 8, 102]).

The use of C++ on the other hand, is to enable the MOT to be run as pre-compiled native binary program (as opposed to interpreted script) to avoid overhead in runtime. Third party library used namely are Point Cloud Library (PCL)[122] for Point Cloud manipulation, OpenCV[123] for shape and bounding box generation, Eigen[124] for Matrix operation, and KITTI Development kit[73] to programmatically access the raw dataset and ground truth.

While the majority of the implementation is written in C++, some matrix operation-intensive logic and filter design are written in MATLAB[125] environment. MATLAB's Embedded Coder is used to convert logic written in MATLAB script into a native, linkable C++ static library. This approach allows rapid algorithm change, debugging and more importantly tuning and evaluation of the designed filter using MATLAB built-in toolbox and plotting tool. The development workflow is summarized in Figure 4-2.

Referring to Figure 4-1 in the previous section, the whole part of Position Tracker subcomponent (that is IMM-UK-PDAF and Track Management) is written entirely in MATLAB, while the others are written in C++. Therefore, the implementation has a mixed source codebase with approximately 15% of it is written in MATLAB and 75% in C++.

To inspect the raw LIDAR data and tracking result visually, the "Visualiserung" (internal program from PEM Chair RWTH Aachen) is used. Visualiserung is a 3D LIDAR data visualizer program which is capable of simultaneously replaying LIDAR 3D points, bounding boxes, ego-vehicle movement and camera image in frame-per-frame basis (see Figure 4-3). The MOT is written to output tracking hypotheses to the Visualiserung data format.

The multi-object tracking system (inclusive of the detection process) is intended to be run on a car embedded computer, which translates into the requirement of low power and constraint of computational performance limited by cooling system[126]. Within these common constraints, it is imperative to ensure that the system can be run in a real-time manner given the available computing power. The term real-time itself here refers to the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical time when these results are produced[65]. Therefore, to some degree, the accuracy the execution time has to be deterministic, or at least predictable.

In this thesis, the author aims that the designed tracking system is able to finish complete a cycle of detection and tracking faster than the LIDAR sensor sampling rate (10 Hz)[18]. The evaluation of computation time can be found in Appendix A-2.

Finally, for metrics-based evaluation purpose, the MOT binary is instructed to export all tracking hypotheses, trajectories, and its attributes to plain CSV files. The benchmark script is written in MATLAB with integrated use of KITTI MATLAB Development kit.

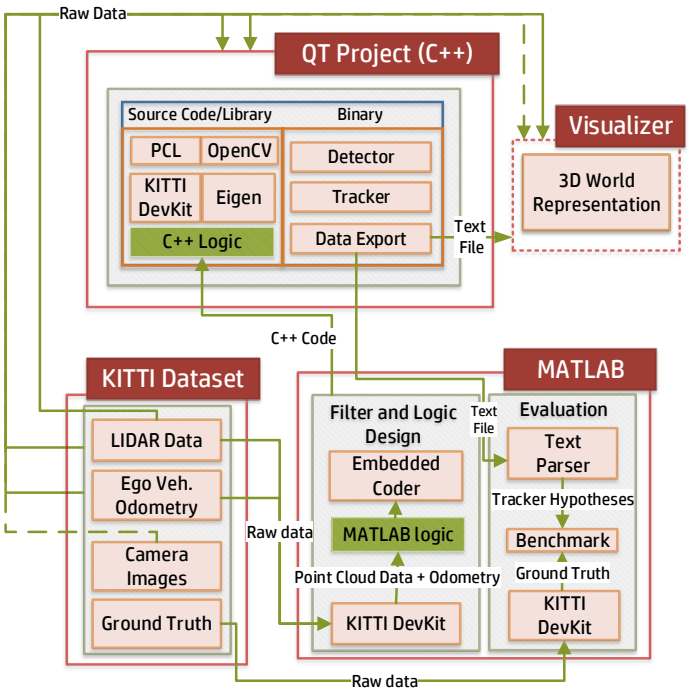


Figure 4-2: MOT Development Framework. Note that the green boxes correspond to the vast majority of Chapter 4 content. Additionally, dashed elements indicate they are not developed as part of the framework and only used as visual aid

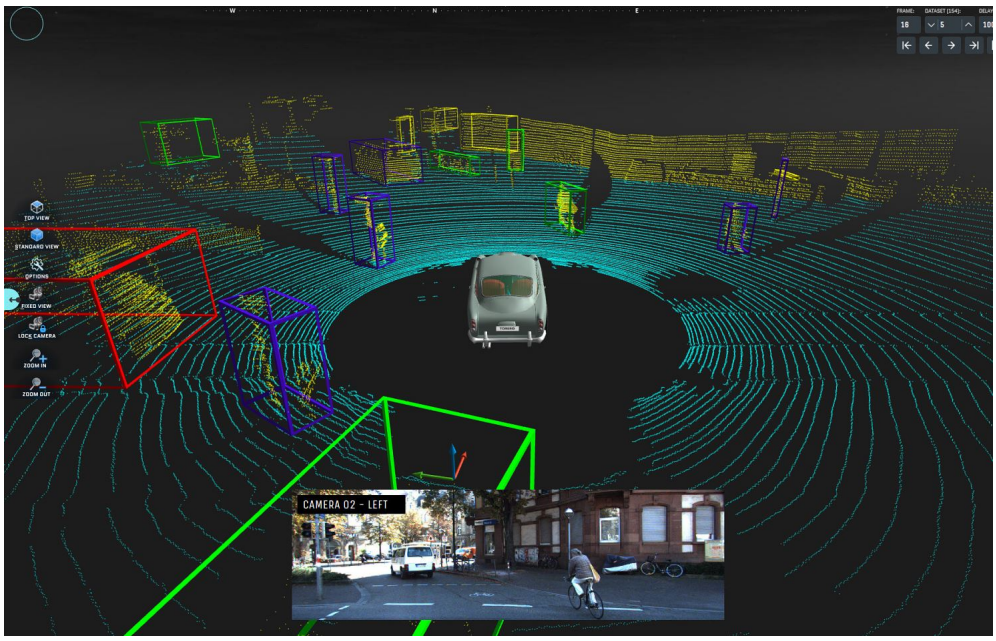


Figure 4-3: 3D world representation of LIDAR data and tracking hypotheses (coloured boxes). The camera image is used as a visual aid

4-4 Detector

The detector component is responsible for initial pre-processing and fitting of bounding boxes which later are to be passed to the tracker. Each step on detection process is going to be presented in the following subsections. Note that the parameters of the detector used in implementations can be found in Appendix C.

4-4-1 Ground Removal

The raw LIDAR data consists of approximately 3.0×10^5 points and a large majority of the points belongs to the ground plane which does not carry meaningful information for tracking purpose, the first step in measurement pre-processing is thus the ground removal. In this process, the removal is done using slope-based channel classification augmented with consistency check and median filtering. The LIDAR measurement with the ground removed is called *elevated points.*

To achieve this purpose slope-based channel classification is utilised following[14] which is shown to handle partial occlusion on LIDAR data efficiently. The slope based channel classification determines the ground height by compartmentalizing the LIDAR point clouds and comparing the difference of height (i.e. *slope*) of the successive compartment.

The procedure is as follows: first, the raw LIDAR scans is divided into a polar grid with $m_{bins} \times n_{channels}$-cells with minimum radius r_{min} being the radius closest to ego vehicle and r_{max} being the farthest radius also from ego-vehicle (see Figure 4-4). The minimum radius is the radius in which the reflection of the ego vehicle is no longer can be seen while the maximum radius is determined by the effective range of the sensor. Mapping of each point cloud data $p_i = \{x_i, y_i, z_i, I_i\}$ (see Subsection 2-2-3) to each channel and bin is thus given as:

$$\begin{aligned} channel(p_i) &= \frac{atan2(y_i, x_i)}{2\pi} \\ bin(p_i) &= \frac{\sqrt{x_i^2 + y_i^2} - r_{min})}{2\pi} \end{aligned} \tag{4-1}$$

Note the height information of the point coming from z_i is stored as the mapped cell's attribute. The lowest z_i point, hereby called *prototype point* is to be used as 'local' ground to determine the lowest possible point for all cells. Additionally, the absolute (local) height in the cell can be enumerated by computing the difference between lowest z_i and the highest z_i within the same cell.

Next, we define the interval $[T_{hmin}T_{hmax}]$ of possible ground height h_i for threshold-based classification. Since the sensor's mounting height *above* ground level h_{sensor} is known a priori, we can extrapolate that the closest point to the sensor must be situated above ground point. Thus h_{sensor} is then used as the T_{hmax} for a point to be considered as ground.

The cell information is filled from the bins which are closest to ego-vehicle to the farthest, if the prototype point of a cell lies inside the interval $[T_{hmin}T_{hmax}]$ then it is set as the h_i of that cell. If the prototype point is higher than T_{hmax}, then the ground level is set to be equal to that of h_{sensor}.

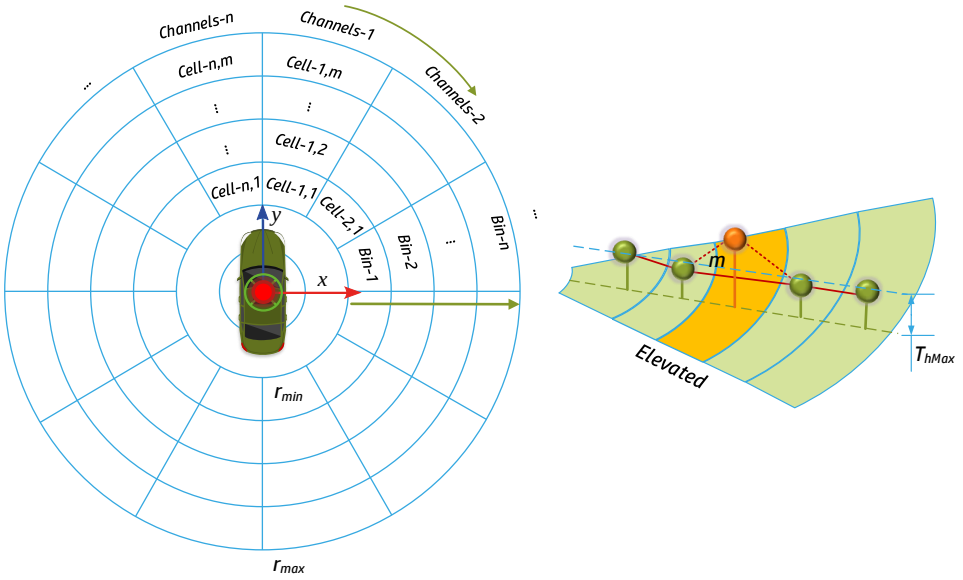


Figure 4-4: Slope-based channel classification. Polar grid mapping (left) and adjacent cells slopes (right). Adapted from[96]

After all cells have been enumerated, a slope (simple Euclidean gradient) m between cells thus can be calculated to inspect if there is sudden height increase between cells. In addition, absolute height difference is also computed as a secondary check to identify possible non-ground object residing in the neighbouring cell. The slopes are illustrated as an orange line in Figure 4-4, normal ground is represented by a green circle, and the elevated point is coloured as yellow due to slope change and height difference with the previous cell exceeding a predefined threshold T_{slope} and T_{hdiff}.

Although this approach is good for smooth terrain, a small, protruding terrain features, such as road bump and grass can still be classified as elevated points. To tackle this Consistency Check and Median Filtering[14] are employed to further flatten the ground plane and yield better ground estimate. Consistency Check is done by iterating non-ground cells which are flanked by non-ambiguous ground cells and then comparing the cells' height consistency with the neighbouring cells. The cell height is compared against a predefined absolute height T_{flat} and its height difference with adjacent cells is compared to threshold $T_{consistency}$. A value below thresholds indicates the cells should belong to the ground and thus they are to be re-classified accordingly.

Median filtering on the other hand deals with missing ground plane information (common due to occlusion), as the name implies, the height value of the missing cell is to be replaced with the median value of neighbouring cells. The polar grid is again iterated to identify ground cells which have missing information but is surrounded by ground cells, the tunable parameter of the filter is kernel (window) size s_{kernel} which indicates the number of neighbouring cells involved.

At the last step, a tolerance value h_{tol} is used during final classification to further smooth the

transition between ground and elevated points in noisy measurement. The whole procedure is summarized in Algorithm 1 (Appendix B).

4-4-2 Object Clustering

Connected Component Clustering is utilised in order to distinguish each possible object in the elevated points. The connected component clustering is originally designed to find the connected region of 2D binary images. However, it is also applicable to LIDAR point cloud[80] since in urban situation traffic object does not stack vertically, and thus the top view of the LIDAR measurement can be treated as a 2D binary image. However, the height information is retained by deriving the difference between the highest and the lowest point in each cluster. The choice of using this approach allows the MOT system to perform in real-time while still preserving the height information of detected objects.

Following two-pass row-by-row labelling[127] written in Algorithm 2 (Appendix B). The approach uses one pass to assign the temporary label of 'connectedness' and the second pass is to replace each label with the unique cluster ID.

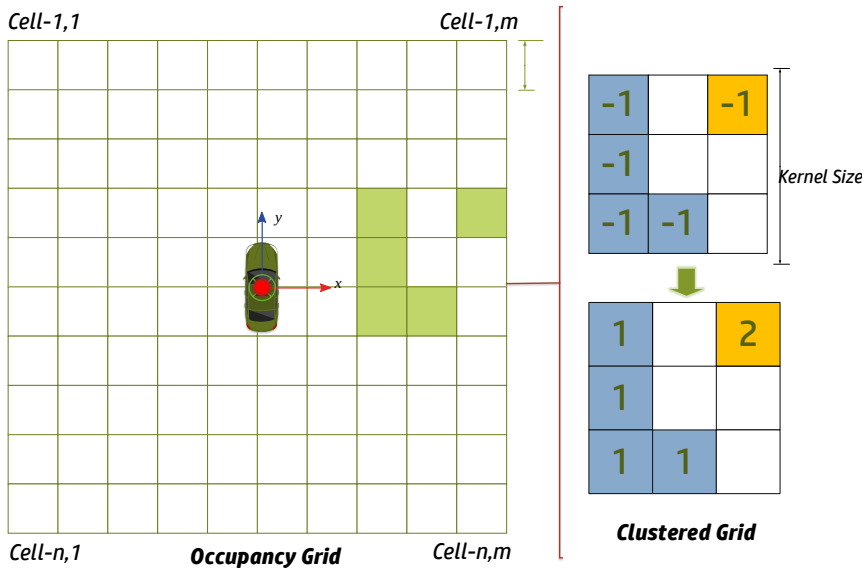


Figure 4-5: Occupancy Grid and Clustering result. Left is the discretized grid and right is clustered grid, there are 2 clusters in section of the grid, with cluster ID 1 and 2

The XY plane of the elevated points is discretized into grids with $m \times n$ cells. The grid is assigned with 2 initial states, empty (0), occupied (-1) and assigned. Subsequently, a single cell in x, y location is picked as a central cell, and the clusterID counter is incremented by one. Then all adjacent neighbouring cells (i.e. $x-1, y+1$, $x, y+1$, $x+1, y+1$ $x-1, y$, $x+1, y$, $x-1, y-1$, $x, y-1$, $x+1, y+1$) are checked for occupancy status and labelled with current cluster ID. This procedure is repeated for every and each x, y in the $m \times n$ grid, until all non-empty cluster has been assigned an ID.

The LIDAR point cloud ground removal and clustering process results can be observed in Figure 4-6.

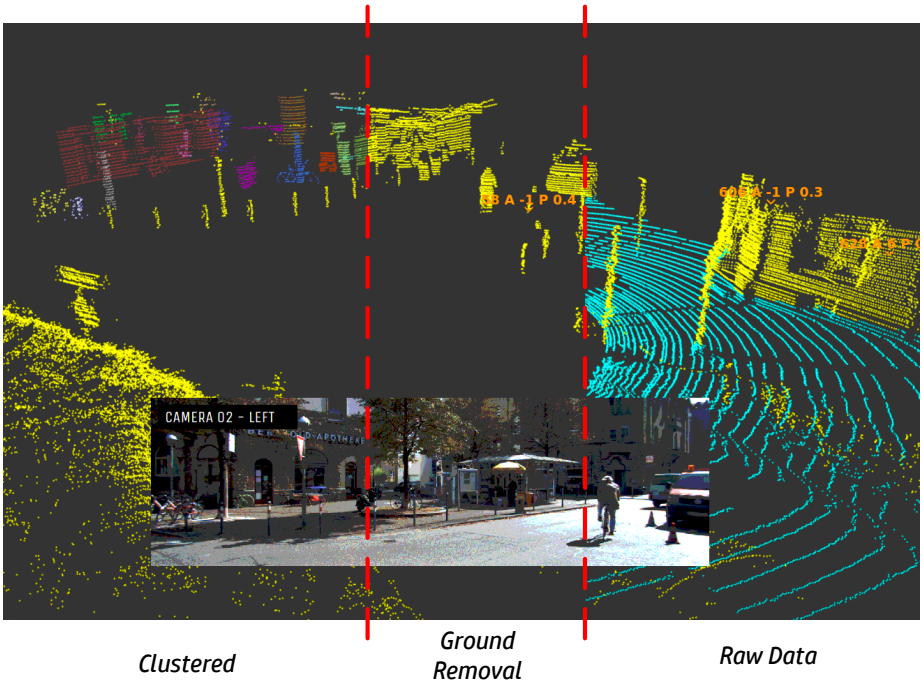


Figure 4-6: Measurement Pre-Processing: Ground Removal and Clustering

4-4-3 Bounding Box Fitting

The clustering process produced candidate object to be tracked. In order to make intuitive semantic information about the object, bounding box is fitted to have uniform dimensional information and yaw direction. A Minimum Area Rectangle (MAR)[128] is applied to each clustered object which results in a 2D box, and when combined with height information retained in the clustering process, becomes a 3D bounding box (see Figure 4-7).

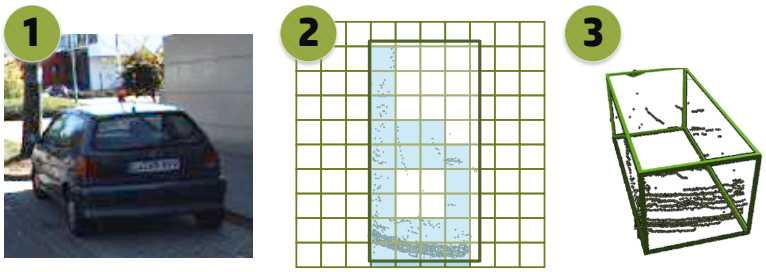


Figure 4-7: MAR box fitting with embedded height

The MAR approach is sufficient for most well-defined measurement of a target object, however it is not guaranteed to correctly enclose partially-occluded object (c.f. Figure 4-8). To tackle this issue, a feature-based L-shape fitting as proposed by Ye[91] is used to deduce correct yaw orientation.

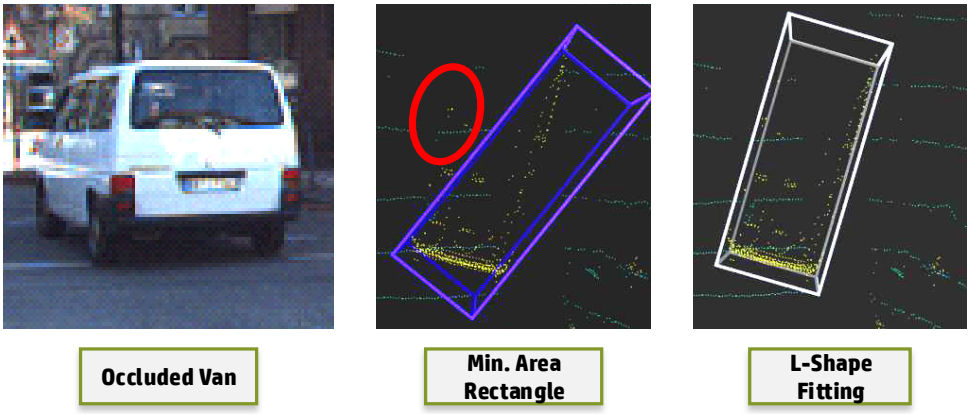


Figure 4-8: Comparison of MAR vs L-shape fiting of occluded object. Notice that some sparse points (red circled) are considered as outlier and the MAR procedure results in an incorrect box, the use of L-shape fitting addressed this issue.

The L-shape fitting procedure is done as follows: first, we select two farthest outlier point x_1 and x_2 which lies on the opposite sides of the object facing the LIDAR sensor. A line L_d is then drawn between the two points, then an orthogonal line L_o is projected from L_d toward the available points. The projected L_o with maximum distance d_{max} and angle close to 90 degrees is then obtained using iteration end-point fit algorithm[129]. The points connected to the orthogonal line then becomes the corner point x_3. Closing a line between x_1, x_2 and x_3 would form an L-shaped polyline. The orientation of the bounding box is then determined by the longest line, as most traffic object (e.g. car, cyclist) is heading in parallel with its longest dimensional side. This procedure is illustrated in Figure 4-9.

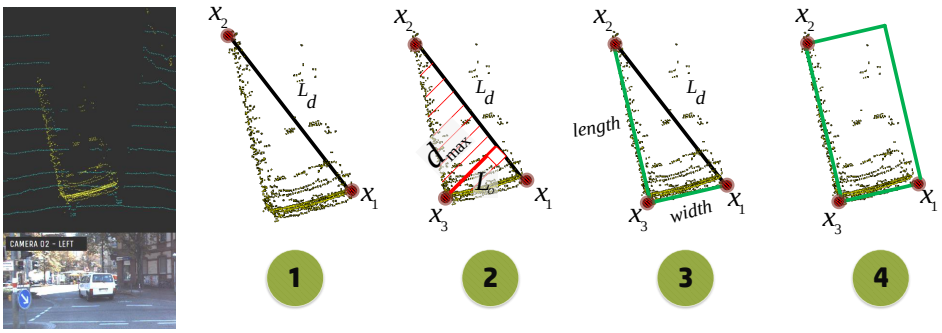


Figure 4-9: L-shape fitting procedure

Note that this approach needs a sufficient amount of measurement points to be able to fit reliable line and is designed for a cuboid-like object. Therefore, the L-shape fitting is only applied on a car-like object, which correspondingly also suffer from occlusion more than smaller objects such as cyclist and pedestrian.

4-4-4 Rule-based Filter

The last step of the detection process is to eliminate majority object of non-interest, such as a wall, bushes, building, and tree. A dimensional thresholding is implemented to achieve this. In addition to 3 standard dimensional sizes (length, width, height), the ratio between length and width is considered to remove disproportionately thin objects such as wall and side rails. Furthermore, amount of LIDAR points and its volumetric density (i.e. point per m^3) are also taken into account.

However, conservative thresholds are used in order to prevent missed detection; it is expected that objects which share a similar dimensional profile, such as pole and pedestrian, or car and large bush are not to be filtered. In addition, recall that occluded object will not have a full dimensional bounding box. Thus, the ratio check is not applicable for over-segmented box-(es) which may belong to object of interest. Therefore, the ratio check is only applied for larger object to distinguish between thin wall-like object and fragment of a real object due to over-segmentation. The rule-based filter is not intended as a catch all measure for false positive, but it is intended to reduce significant number of non-object of interest passed to tracker components.

The list of thresholds and the accompanying descriptions can be seen in Table 4-1. Generated boxes whose attributes are beyond the threshold is to be discarded and will not be passed to tracker component. The results can be observed in Figure 4-10.

Table 4-1: Detector rule-based filter thresholds

No.	Threshold variable	Description
1	$T_{height\_(min\|max)}$	min. and max. height of object
2	$T_{width\_(min\|max)}$	min. and max. width of object
3	$T_{length\_(min\|max)}$	min. and max. length of object
4	$T_{area\_(min\|max)}$	min. and max. top-view area of object
5	$T_{ratio\_(min\|max)}$	min. and max. ratio between length and width
6	$T_{ratiocheck\_l\_(min)}$	min. length of object for ratio check
7	$T_{pt\_per\_m3\_(min)}$	min. point count per bounding box volume.

4-5 Tracker

Ultimately, the output detector component is a bounding box with dimension, yaw and centre position. The next logical step is to maintain an unique and recallable identity of each detected box across all time frame, which is the *raison d'êtr*e of the Tracker component. The said component consists of two separate sub-component that works in tandem: **Position Tracker** and **Bounding Box Tracker**. The later is necessary to preserve dimension since we are implementing point based tracking instead of extended object tracking (recall Section 3-2 for the rationale).

Position tracker uses probabilistic, adaptive Bayesian filtering to deal with uncertainty in movement and association of detected box centre point. While the bounding box tracker utilises logic-based filter based on a heuristic, that is sets of rule obtained by investigating

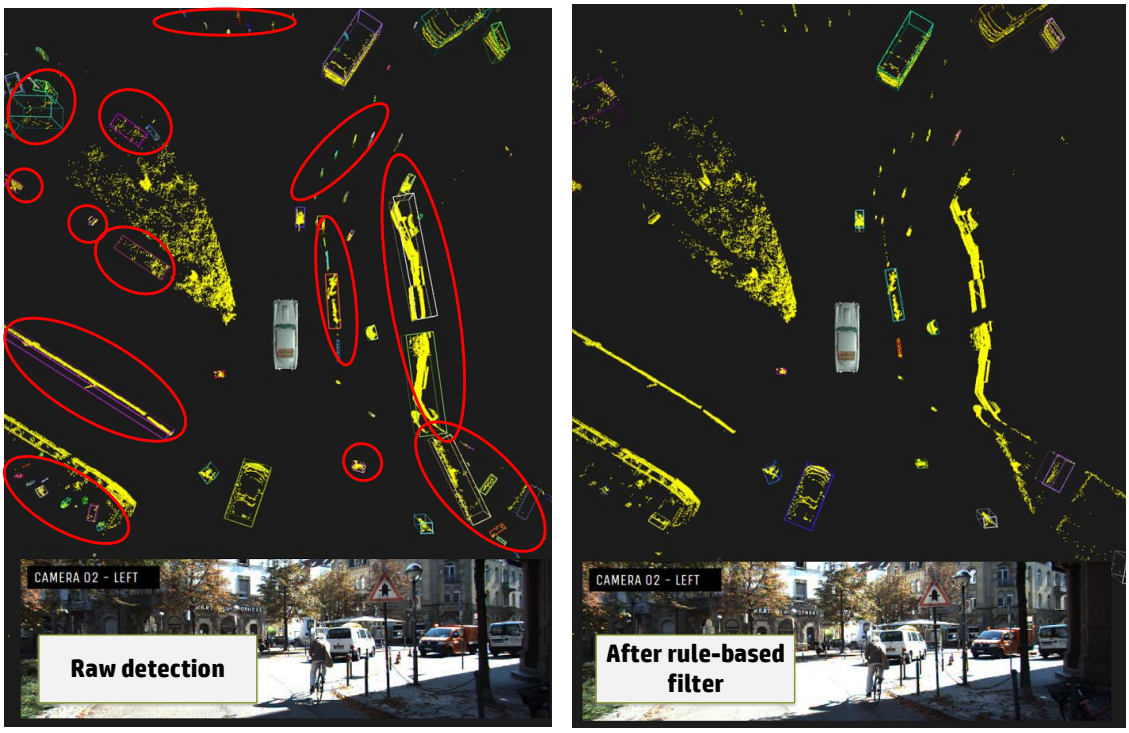


Figure 4-10: Before rule-based filtering: the objects of no-interest are circled in red (left) and after rule-based filtering (right).

the sensor measurement and occlusion characteristics. The details shall be elaborated in the following sections. Note that the Tracker component parameters are listed in Appendix C.

4-6 Position Tracker

There are several challenges that need to be tackled on performing object tracking in urban situation using LIDAR. First, a LIDAR scanner mounted in vehicle navigating in the urban environment would be expected to return incomplete spatial data due to occlusion from other static object or ill-posed perspective of the ego-car (cf. Section 3.2).

Furthermore, a pedestrian, cyclist and vehicle in a crowded situation (e.g. 4-way junctions) are expected to follow multiple motion models, instead of linear constant velocity, as they would be in case of a less-crowded situation. Second, simultaneous tracking of multiple objects in a cluttered environment is as matter-of-fact the basic premise of MOT in the urban situation. To address these two uncertainties, a two different class of tracking filter are combined: the manoeuvring-object tracker and clutter-aware tracker.

The object-manoeuvring filter seeks to address filtering problem in a system characterized by multiple modes of behaviour (i.e. more than one motion models). The clutter-aware filter on the other hand incorporates probabilistic approach in the data association process. Note that both classes of tracker is applied on a different part of MOT steps. This approach can also be found in numbers of tracking literature[113, 81, 112, 130].

4-6-1 IMM-UK-JPDAF

At this point, the requirement of multi-object tracking problem in the urban scenario has been qualitatively defined; the tracker is demanded to be capable of tracking multiple manoeuvring objects coming from uncertain sensor measurement with possible occlusion. These requirements can be translated into different types of uncertainties problems that call for the combined use of different classes of object tracking filter.

First, to handle the behaviour of non-homogenous object, the Interacting-Multiple-Model (IMM) is to be used, also since the kinematic model used is non-linear, the state estimation is to be done using Unscented Kalman Filter (UKF) due to computational power consideration. Finally, to handle tracking of objects in clutter with the possibility of detection noise the Joint Probability Data Association Filter is to be used during data association stage. Together they form a "coupled" filter called IMM-UK-JPDAF. Note that several closely-related implementations such as IMM-UK-PDA[81], IMM-UK-MHT[30], and IMM-PF[131] can also be found for object tracking with 3D LIDAR.

The IMM-UKF-JPDA consists of four steps: Interaction, Prediction-and-Measurement Validation Step, Data Association-and-Model-Specific Filtering Step, and Mode Probability Update-and-Combination Step. This approach is mainly inspired by work of Schreier, et. al[81] but during the Data Association step JPDAF is used instead of the conventional PDAF since we are considering multi-target.

Remark: some part of the derivation is a rehash of Section 3-3 (Object Tracking as A Filtering Problem), but since the filters introduced are to be coupled, they are to be presented again for coherency reasons.

Throughout the derivation we use, j filter each with different motion model, given by a state-space model representing the state vector $\mathbf{x}_k$ with input vector $\mathbf{u}_k$ at time step k:

$$\begin{aligned} \mathbf{x}_{k+1} &= f_j(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_{j,k} \\ \mathbf{z}_k &= h_j(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_{j,k} \end{aligned} \tag{4-2}$$

The process noise $\mathbf{w}_{j,k}$ and measurement noise $\mathbf{v}_{j,k}$ are assumed to be zero-mean, white, Gaussian noise sequences and are mutually independent with covariance matrices $\mathbf{Q}_{j,k}$ and $\mathbf{R}_{j,k}$. The mode switching between modes is assumed to be a Markov process[132] and transition from IMM model index i to index j is given denoted by p_{ij}. and the transition is governed by transition probability matrix Π:

$$\Pi = \begin{bmatrix} p_{1,1} & \cdots & p_{r,1} \\ \vdots & \ddots & \vdots \\ p_{1,r} & \cdots & p_{r,r} \end{bmatrix} \tag{4-3}$$

In the MOT implementation, we used the motion model discussed in Appendix D and the choice of filter parameter (e.g. $\mathbf{Q}_{j,k}$ and $\mathbf{R}_{j,k}$) is to be discussed in a dedicated subsection.

Interaction Step

This step utilises the IMM probability mixing, The conditional mode probabilities $\mu_{(i|j),k-1}$ correspond to probabilities that mode i has been established in the previous cycle given that

mode j is active in current time step is given by:

$$\mu_{(i|j),k-1} = \frac{p_{ij}\mu_{i,k}}{\mu^-_{j,k}} \tag{4-4}$$

where $\mu^-_{j,k} = (\mu^-_{1,k}, \ldots, \mu^-_{r,k})^T$ is the a priori mode probabilities at the current time step, which in turn is given by prediction of the mode probabilities of the last time step p_{ij}, which denotes the probability that a mode transition occurs from model i to model j:

$$\mu^-_{j,k} = \Sigma^r_i = p_{i,j}\mu_{i,k} \tag{4-5}$$

Then, assuming we have j individual filters, combined initial state(s) $\hat{\mathbf{x}}_{j,k-1}$ and covariance $\mathbf{P}_{j,k-1}$ are mixed from each filter j to form single initial state $\hat{\mathbf{x}}^*_{j,k-1}$ and covariance $\mathbf{P}^*_{j,k-1}$. They are given by calculating

$$\begin{aligned}
\hat{\mathbf{x}}^*_{j,k-1} &= \sum_{i=1}^{r} \mu_{(i|j),k-1}\hat{\mathbf{x}}_{i,k-1} \\
\mathbf{P}^*_{j,k-1} &= \sum_{i=1}^{r} \mu_{(i|j),k-1}\hat{\mathbf{x}}_{i,k-1}[\mathbf{P}_{i,k-1} + (\hat{\mathbf{x}}_{j,k-1} - \hat{\mathbf{x}}^*_{j,k-1})(\hat{\mathbf{x}}_{j,k-1} - \hat{\mathbf{x}}^*_{j,k-1})^T]
\end{aligned} \tag{4-6}$$

This results in the initial states of each filter of index j computed in previous time step.

Prediction and Measurement Validation Step

The prediction part is being done by UKF, the sigma points are propagated through system function f and the weighted sigma points are to be recombined to the predicted state and its corresponding covariance. The $\mathbf{P}^-_{k|k-1}$ value is subsequently used to choose new sigma points to be propagated into measurement function h. Finally, the weighted recombination of the sigma points is used to produce the covariance matrix $\mathbf{z}_k$ and the predicted measurement $\hat{\mathbf{z}}^-_k$. and $\hat{\mathbf{z}}^-_k$ can be directly used to formulate the validation gate. The steps are given as follows: First, choose a sigma points such that:

$$\begin{aligned}
\chi^0_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} \\
\chi^i_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_i, \qquad i = 1, \ldots, L \\
\chi^i_{k-1|k-1} &= \mathbf{x}^*_{k-1|k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_{i-L}, \qquad i = L+1, \ldots, 2L
\end{aligned} \tag{4-7}$$

where $\left(\sqrt{(L+\lambda)\mathbf{P}^*_{k-1|k-1}}\right)_i$ is the "i"th column of the matrix square root of $(L+\lambda)\mathbf{P}^*_{k-1|k-1}$. The sigma points are then propagated through the transition function f.

$$\chi^{*,i}_{k|k-1} = f(\chi^i_{k-1|k-1}) \quad i = 0, \ldots, 2L \text{ where } f\ R^L \rightarrow R^{|\mathbf{x}|}. \tag{4-8}$$

The weighted sigma points are recombined to produce the predicted state and covariance.

$$\begin{aligned}
\hat{\mathbf{x}}^-_{k|k-1} &= \sum_{i=0}^{2L} W^i_s \chi^{*,i}_{k|k-1} \\
\mathbf{P}^-_{k|k-1} &= \sum_{i=0}^{2L} W^i_c\ [\chi^{*,i}_{k|k-1} - \hat{\mathbf{x}}^-_{k|k-1}][\chi^{*,i}_{k|k-1} - \hat{\mathbf{x}}^-_{k|k-1}]^{\mathrm{T}} + Q_{k-1|k-1}
\end{aligned} \tag{4-9}$$

where the weights for the state and covariance are given by

$$\begin{aligned} W_s^0 &= \frac{\lambda}{L+\lambda} \\ W_c^0 &= \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) \\ W_s^i &= W_c^i = \frac{1}{2(L+\lambda)} \\ \lambda &= \alpha^2(L+\kappa) - L \end{aligned} \tag{4-10}$$

where α and κ control the spread of the sigma points. β is related to the distribution of x. Normal values are $\alpha = 10^{-3}$, $\kappa = 0$ and $\beta = 2$. If the true distribution of x is Gaussian, $\beta = 2$ is optimal

Next, in the update step. Similar to the prediction step, a set of $2L+1$ sigma points is to be derived where L is the dimension of the augmented state.

$$\begin{aligned} \chi_{k|k-1}^0 &= \mathbf{x}_{k|k-1}^- \\ \chi_{k|k-1}^i &= \mathbf{x}_{k|k-1}^- + \left(\sqrt{(L+\lambda)\mathbf{P}_{k|k-1}^-}\right)_i, \qquad i = 1,\dots,L \\ \chi_{k|k-1}^i &= \mathbf{x}_{k|k-1}^- - \left(\sqrt{(L+\lambda)\mathbf{P}_{k|k-1}^-}\right)_{i-L}, \qquad i = L+1,\dots,2L \end{aligned} \tag{4-11}$$

The sigma points are then projected through the observation function h

$$\mathbf{Z}_k^i = h(\chi_{k|k-1}^i) \quad i = 0,\dots,2L \tag{4-12}$$

The weighted sigma points are recombined to produce the predicted measurement and predicted measurement covariance.

$$\hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} W_s^i \mathbf{Z}_k^i \tag{4-13}$$

$$\mathbf{S}_k = \sum_{i=0}^{2L} W_c^i \, [\mathbf{Z}_k^i - \hat{\mathbf{z}}_k^-][\mathbf{Z}_k^i - \hat{\mathbf{z}}_k^-]^{\mathrm{T}} + \mathbf{R}_{k|k-1} \tag{4-14}$$

The state-measurement cross-covariance matrix becomes

$$\mathbf{C}_{\mathbf{x}_k z_k} = \sum_{i=0}^{2L} W_c^i \, [\chi_{k|k-1}^i - \hat{\mathbf{x}}_{k|k-1}^-][\mathbf{Z}_k^i - \hat{\mathbf{z}}_k^-]^{\mathrm{T}} \tag{4-15}$$

is used to compute the UKF Kalman gain.

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k z_k} \mathbf{P}_{z_k z_k}^{-1} \tag{4-16}$$

A measurement is considered valid if it lies inside a validation gate in (3-34), with the addition that we have to incorporate predicted measurement states $\hat{z}_{j,k}^-$ and covariance matrix $\mathbf{S}_{j,k}$ for each filter j.

Data Association and Model-Specific Filtering Step

The data association step and filtering/state estimation step are done largely the same with conventional PDA filter (i.e. as explained in subsection 3-4-3 - 3-4-4) but here we need to incorporate each model in the process, i.e. $\hat{\mathbf{x}}_{j,k}^{-}$ and $\mathbf{P}_{j,k}^{-}$.

The updated state is simply the predicted state plus the innovation weighted by the Kalman gain; however, the associated innovation term is multiplied with association probabilities β.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(v(k)) \\ \text{with } \mathbf{v}_k &= \sum_{m=1}^{N_v} \beta_{m,k}\mathbf{z}_k - \hat{\mathbf{z}}_k^{-} \end{aligned} \tag{4-17}$$

and the updated covariance

$$P_k = \beta_0 \mathbf{P}_k^{-} + (1-\beta_{0,k})(\mathbf{P}_k^{-} - \mathbf{K}_k\mathbf{S}_k K_k^T) + \mathbf{K}_k(\sum_{m=1}^{N_v} \beta_{m,k}\mathbf{v}_{m,k}v_{m,k}^T - \mathbf{v}_k\mathbf{v}_k^T)K_k^T \tag{4-18}$$

Model Probability Update and Combination Step

The mode probabilities are updated based on the likelihood the measurement fit to a model, given as

$$\mu_{j,k} = \frac{\mu_{i,k}^{-}\lambda_{i,k}}{\sum bfu_{i=1}^{r}\mu_{i,k}^{-}\lambda_{i,k}} \tag{4-19}$$

where the Gaussian mixture likelihood $\lambda_{i,k}$ is given as:

$$\lambda_{i,k} = \frac{1-(P_D P_G)}{(V_K)^{N_v}} + \frac{P_D V_k^{1-N_v}}{N_v\sqrt{|2\pi\mathbf{S}_{j,k}|}} \sum_{m=1}^{N_v} e^{\frac{1}{2}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})^T\mathbf{S}^{-1}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})} \tag{4-20}$$

Finally, each filter updated states are to be combined again into single final state and covariance estimate:

$$\begin{aligned} \hat{\mathbf{x}} &= \sum_{j=1}^{r} \mu_{j,k}\hat{\mathbf{x}}_{j,k} \\ \mathbf{P}_k &= \sum_{j=1}^{r} \mu_{j,k}[\mathbf{P}_{j,k} + (\hat{\mathbf{x}}_{j,k} - \hat{\mathbf{x}}_{j,k})(\hat{\mathbf{x}}_{j,k} - \hat{\mathbf{x}}_k)^T] \end{aligned} \tag{4-21}$$

The IMM-UK-JPDA iteration is summarized in Figure 4-12

The filter parameters are tuned by investigating the system dynamics and the predicted noise characteristics in the context of target object movement in the urban scenario, Filter parameters used in this work implementation is explained in-depth and listed in Appendix C.

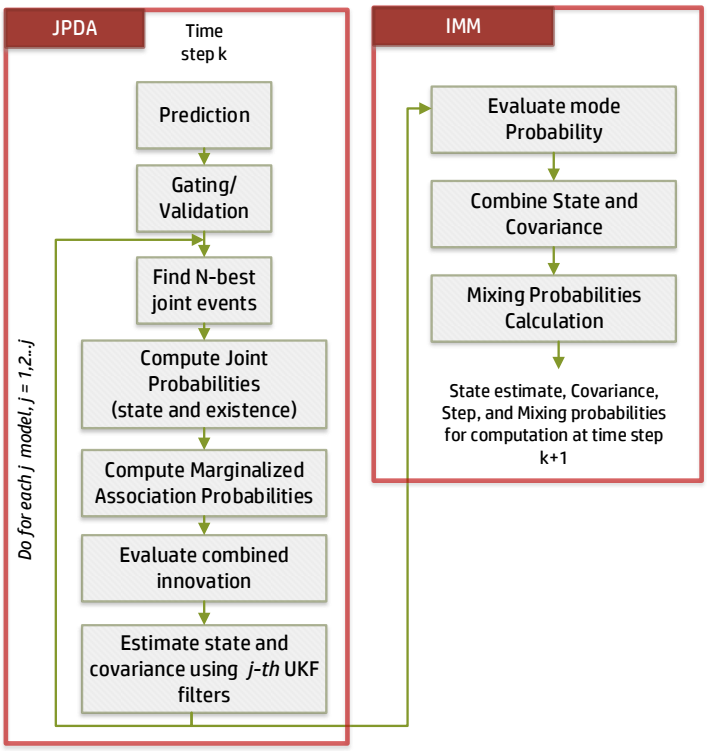


Figure 4-11: IMM-UK-JPDA flowcharts. It can be seen that IMM performs the filtering and JPDA performs the correlation. The JPDA correlation module fits into the IMM model specific filtering block, and the input/output schematics would be identical with Figure 3-6

4-6-2 Track Management

A large number of tracked objects with the associated uncertainties calls for efficient implementation of Track Management. The main purpose of track management is to dynamically limit the number of spurious track list (thus preventing false data association) and maintaining object tracking in case of missed detection. Optionally, track management can also be used for semantic classification purpose based on the track attributes (in our case static-dynamics behaviour and tracks maturity as shown in Figure 4-12). In this thesis implementation, the track management is aimed to meet the above two objectives.

Track States and Maturity

First, to distinguish the track status, each and every track is assigned a unique finite 'state' (not to be confused with the Kalman Filter system states) to reflect its validity and maturity status of the track. The states are summarized in Table 4-2.

The state numbers also act as a scoring threshold. A track that has successful measurement association will be assigned state = 1 (`Initializing`) at time frame k, and if at time frame $k+1$ another successful association occurs, the state number is incremented by 1 until the state number has reached number of 3. After that, in the next time frame the track state will be upgraded to `Tracking` (state number 5).

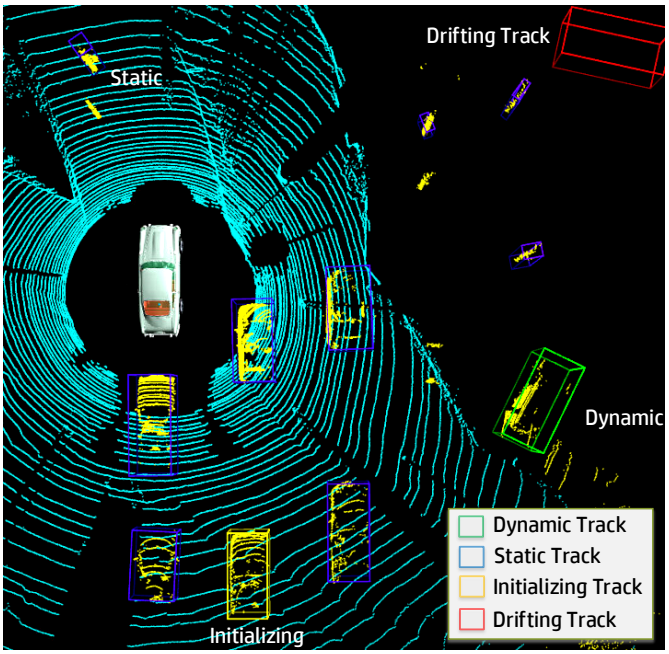


Figure 4-12: Track Management: color-coded classification of Track Maturity and Static/Dynamic behaviour. Dynamic track (green) indicates the object is moving, Static track (blue) indicates the object is stopping together with ego-vehicle, Initializing track (yellow) indicates the track not yet matured since the object has just entered the sensor frame, and Drifting track (red) indicates the track is about to be lost due the track entering blind spot area

Table 4-2: Track states description

State	Name	Description
0	`Invalid`	Uninitialised track, or track with invalid measurement or out of range measurement
1-3	`Initializing`	Track with newly associated measurement
5	`Tracking`	Track with associated measurement that passes gating for more than n time frame (here n = 3)
7-10	`Drifting`	Track with lost measurement may return to `Tracking` state if measurement is found again in the future

The same logic is applied when a track lost its measurement post `Tracking` state, when a track with `Tracking` status at time frame k lose its measurement at $k+1$, the state number will be incremented by 1 and the track enters `Drifting` mode. If at next time frame a measurement is found, the track state number would be decremented by 1 until eventually it regains the `Tracking` status. Alternatively, if no measurements are associated until the track exceeds state number of 10, the track status will be set to `Invalid`. A track with the status of `Tracking` will retain its state number indefinitely if a valid measurement is always found. The state is still being filtered as long as the filter state is not invalid, this means lost track can be recovered because the track is moving "together" with the lost object until it reappears at the predicted

location.

Track Initialization and Re-distribution

A track will start when a statistically nearby measurements are found. In order to enable this, the tracker(s) has to be placed in position nearby to measurement in the first place. For first time step, every track is distributed evenly spaced on the tracking region (defined as the adjustable detection range in the Detector component). After the measurement has been received on the next time step, the tracks are relocated to last known measurement point which has not been associated with the existing track. If all measurements have been associated with existing track, the rest of uninitialized tracks are to be relocated randomly in the tracking region.

Track Pruning

One of the undesirable traits of JPDA is its tendency to coalesce[133] when neighbouring tracks share the same measurement. In order to prevent a duplicate track associated with the same objects, a track pruning mechanism is implemented based on (1) track history and (2) Euclidean distance of the neighbouring track.

First, the last n track Kalman Filter states of each track are stored in history. Then the difference between the states history value of a track toward all other tracks is computed. If the cumulative sum of the standard deviation is less than a predefined threshold (history gating level) then a track is considered duplicate. Finally, the track that has the shorter lifetime is deleted (i.e. to preserve track continuity). The second approach computes the Euclidean distance of each track toward each other, if the distance is less than physically possible threshold distance between two moving traffic object, then the newer track will be deleted.

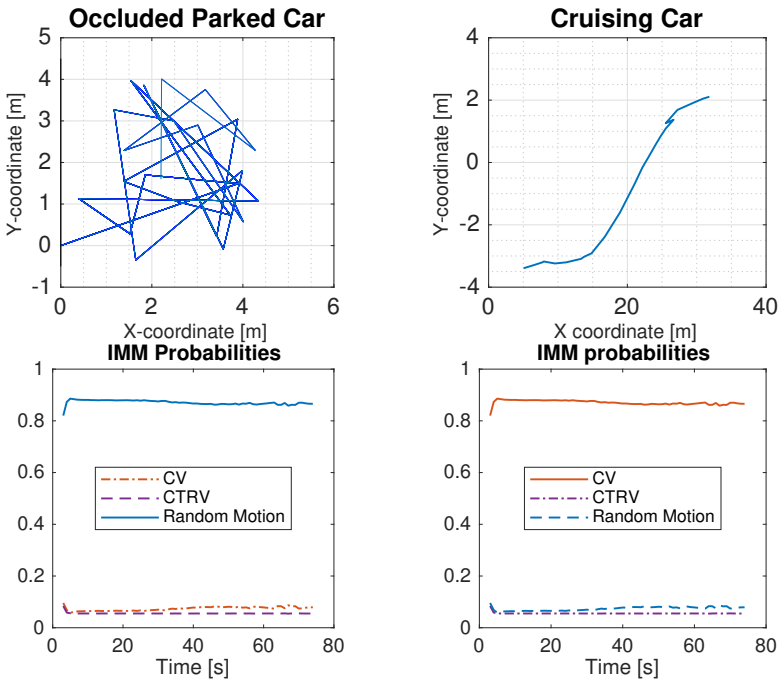
The pruning procedure is summarized in Algorithm 3 (Appendix B).

Static/Dynamic Classifier

A static object is simply an object that does not move relative to ego-vehicle, that is zero relative velocity. However, the presence of noise, occlusion and vehicle constant frame change makes classifying the static object to be non-trivial. on the other hand, the use of IMM filters allows us to some degree to distinguish between static and dynamic objects: static and noisy object will generally have higher probability to evolve under random motion model (which inherently is a stationary model set with large process covariance).

Therefore, this thesis proposes the use combined use of velocity and IMM probabilities informations (similar to[36]) as criteria to classify statics and dynamic objects. Generally, an object with zero or negligible relative velocity would be categorized as static. Furthermore, additional checking is done on the IMM probabilities of the three motion models, if the IMM predicts that the probability that the object is moving under Mode-3 (Random Motion) is higher than the other 2 models, then it is statistically likely that the object is a static object (cf. Figure 4-12).

Naturally, we have to take into account that the estimated velocity is not necessarily smooth, and the IMM probability prediction may take some time steps to converge. To tackle these shortcomings, an average past velocity is used as criteria for static/dynamic classification, additionally a checking if the filter has converged to steady-state value is done before using the IMM probabilities estimation as the basis of classifying an object as static. The procedure is highlighted in Algorithm 4 (Appendix B) and quantitative evaluation result can be found in Appendix A-2.



(a) Occluded parked follows RM model **(b)** Cruising follows CV model.

Figure 4-13: Dynamic vs static Object prediction using IMM

4-7 Bounding Box Tracker

The IMM-UKF-JPDA filter provides estimated poses for point-based object, in order to incorporate dimensional and heading information (i.e. that of bounding box) a logic-based algorithm is implemented which essentially has three major functionalities (1) Associate detected box with a track (2) to retain detector best-known output and if later a better detection output is found, update the retained bounding box parameter, (3) perform geometrical correction to compensate occlusion and ego-car perspective change.

It is also useful to revisit the rationale behind non-use of the extended object tracking. Aside from computational constraints, we have learned that the nature of LIDAR sensor is prone to self and external occlusion and thus some over-segmentation is likely to be unavoidable. With the existing implementation of object detection, it is reasonable to anticipate that the increase in computational complexity may not necessarily translate into a more accurate estimate of the object shape states.

The logical rules are derived from the range and view-dependant occlusion characteristic of the LIDAR sensor measurement. Additionally, the algorithm also utilises track maturity information from the tracker components to distinguish noise from the real object. The rule shall be elaborated in the following subsections.

Finally, to handle over-segmentation, we employ a so-called bottom-up, top-down approach is an object tracking architecture as proposed by Himmelsbach[25] in which tracker component works together with detector component in a feedback-loop-like fashion. The use of rule-based box updates logic allows efficient and effective false detection reduction without sacrificing performance.

4-7-1 Bounding Box Association

The nature of JPDA tracker is that multiple measurements are to be associated with the tracker in order to augment the update step during filtering. Notwithstanding, the bounding box has to be mapped exclusively to a track since bounding box dimensions are not among the filtered states, and the object tracker is expected to give a single best estimate of bounding box object to the higher level perception module.

The box tracking association is to be run based on pseudo-code in algorithm 6 (Appendix B). To summarize, here the association only occurs when the IMM-UKF-JPDA track has met the threshold of maturity and past a predefined threshold of track lifetime. This approach prevents a box belonging to clutter to be associated with a valid track, as mature track indicates that the generated box falls within gating area and statistically likely (based on the Mahalanobis distance) to be the right measurement.

For a track associated with only single measurement, the one-to-one mapping condition is fulfilled, and thus no further check is required. on the other hand, For a track associated with multiple measurements, the box with the closest Euclidean distance is preferred over the ones that are further, and the track's box association is marked as ambiguous.

Gating process based on adjustable distance threshold is also done to prevent impossible association. The association criterion is purposely made to be simplistic, since the measurement has actually passed more rigorous gating by the IMM-UKF-JPDA position tracker. Therefore, although a false association is possible, the likelihood such association coming from a mature track that has been cruising over certain time threshold is expected to be low.

4-7-2 Bounding Box Dimension and Heading

The bounding box is defined by 4 basic parameters: width, length, height and yaw heading. As has been elaborated in the preceding detector section, the box parameters are generated using minimum area rectangle algorithm based on the clustered points and then corrected using L-shape detection for the box-like object.

While this approach almost guaranteed to be correctly working for a *complete* measurement of an object (i.e. all part of object is observable), such measurements themselves can only be obtained when an object falls within a certain range of the sensor and is free from another occlusion. In other words, as the object goes closer to ego vehicle, more part of a detected object is expected to be seen by LIDAR sensor, and the non-observable part would be reduced.

More importantly, when an occlusion occurs and no sufficiently distinguishable edge line is found in the cluster, the L-shape detection cannot be used and the yaw heading estimates will be erroneous.

Remark: It is also important to highlight that although the motion model used by Kalman filter in the position tracker incorporate yaw and yaw rate as estimated states, it assumes an infinitely small object (i.e. point). This means the yaw estimate is not directly applicable to bounding box heading, since road object does *not* necessarily travel toward its yaw heading. For example, a vehicle moving at relatively high speed would introduce side-slip angle[134]. However, a significant discrepancy between yaw heading and bounding box heading is caused by the constantly moving sensor reference, when the ego-vehicle rotate, the surrounding targets would be seen as manoeuvring by the sensor, and the yaw estimate would follow suit. However, the target object itself does not have a change in orientation. An even simpler example can be found on a parked object: a car parked diagonally should not have its bounding box heading set to its yaw estimate. To summarize; yaw estimate heading is distinct with bounding box heading and the bounding box should reflect detection results.

The evolution of the bounding box is illustrated in Figure 4-14: as the time step increases and "Detected object 1 and 2" are coming into proximity of ego vehicle, the bounding box grows larger. Likewise, as the detector object goes away from the effective range of the LIDAR sensor, the bounding box grows smaller. The wrong yaw estimate can be observed on bounding box fitting of "Detected Object 2" at time step k, as there is no L-shape to fit the box, the resulting bounding box has an incorrect heading. Finally, another scenario in which the bounding box would go smaller is when occlusion and/or over-segmentation occurs. "Detected Object 3" is split into 2 boxes due to tree-like "Occluding Object".

In order to handle the constantly changing dimension and yaw heading of the detected bounding box, several rules are introduced based on heuristics. These rules govern the update routine of every bounding box for each and every time step:

1. **Bounding box area generally should not shrink** - based on shrinkage behaviour observed in Figure 4-14 we can further surmise that bounding box length is the parameter that chiefly changes in most case of self-occlusion.
2. **Bounding box should never rotate too fast** - traffic objects are generally non-holonomic, it is physically improbable for car-like (without loss of generality) object to change yaw heading significantly between sensor sampling time (100 ms) on a normal road.
3. **Bounding box should not have sudden movement in the opposite direction against previous time-steps travel heading** - when occlusion and shrinkage occurs, the centre point of the box will be shifted incorrectly (as seen in Figure 4-14). This shifting results in an unnatural backward movement of the detected object.

These three rules basically describe the noisy behaviour of detector output, if the rules above are violated, we can likely consider the bounding box change as noise.

Conveniently, bounding box dimension does not get larger than the real object except on the rare case of under-segmentation or (later-to-be discussed) possible failure case of over-segmentation handling. The same can be said for the sudden opposite movement, as traffic

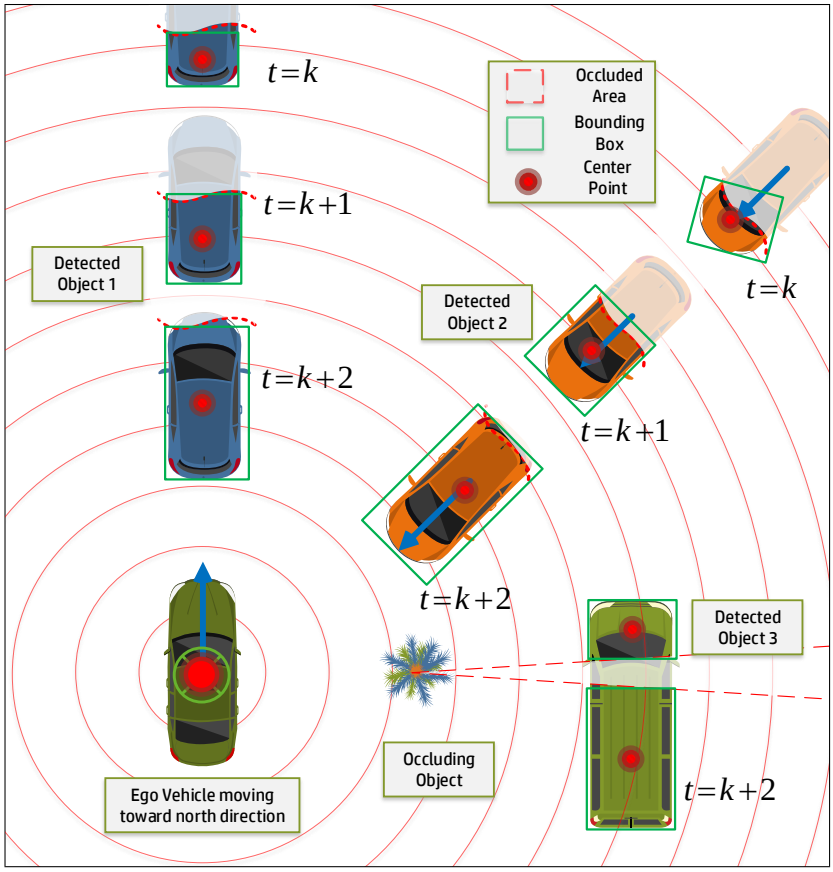


Figure 4-14: Effect of occlusion on bounding box fitting.

object does not simply jump back within 100 ms time frame. on the other hand, there is no clear-cut threshold to distinguish real yaw change due to object manoeuvre and yaw change due to detection noise, so only large erroneous yaw change is guaranteed to be caught by the rule-based filter.

Generally speaking, the condition is one more rationale that makes a rule-based bounding box tracker to be more suited rather than extended object tracking; both try to address the uncertainty of object dimension, but a rule-based filter would be computationally cheaper and sufficient for such relatively simple uncertainty.

In order to realize the rule-based algorithm, this thesis proposed the use of per-frame history and information from to IMM-UKF-JPDA tracker to augment the Box Tracker. Per-frame history enables the Box Tracker to store last-known good bounding box at time step $k-1$ and compare it to the newly generated bounding box at time step k. The so-called first last-known bounding box is obtained during bounding box association process described in preceding subsection.

If the newly generated bounding box at k, when compared with that of $k-1$ passed the rule-based filter, then do necessary change: update the dimension, or update the yaw, or do both. Otherwise, retain the last known box if no sane dimension and yaw update is detected. This rule can be seen in Algorithm 6 (Appendix B).

This algorithm works for the common scenario where occlusion does not persist, in the case when the occlusion persists (e.g. when an object is moving further away from ego car and maintain a similar relative velocity afterwards) the yaw change will not be updated correctly. in order to address this, box information from few steps back $k-n$ (n is adjustable look-back steps parameter) is inspected to discern if box has not been updated for n steps back, which indicates that dimension and/or yaw of the box would need to be updated despite the changes is out of threshold. Finally, a velocity information from the position tracker is used to rule out change of yaw and reduction of dimension for the static object (i.e. zero relative velocity). The main advantage of the introduced logic-based box update is the computation load would be negligible. However, since the threshold is based on heuristics knowledge, the optimal threshold value can only be obtained through tuning and in this case, sensor specific.

4-7-3 Perspective Correction for Self-occlusion

Detected object self-occlusion depends on the perspective of ego-vehicle (see Figure 4-15). This phenomenon results in inconsistencies in the placement of box centre point. Since the position tracker is only aware of box centre point, when self-occlusion occurs centre point correction need to be applied to the tracker box using information of last-best known box dimension.

Naturally, since the tracking is done on ego-vehicle navigation frame (centre point x, y), a trigonometric transformation is done to translate the shift in occluded bounding box from the local frame (centre point x', y') to global frame.

The centre point shifting is illustrated in Figure 4-16, and the mathematical relationship between the newly shifted box centre point (x_{shift}, y_{shift}) and the original occluded box centre point x_{orig}, y_{orig}) is given as:

$$\begin{bmatrix} x_{shift} \\ y_{shift} \end{bmatrix} = \begin{bmatrix} x_{orig} \\ y_{orig} \end{bmatrix} + \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} \Delta_x{}' \\ \Delta_y{}' \end{bmatrix} \tag{4-22}$$

where the centre point shift Δ_x' and Δ_y' is determined by the geometrical difference (length s, width w and centre points (x, y)) between that of the tracker box and that of original (occluded) box, given as:

$$\begin{aligned} \Delta_x{}' &= (TrackBox_x \pm \frac{1}{2} TrackBox_l) - (OrigBox_x \pm \frac{1}{2} OrigBox_l) \\ \Delta_y{}' &= TrackBox_y \pm \frac{1}{2} TrackBox_w - (OrigBox_y \pm \frac{1}{2} OrigBox_w) \\ &\quad \mp \frac{1}{2}(TrackBox_w - OrigBox_w) \end{aligned} \tag{4-23}$$

and the $\pm$ operator depends on the shifting direction of the box, an addition operator is used to shift the box to top edge (for back occlusion), and subtraction operator is used to shift the box to lower edge (for frontal occlusion). The shifting direction itself is determined by the object location in Cartesian coordinate relative to ego vehicle (refer to Figure 4-15).

However, a further check has to be put to prevent box shifting of an imperfect box caused *not* by self-occlusion. This check is made possible by checking the relative velocity in the longitudinal direction, if the object is moving in the same direction as ego-vehicle, it would be

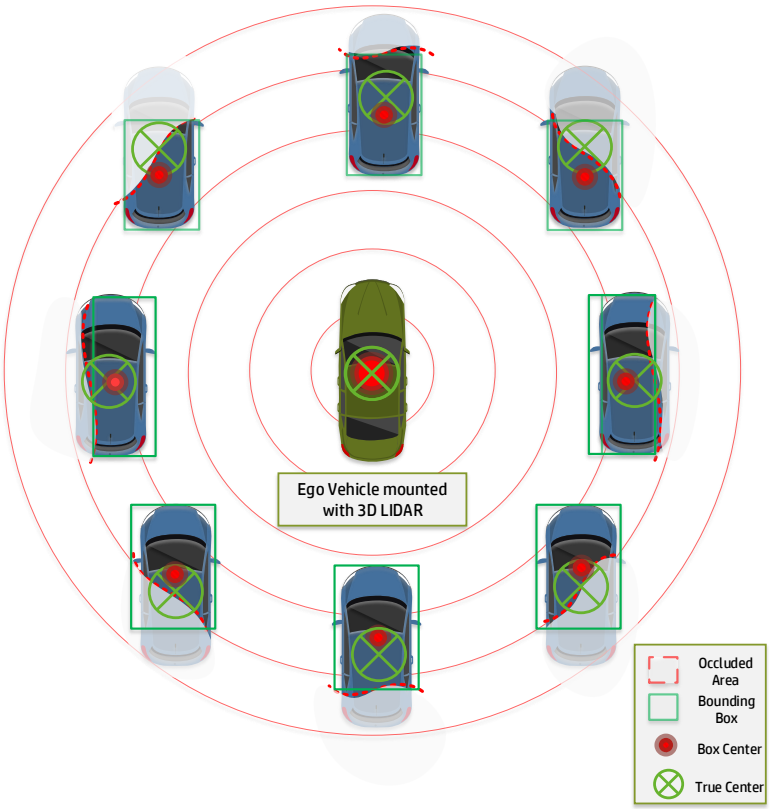


Figure 4-15: Occlusion of detected object on different viewpoint locations. The bounding box centre points become incorrect

incorrect to shift the box downward and vice versa. Naturally, as even static object would have a small non-zero velocity due to noise, an adjustable threshold is utilised in the procedure. The shifting procedure is summarized in Algorithm 8 (Appendix B) and the results can be observed in Figure 4-17.

4-7-4 Over-segmentation Handling

The segmentation process based Connected Component Clustering relies on a well-defined space between cluster. This space can also inadvertently be created in the case of occlusion caused by another object blocking LIDAR sensor direct line-sight (refer to Figure 4-14). The occluded area becomes a blind spot and results in over-segmentation. The tracker component can assist the detector to predict beforehand for such over-segmentation based on the object travel direction and last known good bounding box dimension. Himmelbach[25] notably proposed a similar approach, but instead of dynamically changing the Connected Component Clustering parameter, this work proposes the use of area percentage threshold to avoid repeating the steps of clustering.

The over-segmentation handling is made possible by the following procedure: in time step $k-n$ the full dimensioned box already is stored, in addition to that, the tracker has relative velocity information for all tracked object in each time step. At time step k the detector component

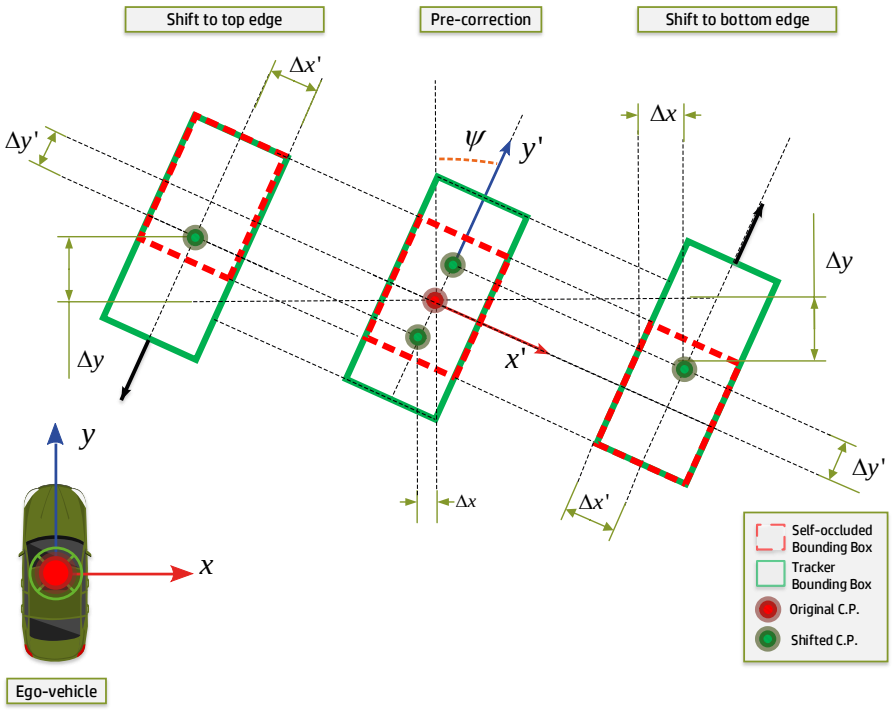


Figure 4-16: Perspective correction for the bounding box.

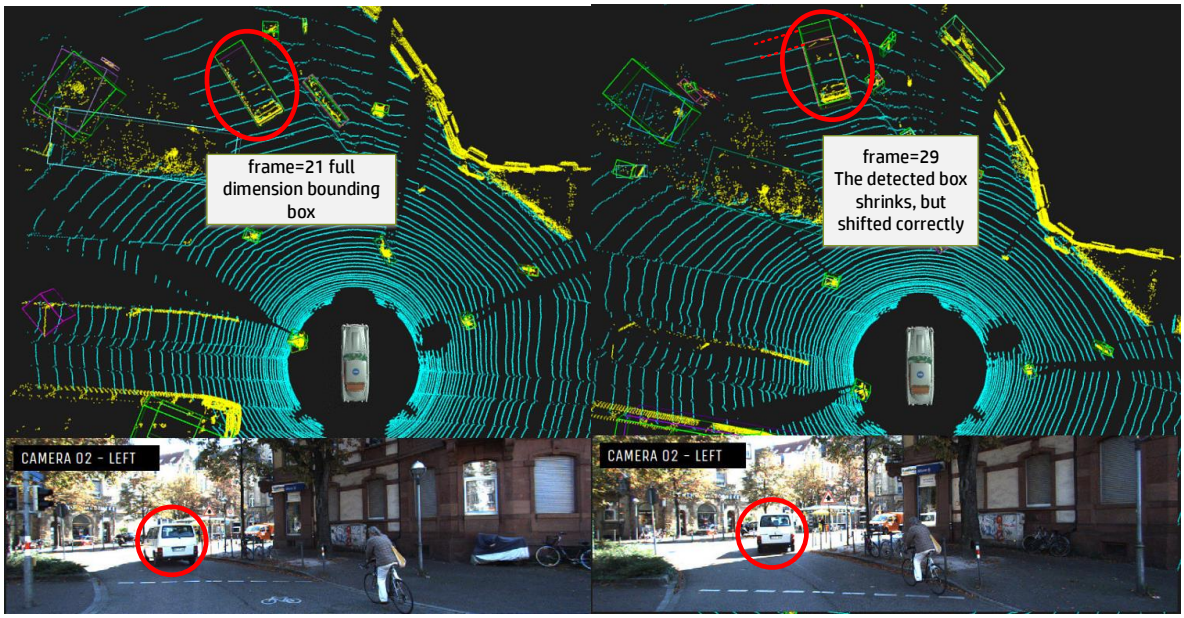


Figure 4-17: Instance of perspective correction. Notice the self-occlusion of a van located in front of ego-vehicle at frame 29, the detected box becomes smaller (occluded length denoted by dashed red-line). The tracker retains full dimension are retained and the bounding box is shifted downward so that centre point remains accurate (KITTI Dataset 0005)

then uses the tracker predicted position *prior* to bounding box fitting, if the predicted box encloses or overlaps significantly with newly generated clusters, then such clusters are to be merged. This procedure, however may induce under-segmentation: two correctly clustered objects located in proximity may be incorrectly merged to become a larger cluster. To address this, the dimension of potential merged cluster is made sure to not exceed that of predicted box before the merging is finalized. The procedure is depicted in Figure 4-18 and formalized in Algorithm 8 (Appendix B). One instance where the procedure is applied in KITTI Dataset (Dataset 0009) can be seen in Figure 4-19.

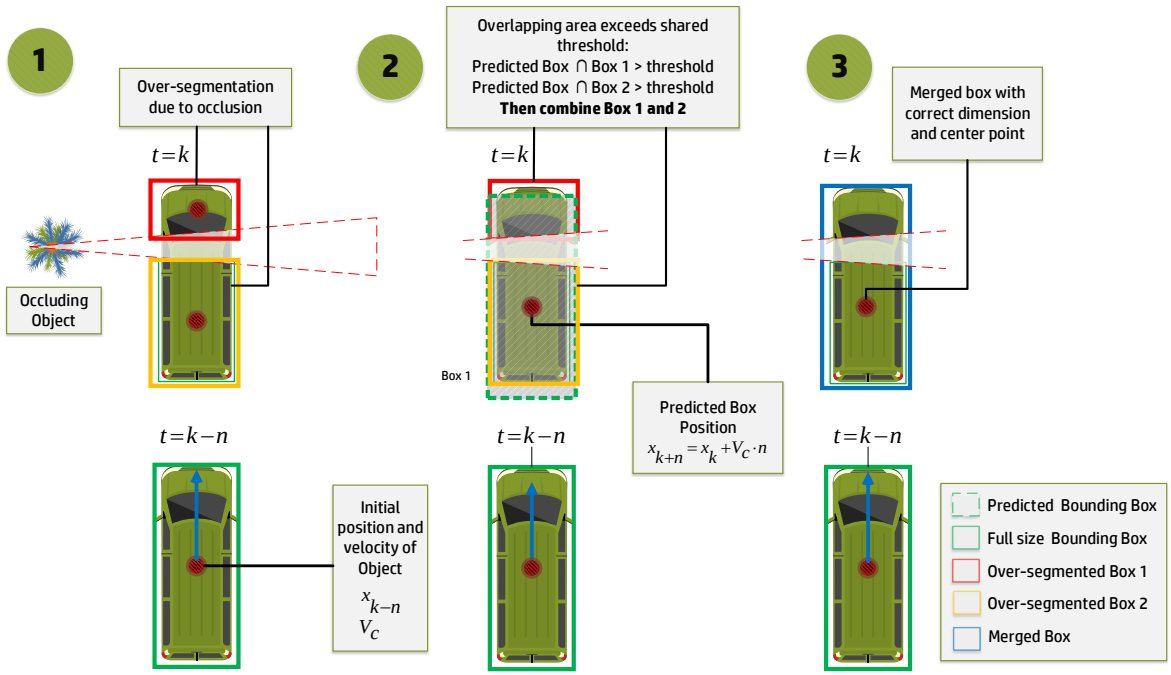


Figure 4-18: Over-segmentation handling

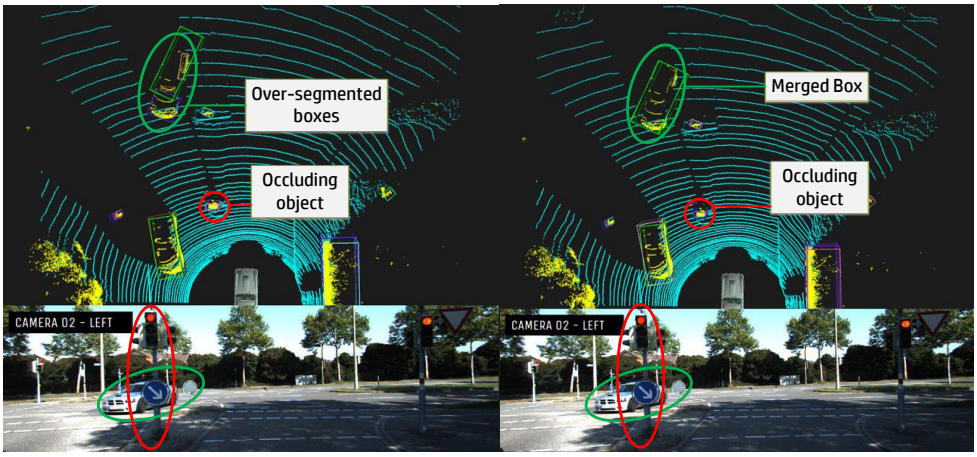


Figure 4-19: Example of over-segmentation handled by box merging

Chapter 5

Evaluation

5-1 Overview

In this chapter, the evaluation of the designed MOT system will be presented. A benchmark using established metrics and real data representing complex urban situation is to be carried out. The following discussion shall elaborate the underlying reason behind the evaluation results in term of tracker implementation, environment and tracking targets dynamics.

Remark: a complementary evaluation can be found in Appendix A, which covers individual analysis of sampled track, real-time capability assessment and investigate the performance of static and dynamic classification.

5-2 Evaluation Metrics

The tracker performance is to be evaluated using MOT16 benchmark method proposed by Milan et. al[70] which combines the CLEAR quantitative metric[135] augmented with sets of Track Quality Measure[136]. Note that this thesis work does not classify the tracked object beyond the dynamic and static nature. Therefore, some specific metrics that require target classification of the tracked object are excluded. The evaluation metrics used will be discussed briefly to better understand the evaluation results, the interested reader may refer to the source literature[70] for more comprehensive elaboration and underlying reasoning.

1. **Tracker-to-target assignment**

   This metric describes the reliability of the Multi-Object Tracker by measuring the number of False Positive (FP) and False Negative (FN) per and across all frames. Tracker robustness is also tested by measuring the number of fragmentation (i.e. losing the track and starting a new one) for a tracked object; this is indicated by the number of tracker ID switch (IDSW) across all frames.

2. **Multiple Object Tracking Accuracy** MOTA combines FP, FN and IDSW to indicates overall performance of the tracker, this metrics is given as:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{5-1}$$

where t is the time step (frame index) and GT is number of Ground Truth. Value of MOTA can also be negative if the number of errors exceeds the number of actual objects, it is maximized at 100.

3. **Multiple Object Tracking Accuracy** MOTP is the averaged differences between True Positive (TP) and Ground Truth (GT) count; it gives the average overlap between all correctly assigned track and the detected object. Essentially, it indicates the local accuracy of the tracker. MOTP is given as:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_i d} \tag{5-2}$$

where c_t denotes the amount of tracker-target match in frame t and $d_{t,i}$ is the bounding box overlap between tracked target i with the GT. The overlap is measured by intersection over union[137]. Top of the bounding box is used for this metrics, since target object in the urban situation does not move vertically and the horizontally mounted 3D LIDAR sensor does not suffer from any significant occlusion in measuring the height.

4. **Track quality measure** .

A classification is made based on how many trajectory of the GT is covered by the tracker. Mostly Tracked (MT) correspond to at least 80% coverage, and Mostly Lost (ML) means that track is only covered for only less than 20%. Another indicator is Fragmentation Number (FM), which is the number of a track interruption before it resumes the previously lost trajectory.

5-3 RAW Data and Ground Truth: KITTI Dataset

In order to verify our MOT system against the real-world situation, it is imperative to use non-synthetic data. For this purpose, the KITTI datasets[73] is used throughout the thesis work. The public dataset provides the recording of LIDAR sensors, among other sensors in a diverse urban driving scenario recorded on the city of Karlsruhe, Germany.

The recording captured real-world traffic situation and range from highways over rural areas to inner-city scenes with high-quality hand-labelled annotation. The sensor used by KITTI team is also Velodyne HDL-64E operating in 10 Hz interval, producing 64 layers of LIDAR data with 0.09 degree angular resolution, and 2 cm distance accuracy. The data stream consists of $\approx$ 1.3 million points/second within 120 m radius of 360 deg horizontal and 26.8 deg vertical field-of-view.

The Velodyne scans are encoded in floating point binaries which are then manipulated in C++ using Point Cloud Library[109] in the tracker implementation (see Section 2-2 and Section 4-3). Each point is stored with its x, y, z coordinate and an additional reflectance value. While

the number of points per scan is not constant, on average each file/frame has a size of $\tilde{1}.9$ MB which corresponds to approx. 1.20×10^5 3D points and reflectance values.

The KITTI authors also provide synchronized camera raw image sequences in addition LIDAR recording. Furthermore, for validation purpose, object labels in the form of *3D tracklets* can be used as ground truth. The tracklets are encoded in XML format and the following information are used during evaluation process:

1. **Object Class**- "Car"/"Van"/"Truck"/"Pedestrian"/"Person"/"Cyclist"/"Tram"/"Misc."
2. **3D Box Size** - length, width, height
3. **Object Translation and Rotation in 3D ego-vehicle frame**: translation of x,y,z in metre and yaw rotation in degree

The KITTI Data also provides a benchmarking platform which shares some of its methodologies with MOT16, however much like MOT16, the benchmark is intended primarily for tracking using camera, with LIDAR data synchronized to the camera image. Therefore, a LIDAR object is only annotated *if and only if* the object is within camera frame. There are naturally numerous frames when the object is visible on camera but completely missed by LIDAR sensor (i.e. out of range). The inverse is also true: the LIDAR is 360 deg surround sensor while the camera is a single view frontal view sensor, therefore objects that are located on the sides and back of the ego vehicle will not have corresponding LIDAR ground truth.

Moreover, the Velodyne range correlates with object reflectivity, with a quoted range of 50 m (10% reflectivity) and 120 m (80% reflectivity) and lower vertical FOV (32 deg)[18]. In practice, the effective range of the sensor is 20-40 m[138, 11]. This is confirmed by independent observation in the Visualizer program that an object formed a discernible shape mostly when it falls within 30 m horizontal range and 20 m vertical range. Beyond this range, the number of points reflected is very low and even to human eyes the object shape is not easily recognized. The tracker is still capable of tracking the position and predict the trajectory, but no useful dimensional information can be derived until such object comes into the sensor effective range.

Since this work deals purely with LIDAR tracking, we have to rely on a comparable ground truth that either provides an identical frame of reference with the sensor across all compared time frame or alternatively, provides an annotated state of occlusion to adjust the evaluation score. In this respect, The KITTI ground truths offer annotated occlusion state in camera frame, but no equivalent is found for the LIDAR frame. In addition, unlike camera-based tracker benchmark, in which the tracking is done on a 2D plane (camera frame), we are tracking in world coordinate, so the dimension of ground truth stays the same across all frame, regardless of actual dimension seen by the sensor. To address this issue, a subset of KITTI ground truth is selected based on the following criteria:

1. the ground truth has to be always visible by both camera and LIDAR sensor across the object lifetime (the real object can still be occluded *after* the first appearance)
2. the ground truth has to fall within Velodyne effective range (30 m horizontal and 20 m vertical)
3. the ground truth that meets the criterion 1 and 2 within a subset of its lifetime is also eligible, and the comparison is made during this subset time frame.

To evaluate the relevant urban situation, KITTI datasets within category "City" are used. The selected collection of datasets consist of 10 different driving scenarios with the cumulative frame number of 2111 frames and 188 unique traffic objects. The composition of each dataset is represented in Table 5-1 and Figure 5-1.

Table 5-1: Evaluation datasets

Dataset	Frame count	Unique objects	Object instances (#Box)
0001	106	11	142
0002	75	2	45
0005	152	14	473
0009	441	82	1413
0013	142	4	101
0017	112	5	84
0018	268	12	196
0048	20	7	81
0051	436	40	381
0057	359	12	471
Sum	**2111**	**189**	**3387**

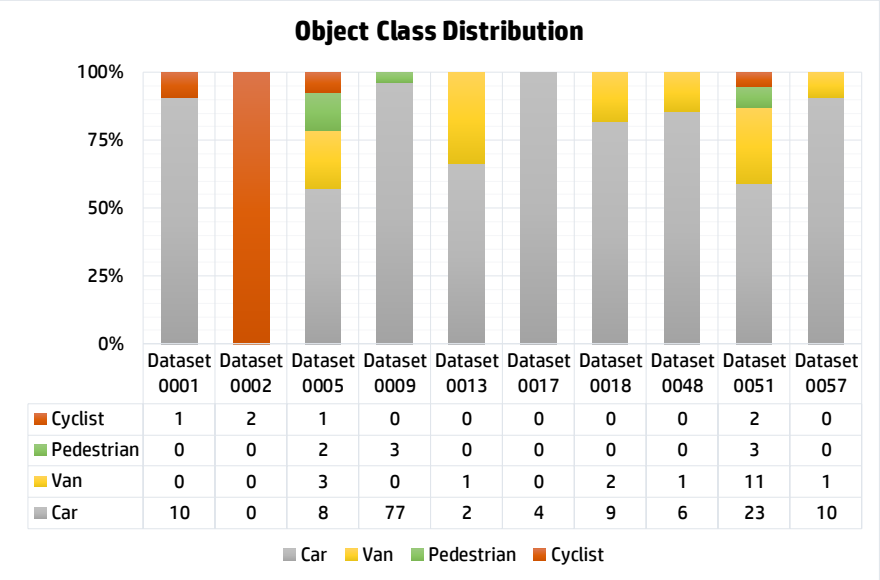


Figure 5-1: Distribution of object class across all evaluation datasets

5-4 Benchmark Results and Discussion

To evaluate the tracking performance quantitatively, the tracking system is run against each dataset with the same filter parameter and rule-based filter threshold for all datasets. The tracking results are then benchmarked against the ground truth, and the benchmark metrics are computed per individual datasets.

It is important to note the datasets are not a uniform: the driving scenario along with the object composition and movement may vary significantly as the datasets change. This is intentional as tracking methods can be heavily over-fitted on one particular dataset and introduced evaluation bias[70]. Therefore, the individual dataset evaluation result is more of an accurate indicator to reflect the MOT performance. Nevertheless, it is still useful to view the overall score as shown in Table 5-2 to aid the reader in grasping the tracker performance.

Table 5-2: Overall evaluation result

(a) CLEAR Metrics

Metrics	Value
MOTA	86.12% ± 6.00
MOTP	91.01% ± 5.03
FP (%)	1.92
FN (%)	11.89
IDSW (sum)	75
Total Obj. Instances	3387
Total Frame	2111

(b) Tracks quality measures

Metrics	Value
Mostly Tracked	70.64% ± 17.47
Mostly Lost	9.33% ± 8.10
Recall rate	88.92% ± 10.18
Precision rate	98.43% ± 2.73
Fragmentation	211

The MOTA scores reflect that the trackers sport a reasonably high degree of accuracy with 86% overall score, the score is lowered chiefly by the number of FN, since the number of FP and IDSW are comparatively low. The MOTP score is capped at 91% which is expected. Since despite perfect tracking, only partial dimensional information can be derived when an object enters the sensor frame from a far distance, and thus, the overlap between tracker generated box and ground truth is *always* low in the first few time steps.

Quality measures-wise, although predictably we see a significant deviation across all datasets, the average results still highlight that the tracker yield higher number of MT than ML. The recall-rate (i.e. sensitivity) and precision (i.e. positive predictive value) indicate that the tracker hypotheses possess a high degree of relevance to the actual object, the lower recall rate is consistent with the number of FN counted in overall. Lastly, the Fragmentation number (FM) is a subset of the FN; here we see the number signals that more than half of FN is caused by track lost that later is able to be successfully resumed, rather than a complete failure of detection across all time frame.

To gain more insight into the tracking performance, we inspect the results which belong to the individual dataset. The MOTP and MOTA scores can be found in Figure 5-2, the quality measures can be inspected in Figure 5-3 and the base metrics score can be seen in Table 5-3,.

The trend is pretty apparent: the scores are generally higher for datasets with less number of object instances; less number of objects in sensor frame would mean there is reduced number

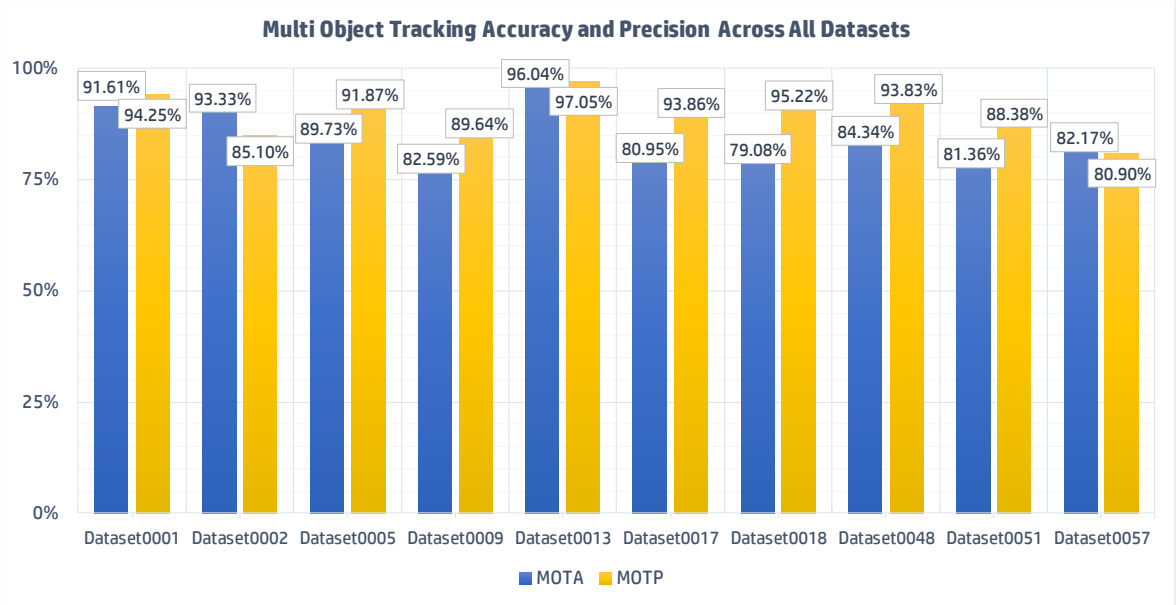


Figure 5-2: Per dataset MOTA and MOTP scores.

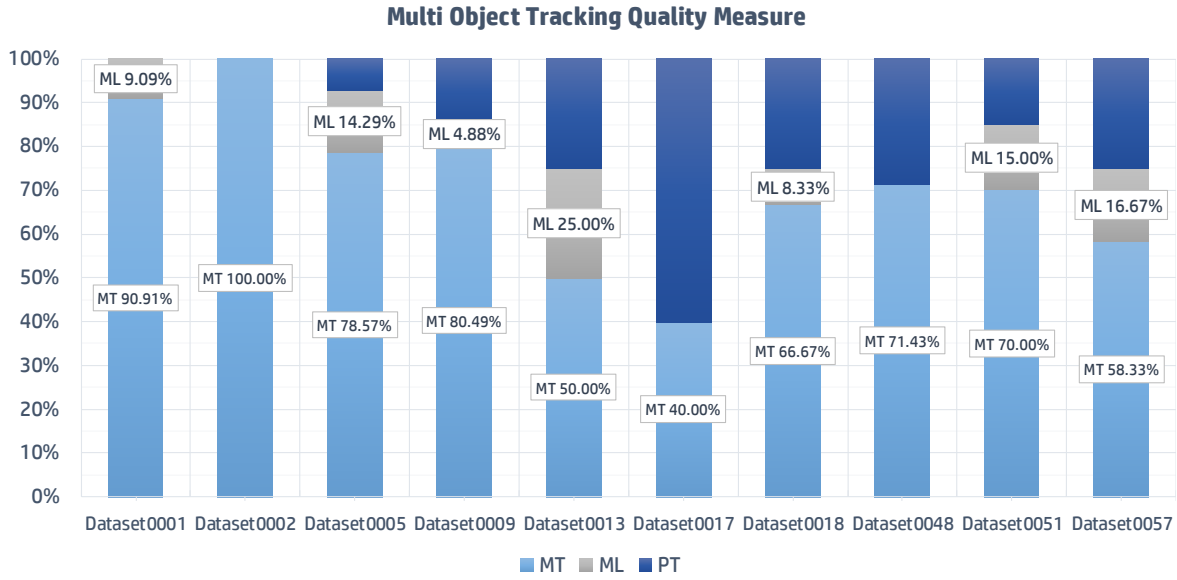


Figure 5-3: Per-dataset Track Quality Measures. Note that "PT" refers to Partially Tracked, that is the track which is not classified as either MT or ML.

clutter and occlusion to interfere with tracking (see Dataset 0001, 0002 and 0013). Still, this kind of scenario is useful for the evaluation of the steady-state performance of tracking (i.e. when the tracked object stays in the sensor frame for an extended duration). Note that the percentage of quality measure has to be seen together with the number of ground truth appearances, in Dataset 0013 for instance, a single loss of target track correspond to 25% of track lost since there are only 4 unique ground truths in total.

In Dataset 0005, while 78% of tracks are considered to be MT, we see a relatively large number

Table 5-3: Per-dataset evaluation results

Dataset	Frame count	Unique objects	FP	FN	IDSW	Recall	Precision	Frag.
0001	106	11	2	8	2	94.41%	98.54%	0
0002	75	2	1	2	0	95.56%	97.73%	0
0005	152	14	7	34	8	92.87%	98.44%	8
0009	441	82	33	172	41	87.83%	97.41%	99
0013	142	4	2	2	0	98.02%	98.02%	0
0017	112	5	0	14	2	83.33%	100.00%	8
0018	268	12	2	37	2	81.12%	98.76%	27
0048	20	7	0	12	1	85.54%	100.00%	3
0051	436	40	5	59	7	84.51%	98.47%	13
0057	359	12	13	66	5	85.99%	96.89%	53

of ML tracks. Here the ego vehicle is moving in a curved urban road and this translates into a constantly changing sensor frame of reference. Combination of self-occlusion (cf. Figure 4-17 in Chapter 4 for visual depiction) due to sensor angle to the targets and fast (relative) turning rate of target objects increases the uncertainties of the target spatial position (i.e. target are moving rapidly). Therefore, we see a reduced tracking accuracies in this situation and some tracks suffer from fragmentation. A more in-depth analysis of individual tracks found in Dataset 0005 can be seen in Appendix A.

Dataset 0009 represents the most complex driving scenario: it has the largest number of unique object compared to other datasets, and also comparatively lengthy frame count. In this dataset, the ego car made a 90-degree turn (sudden change of sensor frame) and stopped at 4-way junction with a known occluding object. The situation can be observed in Figure 4-19 back in Chapter 4.

In addition, there are numerous parked cars which are tightly spaced and the cars on the junction are moving with constant turn rate, varying the degree of self-occlusion that occurs. These challenges reduce the bounding box precision which makes it capped at 89%. The number of FP and FN are also consistent with the bounding box precision loss (not to be confused with precision-rate) along with the number of ID switches and fragmentation.

Nevertheless, handling urban situation uncertainties is the main contribution of this thesis: here we see the MOTA score reflects that the use of probabilistic data association and filter enable the tracker to be able to form hypotheses with sufficient accuracy despite clutter and manoeuvring targets. Degraded performance are found, but since more than 80% of the tracker hypotheses are considered as MT, and only 5% are considered as ML, we can see the detector pose an adequate robustness against persistent and self-occlusion of target object during the occurrence of sensor frame change, turning cars and other occluding objects. Thus, in Dataset 0009 we see the tracker perform as expected in the presence of environmental uncertainties.

MOTA score of Dataset 0017 and 0018 can be seen to be below average by more than 5%; this is mainly caused by a spike of false negative found in both scenario. These datasets are recorded when the ego vehicle is stopped on a 4-way junction (see Figure 5-4) due to traffic light. Here, a static object (traffic light pole) is making a persistent occlusion across all frame counts, and combined with the low reflectance of dark-coloured car, the resulting detected

box is highly fragmented and the tracker cannot reliably associate such detection results to a new track, resulting in numerous false negatives. The over-segmentation handling in the other hand, correctly compensates for the tracking of the light-coloured car, where a track has been successfully started beforehand. Therefore in this particular datasets, the false negative is caused by unavoidable over-segmentation, and the detected bounding box dimension becomes highly unstable contributing to lower MOTP scores.

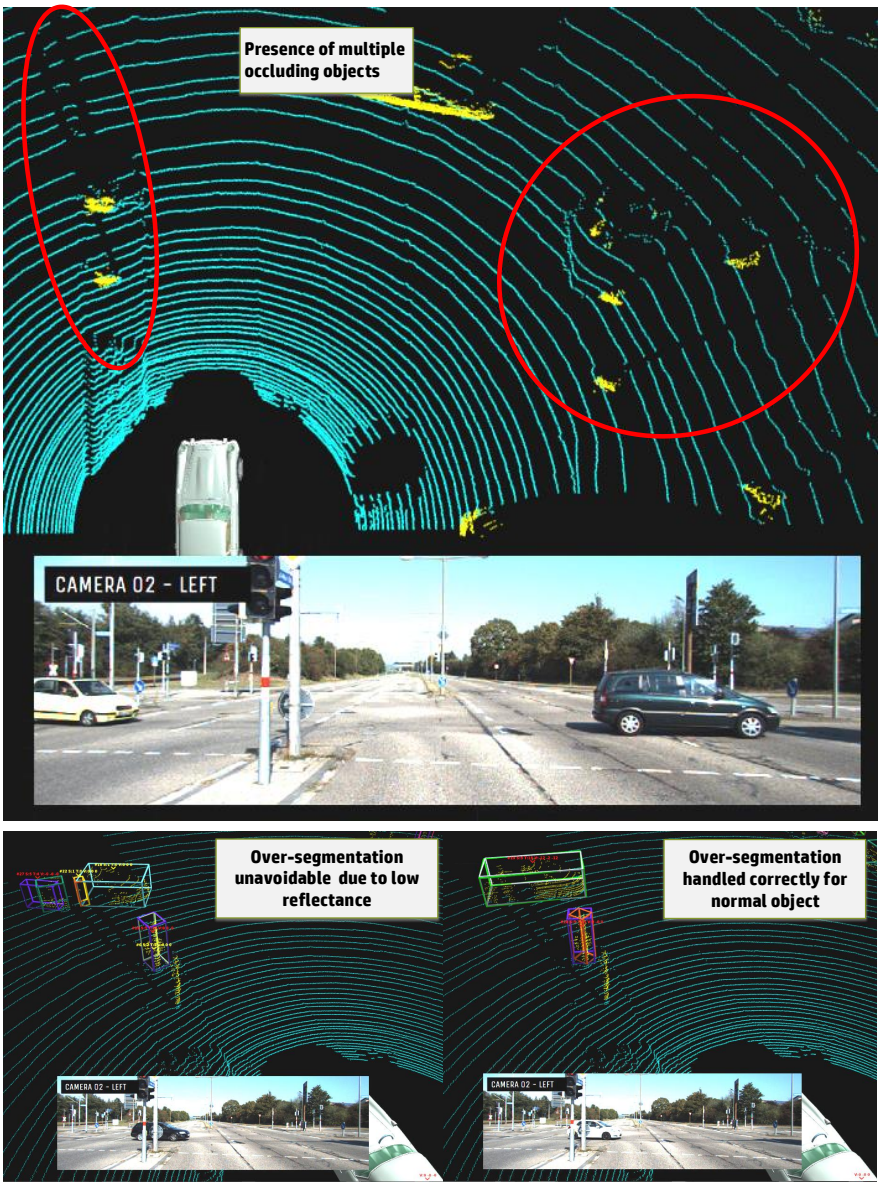


Figure 5-4: False negative due to over-segmentation for an object with low reflectance. Note that the green box and red label indicates track has been started.

Dataset 48 and 51 represent typical urban scenario, busy two-way road with mostly smooth ego vehicle movement, in this case, the tracker works optimally as designed for most of the target objects (all above 80% score for MOTA and MOTP), However in Dataset 0051 we

see larger percentage of ML tracks, this is attributed to some target cars moving in opposite direction with relatively high speed. This speed characteristic is not found in other datasets, while such movement can be tracked by increasing the IMM-UKF-JPDA filter process noise covariance, this would negatively affect the gating and prediction performance in more noisy environment (i.e. risk of more false positive), thus a design choice is made to favour more general case than specific ad-hoc situation. Note that we use identical filter tuning during the evaluation process to better show the tracker general fit to variable urban situation.

Finally, the last dataset (Dataset 0057) actually has the least score among all datasets. This dataset shares a similar situation with that of Dataset 0017 and 0018. The ego vehicle is stopping at large junction due to red traffic light. In this case, another car is situated in front of the ego-vehicle, instead of traffic light poles. The occluding car then produces a large blind spot for the LIDAR sensor (c.f. Figure 5-5). Some target cars can be seen to be completely missed by the LIDAR sensor. The large blind spot contributes to the increasing number of false negative and loss of box precision due to the ego vehicle staying in that position for almost half lifetime of the dataset frame counts.

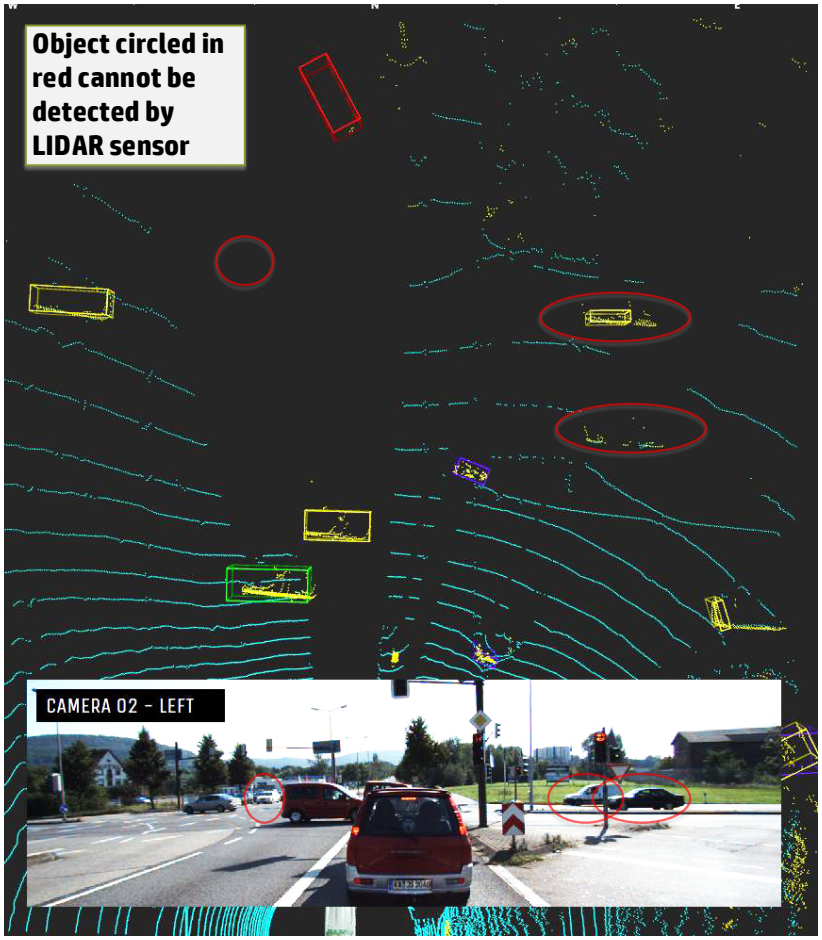


Figure 5-5: Large blind spot caused by occluding car in front

In overall, the benchmarking process yields a better understanding of the tracker performance in variation of urban situation and different class of traffic object. Car, van and pedestrian class objects are tracked reliably by an average of above 86% by the MOT system. Quality measures support the scores of the CLEAR metrics, MT tracks outnumber ML tracks by significant margin in all datasets, including the complex scenario with constant sensor frame change, the presence of persistent occluding object and manoeuvring targets. Note that the designed system provides a robust tracking to the extent of sensor physical limitation: for majority of traffic object uncertainties in position measurement (i.e. detector box) is handled to yield tracking with sufficient accuracy and precision, however for some edge case where part of object is significantly occluded and/or has poor reflectance, we see that the performance visibly deteriorates.

5-5 Comparison to State-of-the-Art

The use of both established MOT metrics and public dataset are also useful to enable objective comparison to the performance of state-of-art tracker. The utilised metrics, namely the MOTP, MOTA, MT, ML, FN and FP are common measures for tracking performance. Publicly ranked benchmarks (see KITTI Object Tracking Evaluation 2012[139] and 2017 MOT challenge[140]) use this metrics, as well as numbers of MOT-related literature[141, 135, 49, 72, 70, 142, 143, 39, 91].

Compared to camera tracking, there are notably fewer LIDAR literature which put significant concern on evaluation using established metrics. Some notable publication which uses both Velodyne and CLEAR as metrics are that of Ye[91] which uses geometric-based tracking circle method, Xiao[144] which uses point assignment task based on energy function, Spinello[145] which uses Bottom-Up Top-Down Detector (BUTD), and Kaetsner[146] which use Generative Object Detection and Tracking. The comparison can be seen in Table 5-4.

Table 5-4: CLEAR comparison to state-of-art trackers

Method	MOTA	MOTP	FN	FP
Proposed Framework	86.12 %	n/a. in m	11.89 %	1.92 %
Tracking circle[91] (averaged)	86.5%	< 0.2 m	3.5%	8.0%
Energy [144]	84.2 %	< 0.12 m	5.8 %	2.77 %
BUTD[144]	89.1 %	< 0.16 m	2.6 %	7.6 %
Generative[146]	77.7 %	< 0.14 m	8.5 %	10.1 %

These works use different criteria to compute the MOTP. The proposed approach takes into account the position *and* dimensional integrity of the tracked objects, thus the bounding box overlap ratio is used. Meanwhile, these works consider only the precision of centre point of the detected object, so the MOTP is based on Euclidean distance error instead. In addition, only work of Ye[91] deals with a sensor mounted in moving car. The other three use the dataset recorded on ETH Zurich Polyterrasse, which deals with a static frame of reference in university canteen scenery and populated only with pedestrian. (i.e. not remotely close to urban scenario). While results of Ye[91] would be the best control comparison to this thesis work, it only uses 2 datasets with unspecified ground truth details.

A general overview indicates our proposed approach has a comparable accuracy (± 3% differences) to state-of-art, but accompanied with quite larger percentage of FN (11.89 % vs 2.6% if we compare with BUTD). In the previous section, it has been found that the vast majority of FN is contributed by a single dataset with complex scenario (out of 10). Nevertheless, if we inspect other datasets individually, the proposed FN percentage would be on par (2-7%). Note that conclusive comparison about performance cannot be made unless same dataset and ground truth are to be used.

In regard to publicly ranked benchmark, to the best of author knowledge, all established MOT ranked benchmark deals with camera-based tracking, so while our results *cannot* be directly compared in 1-to-1 basis to the ranked trackers, a general comparison can be drawn. This is especially true for trackers ranked in KITTI Object Tracking Evaluation 2012, as they share overlapping recorded scenery (urban situation) and ground truth with this thesis.

General comparison from Table 5-5 suggests the proposed tracker is performing on-par with some of the highest ranked trackers (MOTA is 0.5% lower than TuSimple). MOTP score is notably higher, but we should factor that rank assume 2D bounding box in camera coordinate, not world-coordinate. Track Quality measures, however suggest our tracker coverage of target objects would be in third place overall. The higher number of FN and IDSW in a specific dataset is again the major contributor to the higher percentage of ML (3-6 % higher than TuSimple and RRC-IIITH). Note that, although the source of the raw data and ground truth for this thesis implementation is coming from KITTI Dataset, we have to select the subset of the ground truth based on criteria outlined in Section 5-3. Moreover, the ranked benchmark uses a different sequence of datasets than the provided raw data. The evaluation sets notably are not accompanied by necessary tracklet file for LIDAR benchmark, barring its use for benchmarking our approach.

Table 5-5: Top 10 Camera-based KITTI Object Tracking Evaluation 2012 benchmark[73] results for car object. Only online methods are included. Retrieved from http://www.cvlibs.net/datasets/kitti/eval_tracking.php

Method	MOTA	MOTP	MT	ML	Runtime
Proposed Framework.	86.12 %	91.01 %	70.64 %	9.33 %	0.1 s / 1 core
TuSimple	86.62 %	83.97 %	72.46 %	6.77 %	0.6 s / 1 core
RRC-IIITH	84.24 %	85.73 %	73.23 %	2.77 %	0.3 s / 1 core
IMMDP	83.04 %	82.74 %	60.62 %	11.38 %	0.19 s / 4 cores
DuEye	80.64 %	83.52 %	61.85 %	5.85 %	0.15 s / 1 core
JCSTD	80.57 %	81.81 %	56.77 %	7.38 %	0.11 s / 1 core
wan	78.07 %	82.83 %	51.38 %	13.38 %	0.1 s / 1 core
CCF-MOT	77.08 %	78.36 %	52.62 %	13.08 %	1.1 s / 1 core
MDP[147]	76.59 %	82.10 %	52.15 %	13.38 %	0.9 s / 8 cores
SCEA[148]	75.58 %	79.39 %	53.08 %	11.54 %	0.06 s / 1 core
CIWT[149]	75.39 %	79.25 %	49.85 %	10.31 %	0.28 s / 1 core

Summarily, the proposed approach performance results are found to be favourable through general comparison with the state-of-art. Both Track Quality Measure metrics support the result of CLEAR metrics that the proposed approach sufficiently perform the MOT task for its intended domain when compared to state-of-the-art object trackers.

Chapter 6

Conclusions and Future Work

The chapter highlights important finding found throughout the thesis research and implementation phase. First, we summarize the proposed solution and contribution of this work. Then we recall the research objectives and recap how we addressed them in this thesis. Ultimately, potential improvements of the current research shall be formulated as future works.

6-1 General Conclusions

The increasing adoption of autonomous driving toward SAE level 5 demands a reliable object tracking on all environment; we have identified that urban introduce large uncertainties due to clutter and manoeuvring road object which calls for a robust tracker with 3D LIDAR as a suitable sensor. Currently, there is no complete MOT system specifically tailored for urban situation. Therefore, an integrated MOT framework has been introduced in this thesis.

The framework encompasses the complete process of multi-object-tracking: it takes raw 3D LIDAR data, and yield objects kinematic pose embedded with static and dynamic classification which can serve as the basis for the mission accomplishment of higher perception module. The detection result is designed to be occlusion-aware by utilizing slope-based ground removal with consistency check and L-shape fitting. Additionally, over-segmentation is handled by means of predicted information exchange with the tracker component. The tracker itself employs probabilistic adaptive filtering based on coupled IMM-UKF- JPDA that allows the tracking of traffic objects characterized by uncertainties in term of motion type and clutter. In addition, the tracker also preserves the dimensional integrity of tracked objects by means of computationally light logic-based filter based on heuristic and the use box frame history.

Finally, evaluation using established MOT16 metric suggests that the tracking performance is favourable in varying pre-recorded real-world urban scenario. General comparison to state-art demonstrates that the proposed approach is performing on-par with other implementation. Additionally, since the framework is designed and found to run in a real-time manner (under 100 ms) it is expected that the framework is applicable for real autonomous vehicle deployment.

6-2 Fulfillment of Research Objectives

The objectives of this thesis and the aspects addressed within the framework of this research are summarized as follows:

1. *Identify the requirements of object tracking and detection in term of robustness against tracking target uncertainties and 3D LIDAR sensor limitation in urban situation*

   Throughout the introduction of fundamentals and implementation phase, the thesis report has discussed the requirement of MOT followed by proposed solution based on state-of-the-art. To recapitulate, the requirements are as follows:

   (a) The tracker has to be able to track in the presence of clutter.

   (b) The tracker has to be able to estimate and track manoeuvring targets.

   (c) The detector has to detect arbitrary object in the presence of range and view-point dependant occlusion of LIDAR sensor

   (d) Detection and tracking has to be done together in real-time to sufficiently capture fast-moving object, in this case, the sensor sampling rate (10 Hz) is defined as the real-time deadline.

2. *Design algorithms which address the defined requirements to achieve the precise and accurate task of both detection and tracking by a combination logic-based rule to explicitly handle occlusion and probabilistic adaptive filtering*

   The detector employs slope-based groudn removal augmented by consistency check and median filter. This allows occluded ground area to be removed efficiently. In addition, the use of L-shape fitting prevent mis-orientation of bounding box for partially occluded object.

   The use of optimal Bayesian filters succinctly address the presence of uncertainties in tracking problem. The algorithm models tracking object states as random variables which evolve under stochastic process. PDAF filter algorithm is designed to tackle data association confusion and IMM filter is realized to better handle motion uncertainties rather than just treating it as unmodelled dynamics (process and measurement noise). In addition, since the stochastic motion models are of non-linear nature, UKF is implemented to achieve more accurate and precise state estimation. The filters are deliberately chosen due to their computational efficiency. In addition, logic-based rule filter is also designed to augment the rest of detection and tracking, which enhance the accuracy of detection and tracking with minimum computational cost.

3. *Perform real-time implementation and tuning of object detector and tracker in C++ with an interface to real-world, publicly available datasets.*

   The previously designed algorithm is implemented within integrated MOT framework: a real-time, causal and robust MOT system against uncertainties found in urban situation. A coupled filter utilizing three combined Bayesian filters (IMM-UK-JPDAF) simultaneously tackle association uncertainties, motion uncertainties and estimate non-linear stochastic motion model in real-time. Tuning is done based on real-world sensor recording (KITTI Dataset) to achieve practical efficacy and validation.

The framework is made to be modular and exploits the inter-process information exchange to enhance detection process which is otherwise not possible. Note that, reduction of detection imperfection translates to better tracking accuracy and precision. The MOT is developed based on mix of C++ and MATLAB codebase so tuning and algorithm change can be efficiently performed. The end result is a full implementation in C++ which enables real-time execution of the whole system. The implementation is shown to meet the deadline of 100 ms for various real-world datasets.

4. *Design and perform evaluation based on established metrics against publicly accessible real-world LIDAR data for validation purpose and objective comparison with state-of-art works.*

   An evaluation based on established metrics, the MOT16 has been formulated and performed. MOT16 consists of well-known sub-metrics (CLEAR and Track Quality Measure) which enable this work to be compared head-to-head to state-of-art MOT in term of raw number. In addition, the use of KITTI Public Datasets validates the MOT framework against real-world data; diverse datasets are deliberately chosen to evaluate if the MOT is able to perform in a variation of urban driving scenario.

   The evaluation results highlight that the designed and implemented MOT system yield satisfying performance in term of accuracy and precision in most given urban situation. A closer inspection (cf. Appendix A) also shows that the countermeasure of handling uncertainties (probabilistic adaptive filtering and logic-based filter) are shown to increase accuracy and precision of the tracker. In some case, it is also found that the MOT system performance will degrade when the complexity of scenario scales up (namely, more crowd and moving sensor of reference) and limited efficacy of the sensor under attenuation (namely in regard to reflection and blocking object).

5. *Perform comparison with related state-of-art works which also use comparable MOT metrics.*

   General comparison with related MOT implementations which use CLEAR and/or Track Quality Measure metrics has been made. Four related-works with Velodyne-based MOT specifically employ CLEAR metrics for evaluation, although only one of them uses Velodyne mounted in a vehicle. The proposed approach is found to perform on-par but sports a larger number of FN.

   Another comparison with publicly ranked camera-based Object Tracking Evaluation 2012 list has also been made. The proposed MOT has similar accuracy score and a slight edge on the precision top 10 highest ranked tracker in the list. The number of mostly lost tracks is noted to be high relative to the highest ranked tracker.

   Note that on both instances of comparison, only a general comparison can be made due to notable key differences in the type of dataset and modal sensor used. The lack of precisely comparable results in literature re-emphasize one of the thesis contributions on introducing evaluation for 3D-LIDAR based Urban MOT; the evaluation result of this work can be potentially used as a comparable baseline for future works.

6-3 Future Works

Perception for the autonomous vehicle is research in motion; it is expected the limitations found in the proposed MOT will be supplanted by rapidly evolving state-of-art. Here, some of future researches are proposed which attempt to address limitation found in this work.

Specific to implementation presented in this thesis, these are several proposed improvements which can be immediately addressed:

More primitive shape recognition: the detector currently only consider L-shape geometry for deriving a correct bounding box in case of an occluded object. Other primitive-shape detection, like U, V, and I shape as introduced in[43] potentially allows better detection results.

Object Existence Tracker: the JPDA filter is shown to handle tracking in clutter, a further extension of JPDA, the Joint Integrated PDA[31] can address false detection related to over-segmentation for low reflectance object.

Use Dual Estimation in the filter: the UKF filter is currently still tuned by hand and knowledge of the real object motion behaviour, it is possible to employ a dual estimation[150] where both state-estimation and parameter estimation are coupled.

Use alternative motion model: a more advanced motion model aimed at vehicle tracking such as Constant Steering Angle and Velocity (CSAV), and Constant Curvature and Acceleration (CCA)[151] potentially capture the motion of targets object better. More recently, a Constant Speed Changing Rate and Constant Turn Rate (CSCRCTR) model seeks to integrate different motion modes into a uniform model, although notably it is designed for PF[152].

The following are some more general aspects which are proposed as more distant future research questions.

Narrow down Real-Time Constraint: the proposed framework notably are deliberately conservative in regard to choosing algorithm based on computational complexity, this also to some extent limits the use of available sensing data such as camera image that can be a secondary source for sensor fusion. Future works can overcome this limitation by deploying the prototyping in real car embedded platform and narrowing the definition of "real-time", so a precise limitation can be defined more categorically based on current known target hardware. Furthermore, the recently announced autonomous - vehicle focused platform, both hardware and software such as Nvidia drive PX2 and Automotive Grade Linux[153] aims to enable computationally intensive algorithm such Deep Learning[154] and offer dedicated image processing module. This allows more possibilities of doing MOT Task beyond traditional Bayes filter. A more complex semantic classification can be achieved more than static and dynamic, for example using stixel to distinguish car and pedestrian[17].

Reduce heuristic-based, use optimal solver: the rule-based filter is a sensor-dependent approach and is actually analogous to hard decision filter. As the computational power increase, a probabilistic approach, or other more optimal solvers such as tracking by neural network[95] and machine learning[136] should be preferred.

Sensor fusion: reflecting from the framework evaluation, The 3D LIDAR has shown to be a capable sensor to be used as primary sensing means. However, we see that the use of single sensor will make the limitation of such sensor into hard limits of object tracking capability. Use of multi-sensor tracking[50, 155, 55, 156, 57, 157, 158, 159, 58] is a viable alternative.

Appendix A

Evaluation: Beyond Tracking Metrics

A-1 Individual Tracks

The discussed benchmarks in Chapter 5 (Evaluation) encompasses aggregate performance of the tracker in the multi-object scenario. To be able to inspect the tracking performance more closely, two sample of tracked object are selected to evaluate the tracker main functions, namely (1) predicting the target trajectory despite uncertainties and noise, (2) the correction of (self) occlusion and over-segmentation and (3) classification of motion model.

The selected two objects can be found inside Dataset 0005 which was recorded in curved, narrow urban inner-cityroad. Here the sensor frame of reference is constantly moving combined with the target which also manoeuvring, which suggests these two are among the most complex tracking scenario found in KITTI Dataset. Both tracks are also recorded for relatively long compared to the rest of other tracks (averaging on 3-5 second duration for dynamic objects) so the filtering steady-state performance can be more represented.

The accompanying first plot shows three tracker trajectories in X-Y plane superimposed to ground truth. The tracker trajectories are explained as follow:

1. **Filter state** refers to raw IMM-UKF filter state of the predicted position
2. **Measurement** corresponds to the detector output (bounding box).
3. **Perspective corrected** trajectory refers to filtered state after perspective correction has occurred (c.f. Subsection 4-7-3).

The second plot shows the UKF filter states over all time steps, and the third plot shows the probability estimates of the IMM filter for the object moving under certain motion model Mode 1 (CV), Mode (CTRV), and Mode 3 (RM) (c.f. Appendix D).

Van (Dataset 0005)

The tracker is seen to track object van 3 time-steps after the first detection (correspond to 400 ms). Observing spatial evolution in Figure A-1. The measurement followed the ground truth closely until $t = 7$, here persistent occlusion starts to appear and the front part of the van gradually reduces until almost the whole van disappears at $t = 7$ (refer to Figure A-2). Subsequently, the back of the van is detected again at $t = 8$. Here we notice that the measurement of y position has a constant bias: the y position is incorrect since only small part of the van is visible, and the centre point is shifted backwards, the Kalman Filter (see Figure A-3) estimates accordingly also affected by this bias, as seen by the velocity vector.

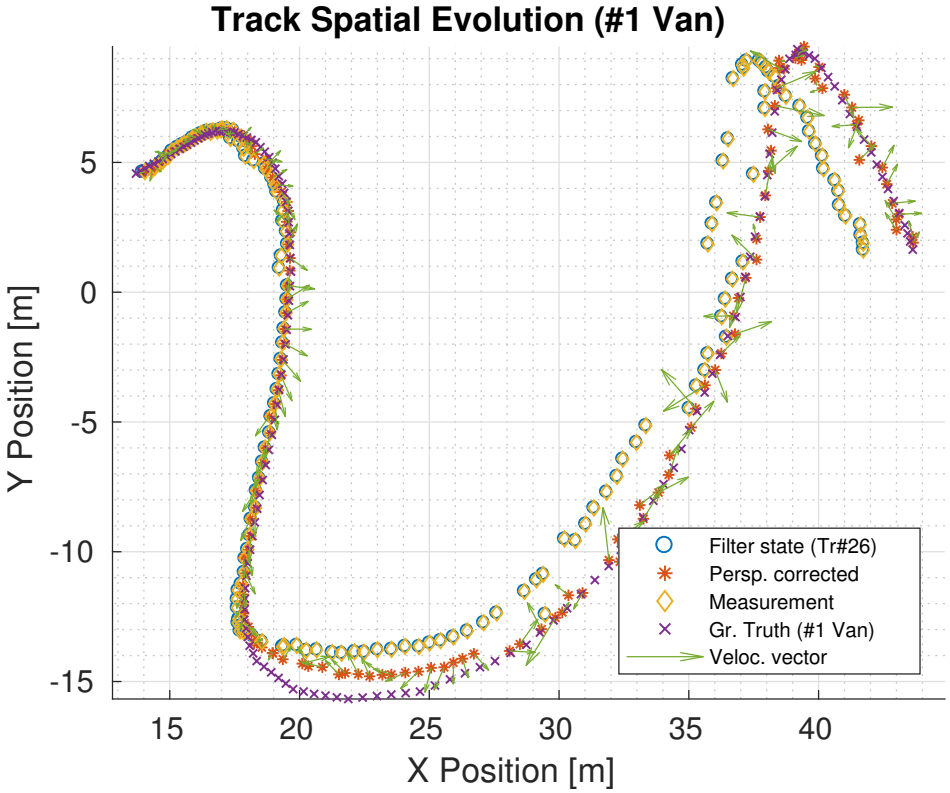


Figure A-1: Spatial Evolution of Object "Van"

However, since the full dimension size has been stored in history, the perspective correction is applied and thus more accurate y position is obtained. At $t = 13$ there is no correct measurement provided and a false positive occurs, the track is associated with another detected object (noise). At the next time step, the tracking resumes with correct measurement until $t = 14$, in which the object completely disappear from the LIDAR sensor frame.

The mode probabilities plots (Figure A-4) reflect that the van mostly was moving under CV model with negligible turn rate (correspond to mode 1), during sharp, sudden turn ($t = 3.5$ and $t = 11$) the IMM filter considers the object to be moving under random motion (mode 3), with small probability it is moving under CTRV model (mode 2). At the end of the track in which the object is leaving the LIDAR sensor frame the measurement deteriorate, and consequently the IMM predicts that the object is a noise moving under RM model.

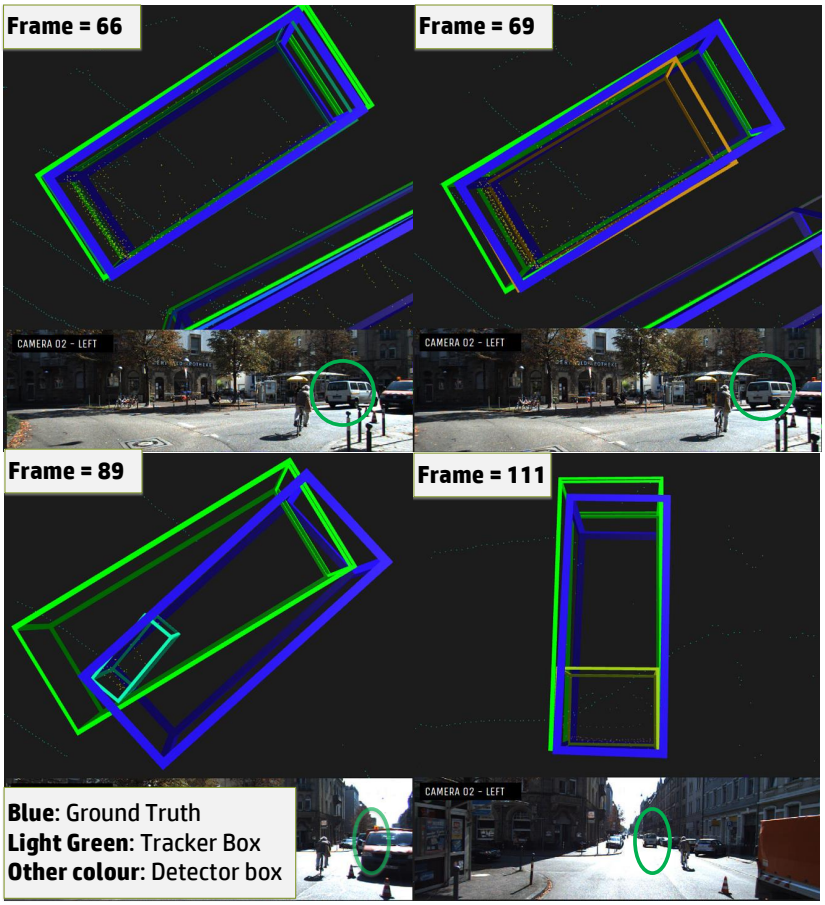


Figure A-2: Tracker box retainment on persistent occlusion

Recall the reasoning that the yaw estimate of Kalman filter is not to be used for bounding box heading. The movement of the Van demonstrates that the yaw estimate of point-based tracking change significantly depending on the movement of the centre point (cf. velocity vector arrow). Especially when an occlusion occurs and the centre point is shifted incorrectly. The use of logic-based, box update can be seen to accurately shift back the box to correct position based on previously known dimension and position of the target object relative to ego-vehicle.

IMM filter switching can be seen to augment the tracker to follow the movement of manoeuvring object, and in this case combined with the change of sensor frame of reference. Should the filter only utilise one single motion model, the tracking performance is expected to at least degrade, or at worst induce track lost.

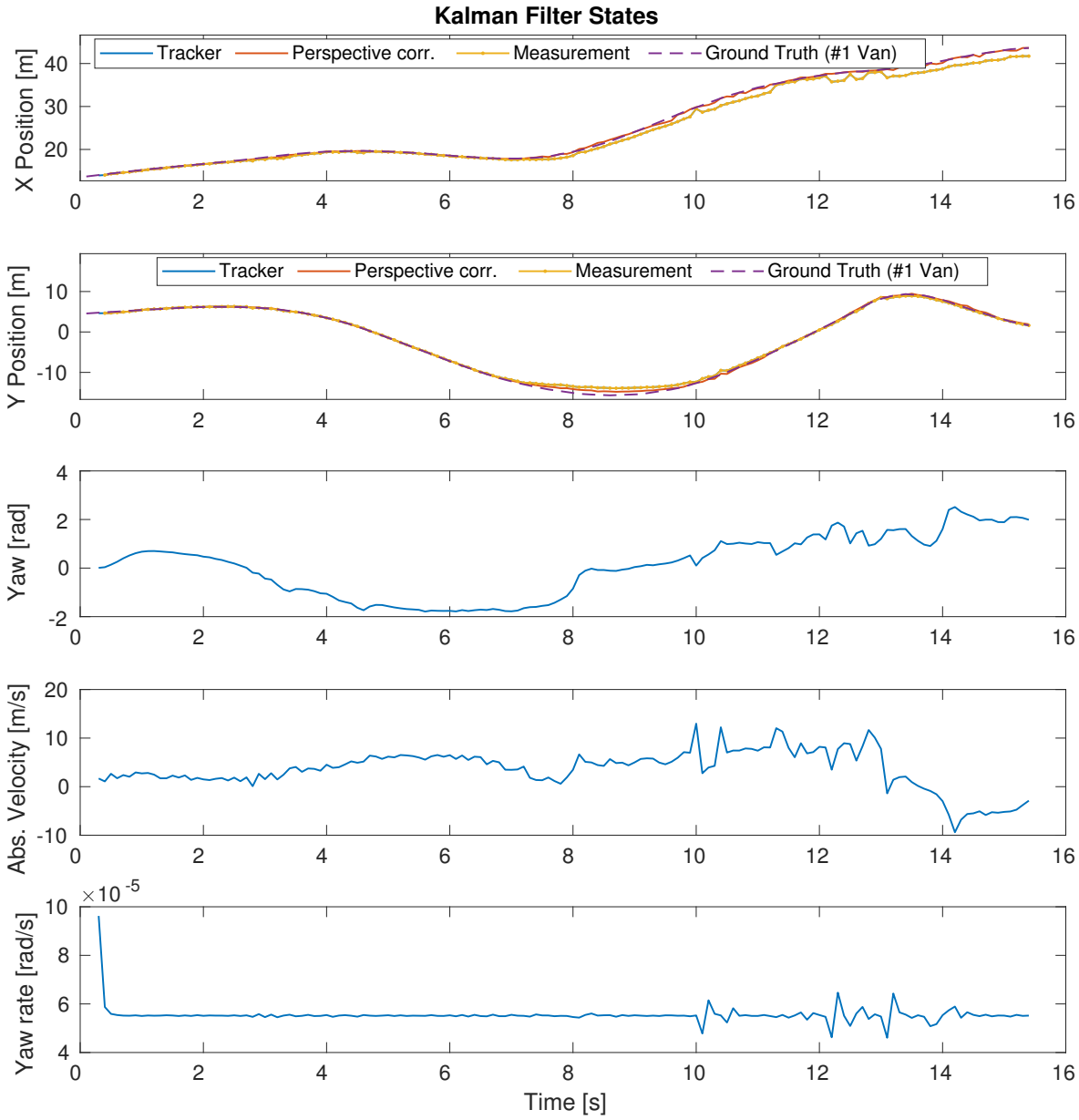


Figure A-3: Kalman Filter States of Object "Van"

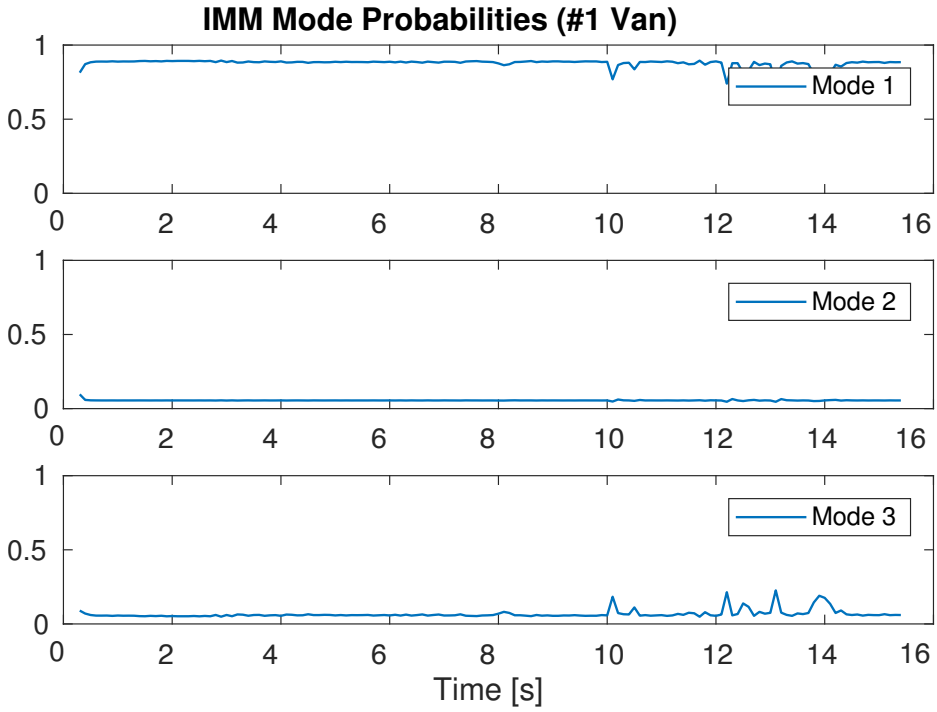


Figure A-4: IMM probabilities of Object "Van"

Cyclist (Dataset 0005)

The cyclist trajectory (Figure A-6) is characterized with persistent self-occlusion which is indicated by a rather constant offset between the ground truth and measurement centre point. A spike in velocity and yaw estimates are expected when the measurement is unreliable (see the velocity vector). Position estimate-wise, the perspective correction compensates for the shrinking size so that the track remains close to the ground truth.

However, there are edge cases where the correction actually exacerbates detection error at such as at $t = 10$. At this time step, the back part of cyclist consists of a minuscule number of point cloud that inadvertently considered as outliers (see Figure A-5). Otherwise, the track follows the ground truth as expected, until the cyclist enters the edge of LIDAR sensor frame where detection becomes too unreliable.

The mode probabilities plot at Figure A-8 shows that the object is also moving under CV model, except when $t = 4.5$, an almost 360 deg loop (correspond to ego-vehicle sharp turn) the turn rate spikes, and thus the probability it is moving under RM rose sharply. In overall the filter is demonstrated to sufficiently estimate the trajectory of the cyclist while simultaneously maintaining dimensional integrity.

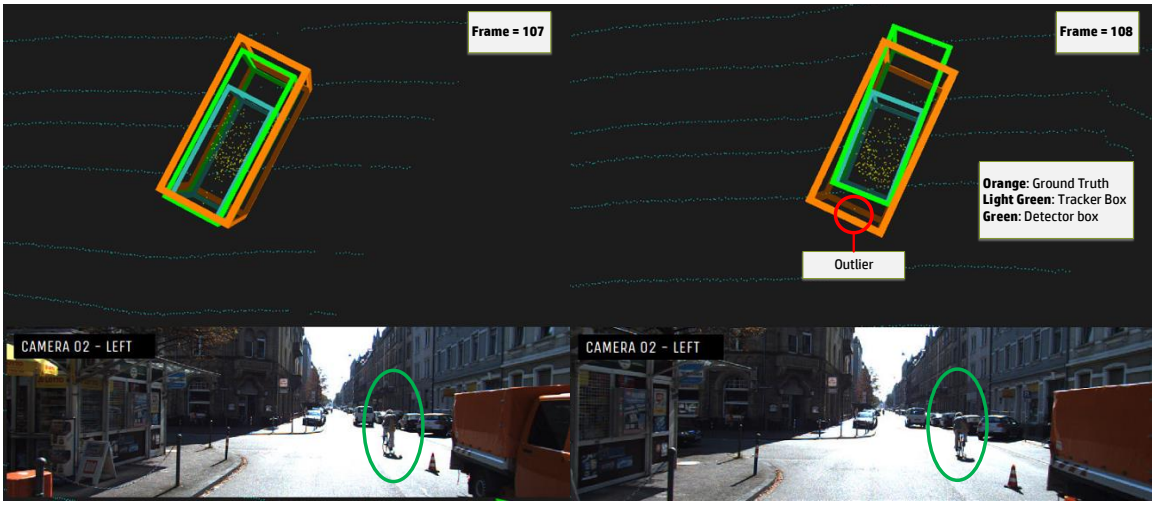


Figure A-5: Outlier points in detection of a cyclist. The detector is shifted frontward causing inaccuracy in the y position.

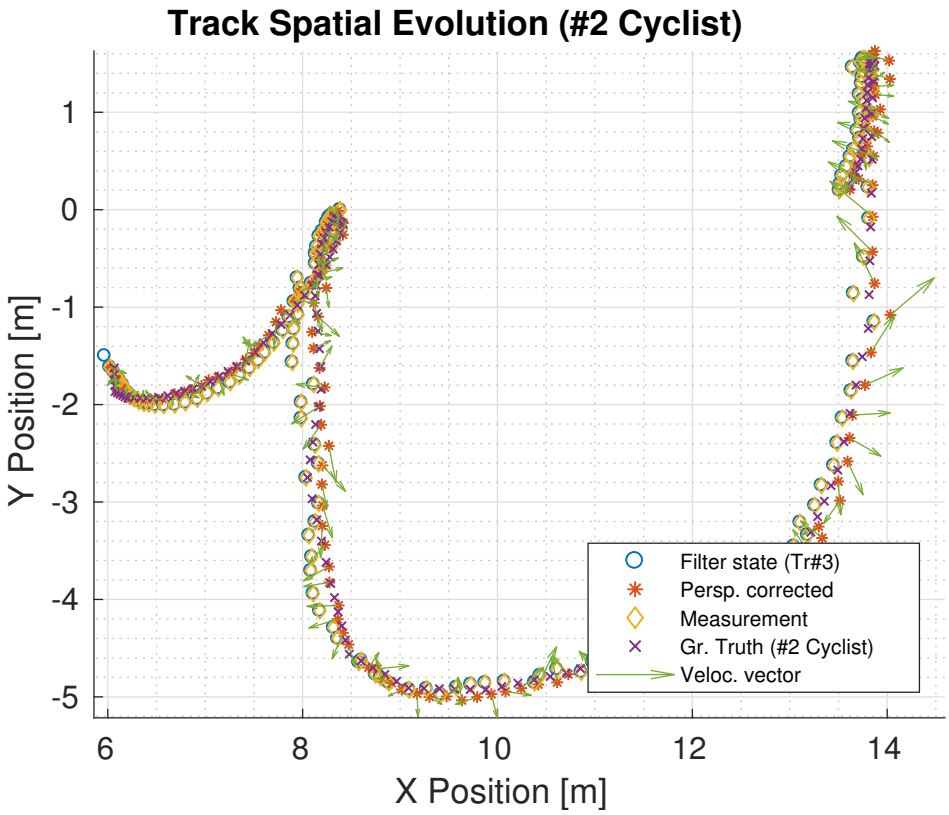


Figure A-6: Spatial Evolution of Object "Cyclist"

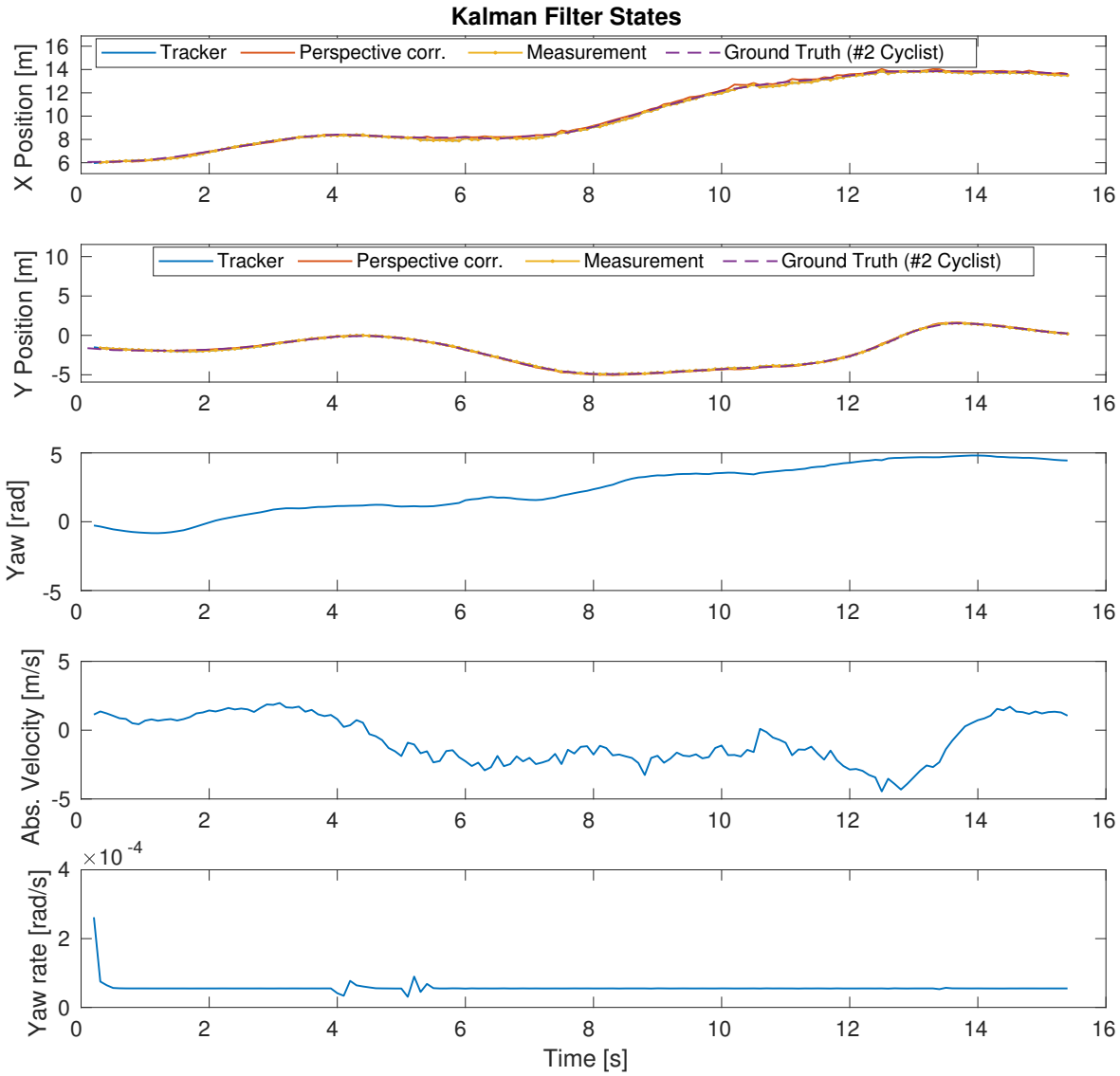


Figure A-7: Kalman Filter States of Object "Cyclist"

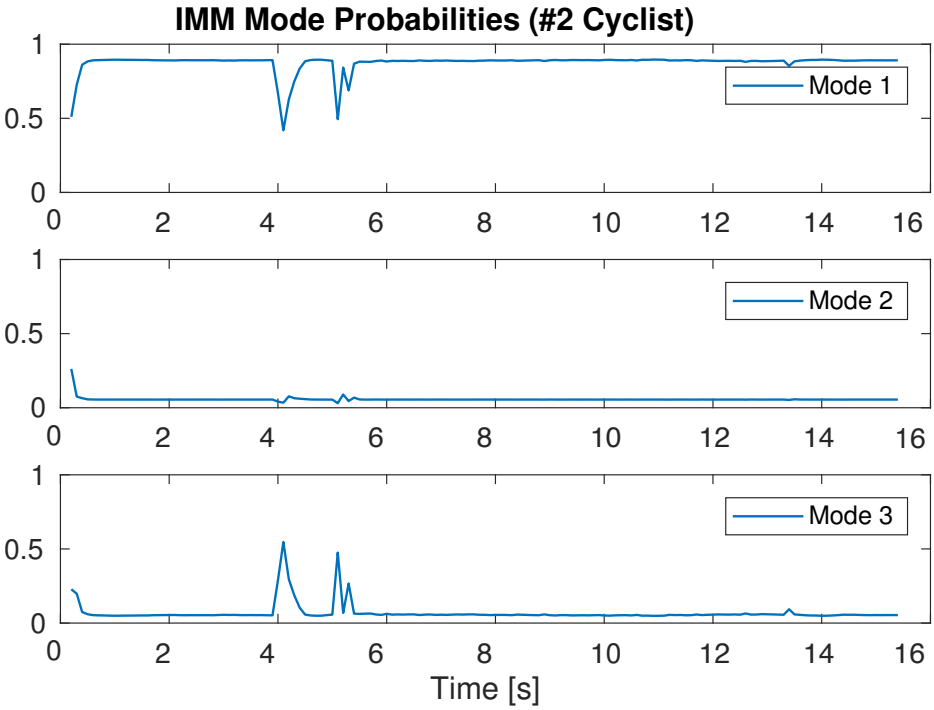


Figure A-8: IMM probabilities of Object "Cyclist"

A-2 Computation Time

One of the major requirement outlined in the introduction and throughout design and implementation phase is that the MOT has to be suitable for real-time, online tracking. To verify this requirement, the per-cycle execution time in 10 datasets used for benchmark were measured using C++ `std::chrono::high_resolution_clock` class, which represents the clock with the smallest tick period provided by the execution platform. From the measurement of 2111 frames, it is found that 96% of the time, the implemented MOT system managed to complete the execution at or below 100ms. The frequency distribution can be observed in Figure A-9.

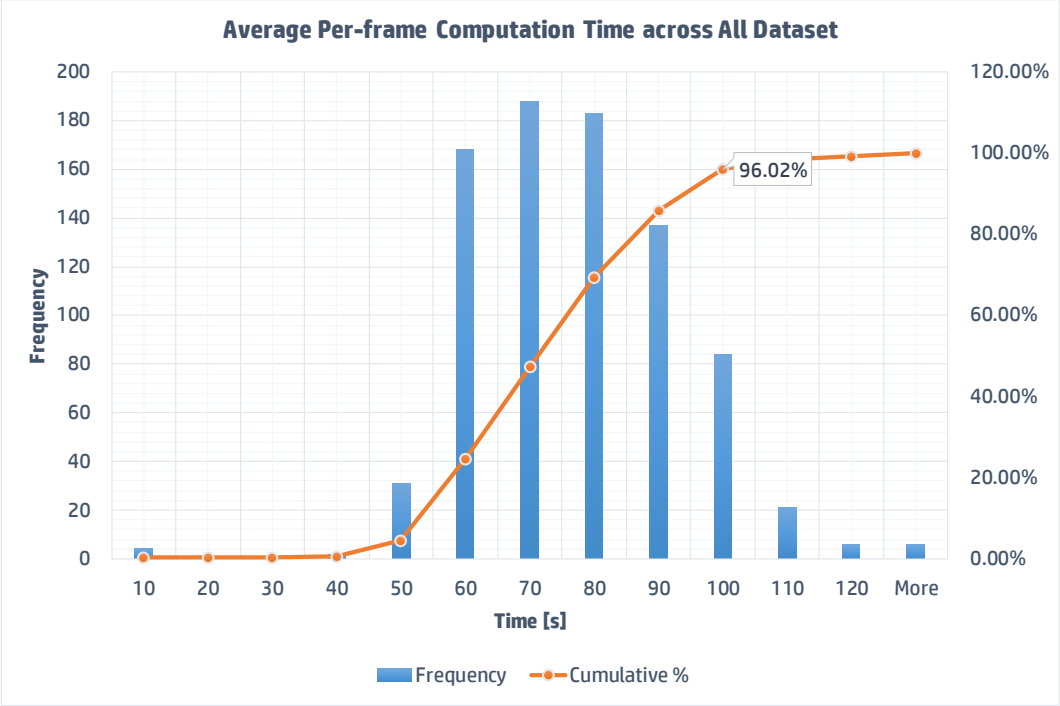


Figure A-9: Histogram of computation time per one cycle of MOT. The significant majority (96.02%) of cycle times are below sensor sampling time (10Hz) .

It is also useful to inspect which of the MOT components are responsible for the majority of computational time, so the per-dataset measurements are averaged and broken down into component specific time as seen in Figure A-10. Notice that the average number of detected objects are also provided to give reader overview on the relation of each dataset scenario and computational demands.

First, we, see the ground extraction runtime (averaged at 17 ms) are predictable since the computational load of slope-based ground extraction relies on the density of polar grid, which is fixed along with the LIDAR reachable range.

Next, we see the majority of computational cycles were spent for detecting the object post ground removal (i.e. clustering and bounding box fitting), a correlation can be suggested by inspecting the average of detected object and the detector component execution time across all datasets. To put it simply without loss of generality: *more objects in LIDAR sensor frame*

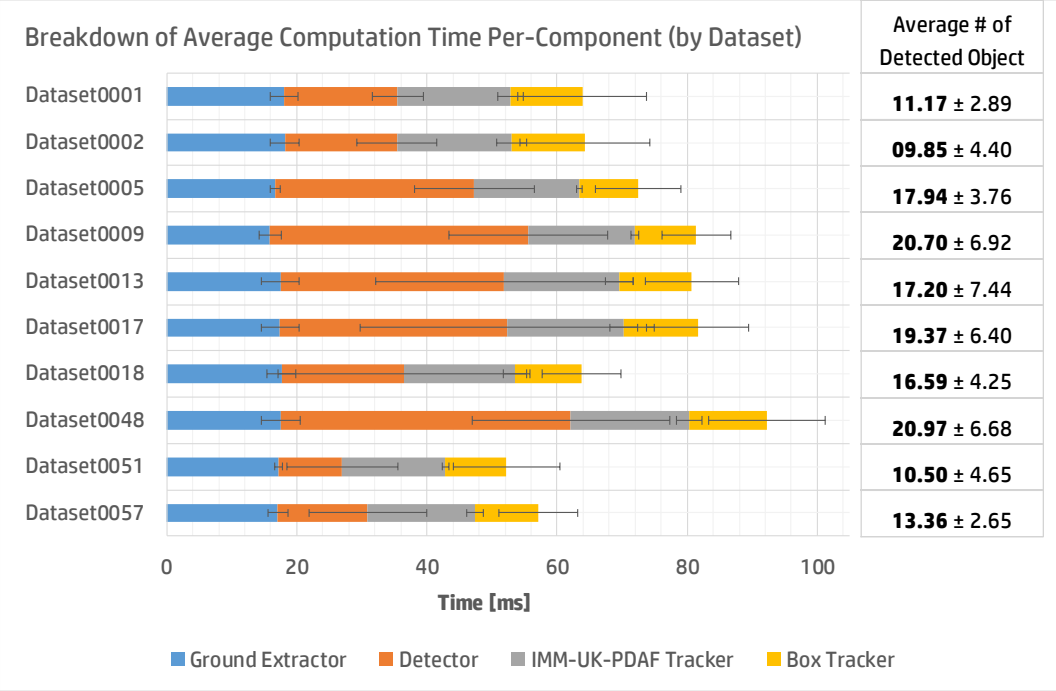


Figure A-10: Breakdown of computation time per MOT component and dataset. Note that increase of detected object in sensor time-frame demands more computational power

translates into a higher computational load. The detector runtime is found to scale roughly linearly to the number of objects. Naturally, we see a significant standard deviation of the averaged data, since the number of road objects varies a lot in reality. Dataset 0048 as has been discussed on the benchmark discussion, comprised of busy two-way roads. Therefore, a high number of traffic objects stays on sensor frame longer compared to other datasets. Thus we see a large overall computational time (48 ms), and several frames actually violate the 100 ms limit.

The IMM-UK-PDAF on the other hand, shows that the Bayesian filter scales rather gracefully with the increasing number of objects, it stays at an average of 17 ms with small deviation across all datasets. Here we used a fixed number of the active track (i.e. filter instances). Therefore, the computation for UKF is within bound as expected. Finally, the box tracker shows the lowest computational time (10 ms) among other components since it is only a logic-based filter, the large standard deviation is found since the computational time directly relates to the number of detected objects.

The evaluation of computational time demonstrated that the proposed MOT implementation capable of consistently meeting the proposed real-time deadline of 100 ms, and scale in (approximately) linear way with the number of objects in sensor frame. Although the execution time is notably measured on non-real-time hardware and Operating System stack (Intel Core i7-7700K with 8GB of RAM, running Ubuntu GNU/Linux 16.0.4.3 LTS), The implementation is deliberately designed to be cross-platform: it is implemented in ISO/IEC 14882-compliant C++14 code, and thus testing on car modern embedded computing platform (such as Nvidia Drive PX) is possible simply with compiler change and minimal code adjustment, paving its way for the application in real autonomous vehicle embedded platform

A-3 Static and Dynamic Classification

The classification of the static and dynamic attribute of each tracked object are also evaluated. The 10 identical datasets are re-used as reference. Since there is no absolute static and dynamic attributes nor absolute speed given embedded on KITTI ground truth, the relative velocity per frame is calculated by finite displacement over sampling time, furthermore using built-in odometry of the ego-vehicle, the absolute velocity is obtained. An object whose absolute velocity is below 5 m/s is categorized as a static object. The use of a somewhat larger threshold is necessary to take into account the uncertainties of vehicle odometry data. In addition, only classification from mature track is taken into account to ensure IMM estimate has converged into a steady-state.

The Confusion matrix is employed to visualize the tracker performance as can be seen in Table A-1. The four numbers in the quadrant correspond to True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP) written in the clockwise direction. Precision, Recall, FN/FP rate, and Accuracy are also added as metrics of evaluation.

Table A-1: Confusion Matrix for static and dynamic classification

Confusion Matrix		**Tracker**		
		Dynamic	*Static*	
Ground Truth	*Dynamic*	1499	61	**Recall** 90.41 %
	Static	77	726	**FP rate** 3.91 %
		Precision 92.25 %	**FN rate** 9.59%	**Accuracy** 94.16 %

First, the large number of TN indicates that large majority if the tracking targets are dynamics objects, we see a similar number of FN and FP, which are comparatively lower than its true counterpart. The number of FN indicates that the result of uncertain detection makes the centre point of the detected object to be constantly changing, if the change is significant and to some extent resembles the moving object trajectory (for instance when combined with sensor reference change), the IMM filter will classify the object as dynamic. FP on the other hand is caused when a dynamic object is moving with close to stopping velocity (e.g. a car about to stop at the traffic light), the false classification occurred for the few time steps before the vehicle completely stops.

In overall, with sufficiently high accuracy (94%), precision (92%) and recall (90%), the proposed approach reliably classify statics and dynamic objects. However, care has to be taken as the IMM requires converging time to be able to reach a reliable estimate. In our implementation all track are assumed and reported to higher perception module as dynamics until a reliable velocity and IMM probability estimates are obtained. This conservative decision is necessary when static and dynamic classification result is to be used for a critical mission (e.g. vehicle path planning); for safety reason only information with high degree of confidence should be propagated.

Appendix B

Algorithms

This appendix lists the pseudo-codes of each implementation found in Chapter 4. All adjustable parameters value can be found in Appendix C. The C++ and MATLAB source code, including the evaluation script may be retrieved from RTWH Aachen *git* repository at https://git.rwth-aachen.de/pem/Object-Tracking

Algorithm 1 Ground Removal and Classification

procedure REMOVEGROUNDANDCLASSIFY(rawPointClouds, groundPoints, elevatedPoints)
    $m_{pgrid} \leftarrow$ <adjustable parameter>
    $n_{pgrid} \leftarrow$ <adjustable parameter>
    $r_{min} \leftarrow$ <adjustable parameter>
    $r_{max} \leftarrow$ <adjustable parameter>
    $T_{hmin} \leftarrow$ <adjustable parameter>
    $T_{hmax} \leftarrow$ <adjustable parameter>
    $h_{sensors} \leftarrow$ <adjustable parameter>
    $T_{slope} \leftarrow$ <adjustable parameter>
    $T_{hdiff} \leftarrow$ <adjustable parameter>
    $T_{flat} \leftarrow$ <adjustable parameter>
    $T_{consistent} \leftarrow$ <adjustable parameter>
    $s_{kernel} \leftarrow$ <adjustable parameter>
    $h_{tol} \leftarrow$ <adjustable parameter>
    polarGridPCL $\leftarrow$ createAndMapPolarGrid(rawPointClouds, $m_{pgrid}, n_{pgrid}, r_{min}, r_{max}$)
    for each cell $\in$ polarGridPC **do**
        $z_i \leftarrow$ min(cells.Z)
        if $z_i > T_{hmin}$ **and** $z_i > T_{hmax}$ **then**
            cell.$h_i \leftarrow z_i$
        else if $z_i > T_{hmax}$ **then**
            cell.$h_i \leftarrow h_{sensors}$
        end if
        cell.$m \leftarrow$ computeGradientoAdjacentCell(cell)
        cell.$h_{diff} \leftarrow$ computeHdiffoAdjacentCell(cell)
        if cell.$m > T_{slope}$ **and** cell.$h_{diff} < T_{hdiff}$ **then**
            cell.isGround $\leftarrow$ true
            cell.$h_{ground} \leftarrow$ cell.h_i
        end if
        if cell.$h_i > T_{flat}$ **and** cell.$h_{diff} < T_{consistent}$ **then** ▷ Consistency Check
            cell.isGround $\leftarrow$ true
            cell.$h_{ground} \leftarrow$ cell.h_i
        end if
    end for
    applyMedianFiltering(polarGridPCL, s_{kernel})
    for each point $\in$ rawPointClouds **do**
        currentCell $\leftarrow$ getCellfromPoints(point,polarGridPCL)
        if point.$z_i >$ currentCell.$h_{ground} \pm h_tol$ **then**
            groundPoints.add(point)
        else if point.$z_i \leq$ currentCell.$h_{ground} \pm h_{tol}$ **then**
            elevatedPoints.add(point)
        end if
    end for
end procedure

Algorithm 2 Connected Component Clustering

```
1: m_clust ← <adjustable parameter>
2: n_clust ← <adjustable parameter>
3: procedure RecursiveConnectedComponent(rawCartesianGrid, clusteredCartesianGrid)
4:     rawCartesianGrid ← negate(rawCartesianGrid)          ▷ set all cell to -1 to
5:     clusterID ← 0
6:     FindComponent(clusteredCartesianGrid, clusterID)
7: end procedure
8: procedure FindComponent(clusteredCartesianGrid, clusterID)
9:     for cellX=1:m_clust do
10:        for cellY=1:n_clust do
11:            clusterID ← clusterID+1
12:            search(clusteredCartesianGrid, clusterID, cellX, cellY)
13:        end for
14:    end for
15: end procedure
16: procedure Search(clusteredCartesianGrid, clusterID, cellX,cellY ) clusteredCartesianGrid(cellX, cellY) ← label
17:    Nset ← neighbours(L,P)          ▷ 4x4 neighbour, scanned counter clockwise
18:    for each cellX',cellY' do ∈ Nset
19:        if clusteredCartesianGrid(cellX',cellY)=-1 then
20:            search(clusteredCartesianGrid, clusterID, cellX, cellY)
21:        end if
22:    end for
23: end procedure
```

Algorithm 3 Duplicate Track Pruning

```
1: procedure PruneTrack(trackList)
2:     historyGateLevel ← <adjustable parameter>
3:     distanceThreshold ← <adjustable parameter>
4:     for i=1:TrackList.size-1 do
5:         for j=1:TrackList.size+1 do
6:             histDiff) ← computeStdDiff(TrackList(i),TrackList(j))
7:             euclidDist ← computeEuclideanDistance(TrackList(i),TrackList(j))
8:             if histDiff < historyGateLevel or euclidDist < historyGateLevel then
9:                 if TrackList(i).Lifetime < TrackList(j).Lifetime then
10:                    TrackList(i).State ← Invalid
11:                else
12:                    TrackList(j).State ← Invalid
13:                end if
14:            end if
15:        end for
16:    end for
17: end procedure
```

Algorithm 4 Static Dynamic Classification

```
1: procedure classifyStatDyn(trackList,boxFrameHistory)
2:     velThreshold ← <adjustable parameter>
3:     k ← getCurrentTimeStep()                          ▷ current frame (time step)
4:     nStepBack ← <adjustable parameter>
5:     for each Track ∈ trackList do
6:         if Track.State = mature and Track.lifeTime > lifeTimeThreshold and
   Track.hasConverge = true then
7:             relVelocity = ← computePastAverage(Track.relVel, boxFrameHistory, nStep-
   Back)
8:             if relVelocity < velThreshold and (Track.probabIMM.M3 <
   Track.probabIMM.M2 or Track.probabIMM.M3 < Track.probabIMM.M1) then
9:                 track.isDynamic ← false
10:            else
11:                track.isDynamic ← true                          ▷ default value
12:            end if
13:        end if
14:    end for
15: end procedure
```

Algorithm 5 Bounding Box Association

```
1: procedure AssociateBB(trackList,detectorBoxList)
2:     distanceThreshold ← <adjustable parameter>
3:     lifeTimeThreshold ← <adjustable parameter>
4:     for each Track ∈ trackList do
5:         if Track.State = mature and Track.lifeTime > lifeTimeThreshold then
6:             nearestMeasurement ← getNearestEuclidCP(Track.MeasurementCPs,
   Track.CP)                                                  ▷ CP is centre point
7:             if euclidDistance(nearestMeasurement, Track.CP) < distanceThreshold then
8:                 measurementIndex ← getIndexMeasurement(nearestMeasurement)
9:                 Track.AssocBB ← detectorBoxList[measurementIndex]
10:            else
11:                Track.AssocBB ← emptyBox        ▷ Track has CP, but not bounding box
12:            end if
13:            if Track.MeasurementCPs.size > 1 then
14:                Track.Ambiguous ← true
15:            else
16:                Track.Ambiguous ← false
17:            end if
18:        end if
19:    end for
20: end procedure
```

Algorithm 6 Bounding Box Update/Retain

procedure UPDATEBB(trackList,detectorBoxList)
    `bbYawChangeThreshold` ← <adjustable parameter>
    `bbAreaChangeThreshold` ← <adjustable parameter>
    `bbVelThreshold` ← <adjustable parameter>
    k ← getCurrentTimeStep() ▷ current frame (time step)
    n ← <adjustable parameter> ▷ look-back step
    bbSmallTol ← <adjustable parameter> ▷ very small number
    frameBoxHistory ▷ box from previous frames, if member at index k is undefined, frameBoxHistory[k] = Track.AssocBB
    for each Track ∈ trackList **do**
        deltaYaw ← |Track.AssocBB].yaw - frameBoxHistory[k-1].yaw|
        deltaArea ← |Track.AssocBB.area - frameBoxHistory[k-1].area|
        deltaWidth ← |Track.AssocBB.width - frameBoxHistory[k-1].width|bbs
        deltaVel ← Track.AssocBB.relVel - frameBoxHistory[k-1].relVel
        deltaYawHist ← |Track.AssocBB.yaw - frameBoxHistory[k-n].yaw|
        deltaAreaHist ← |Track.AssocBB.area - frameBoxHistory[k-n].area|
        if |Vel| < `bbVelThreshold` **and** deltaVel < bbSmallTol **then** ▷ Static object
            Track.AssocBB.area ← frameBoxHistory[k-1].area
            Track.AssocBB.yaw ← frameBoxHistory[k-1].yaw ▷ Restore previously stored box yaw and dimension
        else if deltaYaw < `bbYawChangeThreshold` **and** deltaArea > `bbAreaChangeThreshold` **and** deltaLength > 0 **then** ▷ Sane yaw change detected along with acceptable area increase
            continue ▷ allow box update
        else if deltaYaw < `bbYawChangeThreshold` **and** deltaArea > `bbAreaChangeThreshold` **then** ▷ Sane yaw change detected along with unacceptable area change
            Track.AssocBB.area ← frameBoxHistory[k-1].area ▷ previously stored area
        else if deltaYaw > `bbYawChangeThreshold` **and** deltaArea < `bbAreaMaxThreshold` **and** deltaLength > 0 **then** ▷ Impossible yaw change detected but with acceptable area change
            Track.AssocBB.yaw ← frameBoxHistory[k-1].yaw ▷ set to previously stored yaw
        else if deltaYaw > `bbYawChangeThreshold` **and** deltaArea > `bbAreaMaxThreshold` **then** ▷ Both yaw change and area change are too large
            Track.AssocBB.area ← frameBoxHistory[k-1].area
            Track.AssocBB.yaw ← frameBoxHistory[k-1].yaw ▷ Restore previously stored box yaw and dimension
        else if deltaYawHist < `bbSmallThreshold` **and** deltaAreaHist < `bbSmallThreshold` **and** deltaWidth < 0.25 * frameBoxHistory[k-n].width **then** ▷ Yaw and Area does not change after n-time steps, but width of the box does not change much. Possible persistent self-occlusion
            continue ▷ update immediately
        end if
    end for
end procedure

Algorithm 7 Bounding Box Perspective Correction

```
1: procedure ShiftBB(currentTrack, frameBoxHistory)
2:     bbVelThreshold ← <adjustable parameter>
3:     if currentTrack.AssocBB.posX > 0 and currentTrack.relVel < -bbVelThreshold then
4:         shiftBoundingBoxUp(currentTrack.AssocBB) ▷ relVel is relative velocity of object to ego-vehicle
5:     else if currentTrack.AssocBB.posX < 0 and currentTrack.relVelX > bbVelThreshold then
6:         shiftBoundingBoxDown(currentTrack.AssocBB)
7:     end if
8: end procedure
```

Algorithm 8 Merge Over-segmentation

```
1: procedure MergeOverSegmentedBB(TrackList, clusterList, frameBoxHistory)
2:     k ← getCurrentTimeStep()                          ▷ current frame (time step)
3:     sharedPercThreshold ← <adjustable parameter>
4:     tolAreaThreshold ← <adjustable parameter>
5:     mergeCandidates ← <emptyList>
6:     for each cluster ∈ clusterList and box ∈ frameBoxHistory[k-1] do
7:         sharedArea ← computeSharedArea(cluster, box)
8:         if sharedArea > sharedPercThreshold * cluster.MinimumArea then
9:             mergeCandidate.add(cluster)
10:        end if
11:    end for
12:    for each cluster and predictedBox ∈mergeCandidates do
13:        mergedCluster ←mergeCluster(mergeCandidates.clusters)
14:        if mergedCluster.area > tolAreaThreshold * predictedBox.Area then
15:            clusterList.push(mergedCluster)
16:            clusterList.erase(mergeCandidates.clusters)
17:        end if
18:    end for
19: end procedure
```

Appendix C

Runtime Parameters

C-1 Detector Parameter

The detector parameters used in the provided algorithm is mainly derived based on the LIDAR real-world coordinate measurement of the object of interest, some parameter like kernel-size is obtained by manual tuning. The parameters are provided in Table C-1

Table C-1: Detector parameters

No.	Threshold variable	Unit	Value
1	$(m\|n)_{pgrid}$	-	$120.00, 80.00$
2	$r_{(min\|max)}$	m	$3.40, 120.00$
3	$T_{hmin\_(min\|max)}$	m	$-2.15, -1.40$
4	$h_{sensors}$	m	1.73
5	T_{slope}	m	0.25
6	T_{hdiff}	m	0.30
7	T_{flat}	m	0.20
8	$T_{consistent}$	m	0.30
9	s_{kernel}	m	1.00
10	h_{tol}	m	0.25
11	$(m\|n)_{clust}$	-	$100.00, 100.00$
12	$T_{height\_(min\|max)}$	m	$1.20, 2.60$
13	$T_{width\_(min\|max)}$	m	$0.50, 3.50$
14	$T_{length\_(min\|max)}$	m	$0.50, 14.00$
15	$T\_\{area\_(min\|max)\}$	m^2	20.00
16	$T_{ratio\_(min\|max)}$	-	$1.30, 5.00$
17	$T_{ratiocheck\_l\_(min)}$	m	3.00
18	$T_{pt\_per\_m3\_(min)}$	point/m^3	8

C-2 IMM-UKF-JPDAF Parameter

Table C-2: IMM-UKF-JPDAF

Par.	Value
α, β, κ	$0.0025, 2, 0$
P_G, P_D	0.8, 0.9
dT	0.103596489
$\mathbf{Q}_1$	$\texttt{diag}[1 \times 10^{-25}\,\mathrm{m/dT}, 1 \times 10^{-25}\,\mathrm{m/dT}, 0\,^\circ/\mathrm{dT}, 4 \times 10^{-3}\,\mathrm{m/dT^2}, 1 \times 10^{-15}\,^\circ/\mathrm{dT^2}]$
$\mathbf{Q}_2$	$\texttt{diag}[1 \times 10^{-25}\,\mathrm{m/dT}, 1 \times 10^{-25}\,\mathrm{m/dT}, 4 \times 10^{-2}\,^\circ/\mathrm{dT}, 4 \times 10^{-3}\,\mathrm{m/dT^2}, 1 \times 10^{-15}\,^\circ/\mathrm{dT^2}]$
$\mathbf{Q}_3$	$\texttt{diag}[1\,\mathrm{m/dT}, 1\,\mathrm{m/dT}, 1 \times 10^{-2}\,^\circ/\mathrm{dT}, 4 \times 10^{-1}\,\mathrm{m/dT^2}, 1 \times 10^{-15}\,^\circ/\mathrm{dT^2}]$
$\mathbf{R}$	$\texttt{diag}[1 \times 10^{-3}\,\mathrm{m}, 1 \times 10^{-3}\,\mathrm{m}]$
Π	$\begin{bmatrix} 0.90 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.9 \\ 0.05 & 0.05 & 0.09 \end{bmatrix}$
μ_{ip}	$\begin{bmatrix} 0.33 & 0.33 & 0.33 \end{bmatrix}$
x_0	$\begin{bmatrix} 0.0 & 0.0 & -0.1 & -0.83 & 1 \times 10^{-6} \end{bmatrix}$
$\mathbf{P}_0$	$\texttt{diag}[1 \times 10^{-4}\,\mathrm{m}, 3 \times 10^{-4}\,\mathrm{m}, 3 \times 10^{-4}\,^\circ, 3 \times 10^{-4}\,\mathrm{m\,s}, 3 \times 10^{-4}\,^\circ\,\mathrm{s^{-1}}]$

The reasoning behind the chosen parameters value are as follow:

1. The UKF parameter is initially set to literature[106, 107] default value for state estimation ($\alpha = 0.001$, $\beta = 2$. $\kappa = 0$), the default value works reasonably well as the filter converges to true estimate for most tracks within 2-3 time frame, indicating that the actual posterior pdf. is likely close to a Gaussian or uncertainties involved are small, we found by observation that for noisy detection result, increasing the α slightly to 0.0025 improves the filter convergence, this is expected since α influences the spread of sigma points, and thus more uncertainty can be accommodated.

2. JPDA probability of gating P_G is chosen to fit the requirement of having the gate large enough to actually cover the true measurement, since the random motion model has already big state covariance, value higher than 0.9 is not desired, otherwise there is higher chance that for a closely spaced objects, the measurement of multiple objects could fall into the same gate and corrupted the filter estimates. on the other hand, P_D is set to be 0.8 since complete occlusion can occasionally be found, also based on observation the detector is likely to yield a true positive albeit with a highly uncertain position.

3. The choice of Process Covariance Q_j: generally higher process covariance would enable each motion model to capture noisy movement and minimize the track lost. However, this would defeat the purpose of IMM to some extent. Therefore, the tuning starts from a very small value (1×10^{-25}) and incrementally increased until desired filter performance is found (this is done by inspecting highly manoeuvring object tracking result, at initial value lost track is consistent). The first two diagonals correspond to process noise in x and y position estimation respectively. Since they are observable states, the value is kept to be small and the uncertainty weighting is shifted to the measurement noise.

In addition, the Q_j for each model j is also based on the physical uncertainty of each motion model, for instance in Q_1 for the third diagonal that correspond to yaw is set to be relatively high since it is found that smaller value than 4×10^{-2} makes the filter to fail to track reasonably fast turning car. Meanwhile, the fourth diagonal correspond to velocity, 4×10^{-4} found to be a reasonable value for the filter to be still able to capture fast incoming car, while still provides smooth enough estimates for slow-moving pedestrian. The value of Q_3 is set to very large since noisy static object would change its dynamic unpredictably and IMM Mode 3 is expected to capture such dynamics.

4. The measurement covariance R diagonal corresponds directly to uncertainty in detector output of centre points: pretty high due to self-occlusion. This values also influence gating as it is performed based on the threshold-cutting the Mahalanobis distance between the predicted states and the measurement, a value too large would make the gating area to be too large also. Therefore, it is set to be high enough at 3×10^{-4} just before the gating started to be adversely affected (e.g. false positive association)

5. The matrix Π governs the transition probability between IMM motion model. The value follows[116] where frequent, agile mode changes thatare influenced by large off-diagonal is balanced by large diagonal values which makes the transition to be less dynamic but also smoother.

6. The vector μ_{ip} is simply the initial probability of a track to follow a motion model, it may help the filter converge faster, but since there is no a priori knowledge what kind of object (car/pedestrian/noise) is being tracked, the vector element is set to be equal to each other.

7. The initial value of the state x_0 is set to that of a car moving in parallel in the opposite direction to ego-vehicle with low-medium speed ($40\,\mathrm{m\,s^{-1}}$). These states value are observed on a large number of tracked object in the urban scenario and thus is chosen. However, the tracker will also initialize the x and y position based on last known measurements, along with velocity from finite distance difference divided by dT after starting from the second time frame, so this value is applicable for the first time frame.

8. The initial value of error covariance P_0 is set to be identical with the R matrix: pretty large. This is due to the fact that we do not have high confidence estimate of the initial states of the filter.

C-3 Box Tracker Parameter

Table C-3 lists all rule-based thresholds for box tracker.

Table C-3: Box Tracker Parameters

No.	Threshold variable	Unit	Value
1	historyGateLevel	-	0.01
2	distanceThreshold	-	0.25
3	velThreshold	$\mathrm{m\,s^{-1}}$	0.05
4	nStepBack	-	3.00
5	lifeTimeThreshold	dT	8
6	bbYawChangeThreshold	rad/dT	0.3
7	bbAreaChangeThreshold	$\mathrm{m^2/dT}$	0.2
8	bbSmallTol	-	0.01
9	sharedPercThreshold	%	85.00
10	tolAreaThreshold	%	20.00
11	dT	s	0.13

Appendix D

Motion Models

D-1 Constant Velocity (CV) Model

Constant Velocity (CV) Model represents motion behaviour of an object with nearly constant velocity and the object is assumed to have zero turning rate and thus is heading to the same bearing on all time steps. The system function is given as:

$$F_{CV_k} = \begin{bmatrix} x_{pos}k + v_k T \sin(\psi_k) \\ y_{pos}k + v_k T \cos(\psi_k) \\ 0 \\ v_k \\ \dot{\psi}_k \end{bmatrix} \tag{D-1}$$

D-2 Constant Turn-Rate Velocity (CTRV) Model

Turning motion is expected for road object in an urban situation. The assumptions are velocity is (nearly) constant and the turn rate is also constant. These assumptions are appropriate for both rectilinear motion with a uniform acceleration and a coordinated turn, which are two typical manoeuvres of urban vehicles[160]. The CTRV system function is given as:

$$F_{CTRV_k} = \begin{bmatrix} x_{pos}k + \frac{v_k}{\dot{\psi}_k}(-\sin(\psi_k) + \sin(T\psi_k + \psi_k)) \\ y_{pos}k + \frac{v_k}{\dot{\psi}_k}(\cos(\psi_k) - \cos(T\psi_k + \psi_k)) \\ T\psi_k + \psi_k \\ v_k \\ \dot{\psi}_k \end{bmatrix} \tag{D-2}$$

D-3 Random Motion (RM) Model

It is given that detection can likely be a noise (i.e. false positive). For instance, over-segmented cluster due to occlusion typically change position and shape on every time steps. Making it seems to move at random. Additionally, there are classes of static object such as a traffic sign that does not actually move at all. To capture these classes of object, the stationary model with relatively larger motion noise $\mathbf{Q}_k$ is used. The RM system function is thus identical with the state vectors:

$$F_{RM_k} = \begin{bmatrix} x_{pos}k \\ y_{pos}k \\ \psi_k \\ v_k \\ \dot{\psi}_k \end{bmatrix} \tag{D-3}$$

These three motion models are to be used in parallel using Interactive Multiple Model scheme which is detailed in Chapter 4.

Bibliography

[1] SAE International, "SAE, Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems, J3016," 2014.

[2] B. W. Smith, "Summary of SAE Levels of Driving Automation," 2015.

[3] R. Matthaei and M. Maurer, "Autonomous driving - A top-down-approach," *At-Automatisierungstechnik*, vol. 63, no. 3, pp. 155–167, 2015.

[4] J. Rasmussen, "Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-13, pp. 257–266, may 1983.

[5] A. R. HALE, J. STOOP, and J. HOMMELS, "Human error models as predictors of accident scenarios for designers in road transport systems," *Ergonomics*, vol. 33, pp. 1377–1387, oct 1990.

[6] J. A. Michon, "A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?," in *Hum. Behav. Traffic Saf.*, pp. 485–524, Boston, MA: Springer US, 1985.

[7] S. Ingle and M. Phute, "Tesla Autopilot : Semi Autonomous Driving , an Uptick for Future Autonomy," pp. 369–372, 2016.

[8] C. Thorpe and H. Durrant-Whyte, *The DARPA Urban Challenge*, vol. 56 of *Springer Tracts in Advanced Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[9] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *J. F. Robot.*, vol. 25, pp. 425–466, aug 2008.

[10] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *J. F. Robot.*, vol. 25, pp. 569–597, sep 2008.

[11] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A perception-driven autonomous urban vehicle," *J. F. Robot.*, vol. 25, pp. 727–774, oct 2008.

[12] T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J. R. Böhmer, "Stadtpilot: First fully autonomous test drives in urban traffic," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 919–924, 2011.

[13] P. Grisleri and I. Fedriga, *The BRAiVE autonomous ground vehicle platform*, vol. 7. IFAC, 2010.

[14] J. Rieken, R. Matthaei, and M. Maurer, "Benefits of using explicit ground-plane information for grid-based urban environment modeling," *Fusion*, pp. 2049–2056, 2015.

[15] J. Rieken, R. Matthaei, and M. Maurer, "Toward Perception-Driven Urban Environment Modeling for Automated Road Vehicles," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2015-Octob, pp. 731–738, 2015.

[16] K. Dietmayer, "Predicting of Machine Perception for Automated Driving," in *Auton. Driv.*, pp. 407–424, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[17] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth, "Semantic Stixels: Depth is not enough," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016-Augus, pp. 110–117, 2016.

[18] Velodyne, "Velodyne's HDL-64E: A High Definition Lidar Sensor for 3-D Applications. 2007," *White Pap.*, p. 7, 2007.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[20] K. C. J. Dietmayer, S. Reuter, and D. Nuss, *Handbook of Driver Assistance Systems*. 2014.

[21] A. Petrovskaya, A. Petrovskaya, S. Thrun, and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Auton. Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[22] M. Darms, C. R. Baker, P. E. Rybski, and C. Urmson, "Vehicle detection and tracking for the Urban Challenge Vehicle Detection and Tracking for the Urban Challenge," vol. 10, no. 3, pp. 57–67, 2008.

[23] J. Cesic, I. Markovic, S. Juric-Kavelj, and I. Petrovic, "Detection and tracking of dynamic objects using 3D laser range sensor on a mobile platform," *Informatics Control. Autom. Robot. (ICINCO), 2014 11th Int. Conf.*, vol. 2, pp. 110–119, 2014.

[24] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A Lidar and vision-based approach for pedestrian and vehicle detection and tracking," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 1044–1049, 2007.

[25] M. Himmelsbach and H. J. Wuensche, "Tracking and classification of arbitrary objects with bottom-up/top-down detection," *IEEE Intell. Veh. Symp. Proc.*, pp. 577–582, 2012.

[26] S. Gidel, P. Checchin, C. Blanc, T. Chateau, and L. Trassoudaine, "Pedestrian Detection and Tracking in an Urban Environment Using a Multilayer Laser Scanner," *Intell. Transp. Syst. IEEE Trans.*, vol. 11, no. 3, pp. 579–588, 2010.

[27] N. Wojke and M. Haselich, "Moving Vehicle Detection and Tracking in Unstructured Environments," *2012 IEEE Int. Conf. Robot. Autom.*, pp. 3082–3087, 2012.

[28] T. Chen, B. Dai, D. Liu, H. Fu, J. Song, and C. Wei, "Likelihood-Field-Model-Based Vehicle Pose Estimation with Velodyne," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2015-Octob, pp. 296–302, 2015.

[29] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Syst. Mag.*, vol. 29, no. 6, pp. 82–100, 2009.

[30] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 5–18, 2004.

[31] D. Mušicki and R. Evans, "Joint Integrated Probabilistic Data Association - JIPDA," *Proc. Fusion 2002*, no. I, pp. 1120–1125, 2002.

[32] A. Ess, K. Schindler, B. Leibe, and L. V. Gool, "Improved Multi-Person Tracking with Active Occlusion Handling," *ICRA 2009 Work.*, no. May, 2009.

[33] S. Song, Z. Xiang, and J. Liu, "Object Tracking with 3D LIDAR via Multi-Task Sparse Learning," no. 61071219, pp. 2603–2608, 2015.

[34] J. Yan, D. Chen, H. Myeong, T. Shiratori, and Y. Ma, "Automatic extraction of moving objects from image and LIDAR sequences," *Proc. - 2014 Int. Conf. 3D Vision, 3DV 2014*, pp. 673–680, 2015.

[35] T. Ogawa and G. Wanielik, "Track-Before-Detect approach on LIDAR signal processing for low SNR target detection," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016-Augus, no. Iv, pp. 676–682, 2016.

[36] M. Schreier, V. Willert, and J. Adamy, "Compact Representation of Dynamic Driving Environments for ADAS by Parametric Free Space and Dynamic Object Maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 367–384, 2016.

[37] L. Zhang, Q. Li, M. Li, Q. Mao, and A. Nüchter, "Multiple Vehicle-like Target Tracking Based on the Velodyne LiDARâŃĘ," no. 2005, 2011.

[38] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, "Multi-Target Tracking using a 3D-Lidar sensor for autonomous vehicles," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, no. Itsc, pp. 881–886, 2013.

[39] W. Luo, J. Xing, X. Zhang, X. Zhao, and T.-K. Kim, "Multiple Object Tracking: A Literature Review," *arXiv, 1-39. Retrieved from http//arxiv.org/abs/1409.7618*, vol. V, no. 212, 2014.

[40] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," *2011 IEEE Int. Conf. Robot. Autom.*, pp. 2798–2805, 2011.

[41] R. MacLachlan and C. Mertz, "Tracking of Moving Objects from a Moving Vehicle Using a Scanning Laser Rangefinder," *2006 IEEE Intell. Transp. Syst. Conf.*, pp. 301–306, 2006.

[42] R. Matthaei, B. Lichte, and M. Maurer, "Robust grid-based road detection for ADAS and autonomous vehicles in urban environments," *16th Int. Conf. Inf. Fusion*, pp. 938–944, 2013.

[43] J. Rieken, R. Matthaei, and M. Maurer, "Toward Perception-Driven Urban Environment Modeling for Automated Road Vehicles," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2015-Octob, pp. 731–738, 2015.

[44] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1773–1795, 2013.

[45] Z. Luo, S. Habibi, and M. Mohrenschildt, "LiDAR Based Real Time Multiple Vehicle Detection and Tracking," vol. 10, no. 6, pp. 1083–1090, 2016.

[46] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1773–1795, 2013.

[47] K. Kovacic, E. Ivanjko, and H. Gold, "Computer Vision Systems in Road Vehicles: A Review," *Proc. Croat. Comput. Vis. Work.*, pp. 25–30, 2013.

[48] S. Sivaraman and M. M. Trivedi, "A review of recent developments in vision-based vehicle detection," *Proc. IEEE Intell. Veh. Symp.*, no. Iv, pp. 310–315, 2013.

[49] S. Piao, T. Sutjaritvorakul, and K. Berns, "Compact Data Association in Multiple Object Tracking: Pedestrian Tracking on Mobile Vehicle as Case Study," *9th IFAC Symp. Intell. Auton. Veh.*, vol. 49, no. 15, pp. 175–180, 2016.

[50] S. Han, X. Wang, L. Xu, H. Sun, and N. Zheng, "Frontal object perception for Intelligent Vehicles based on radar and camera fusion," *Chinese Control Conf. CCC*, vol. 2016-Augus, pp. 4003–4008, 2016.

[51] B. Tian, Y. Li, B. Li, and D. Wen, "Rear-view vehicle detection and tracking by combining multiple parts for complex Urban surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 597–606, 2014.

[52] A. Börcs, B. Nagy, and C. Benedek, "Dynamic 3D Environment Perception and Reconstruction Using a Mobile Rotating Multi-beam Lidar Scanner," pp. 153–180, 2015.

[53] R. H. Rasshofer and K. Gresser, "Automotive radar and lidar systems for next generation driver assistance functions," *Adv. Radio Sci.*, vol. 3, pp. 205–209, 2005.

[54] S. Schneider, M. Himmelsbach, T. Luettel, and H. J. Wuensche, "Fusing vision and LIDAR - Synchronization, correction and occlusion reasoning," *IEEE Intell. Veh. Symp. Proc.*, pp. 388–393, 2010.

[55] G. Zhao, X. Xiao, J. Yuan, and G. W. Ng, "Fusion of 3D-LIDAR and camera data for scene parsing," *J. Vis. Commun. Image Represent.*, vol. 25, no. 1, pp. 165–183, 2014.

[56] Y. J. Li, Z. Zhang, D. Luo, and G. Meng, "Multi-sensor environmental perception and information fusion for vehicle safety," pp. 2838–2841, aug 2015.

[57] R. O. Chavez-garcia and O. Aycard, "Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking," *Ieee Trans. Intell. Transp. Syst.*, vol. PP, no. 99, pp. 1–10, 2015.

[58] T.-D. Vu, O. Aycard, and F. Tango, "Object Perception for Intelligent Vehicle Applications: A Multi-Sensor Fusion Approach," *Iv '14*, no. Iv, pp. 774–780, 2014.

[59] S. Lange, F. Ulbrich, and D. Goehring, "Online vehicle detection using deep neural networks and lidar based preselected image patches," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016-Augus, no. Iv, pp. 954–959, 2016.

[60] H. Gotzig and G. Geduld, "Automotive LIDAR," in *Handb. Driv. Assist. Syst.*, pp. 1–20, Cham: Springer International Publishing, 2015.

[61] Velodyne, "Datasheet for Velodyne HDL-64E S2," *Velodyne Morgan Hill, CA, USA. Available online http//www. velodyne. com/lidar/products/brochure/HDL64E% 20S2% 20datasheet_2010_lowres. pdf (accessed 22 January 2010).*

[62] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Rob. Auton. Syst.*, vol. 83, pp. 299–311, 2016.

[63] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data.," *2008 IEEE Int. Conf. Robot. Autom.*, pp. 4043–4048, 2008.

[64] Z. Liu, D. Liu, and T. Chen, "Vehicle detection and tracking with 2D laser range finders," *Proc. 2013 6th Int. Congr. Image Signal Process. CISP 2013*, vol. 2, no. 91220301, pp. 1006–1013, 2013.

[65] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications.* 2011.

[66] M. Schreier, V. Willert, and J. Adamy, "Grid mapping in dynamic road environments: Classification of dynamic cell hypothesis via tracking," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3995–4002, 2014.

[67] L. Linsen, "Point cloud representation," *Univ. Karlsruhe, Ger. Tech. Report, Fac. Informatics*, pp. 1–18, 2001.

[68] X. Chen and Y. Zhu, "3D Object Proposals for Accurate Object Class Detection," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2015.

[69] F. Zhang, D. Clarke, and A. Knoll, "Vehicle detection based on LiDAR and camera fusion," *2014 17th IEEE Int. Conf. Intell. Transp. Syst. ITSC 2014*, pp. 1620–1625, 2014.

[70] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," pp. 1–12, 2016.

[71] P. Xu, F. Davoine, J.-B. Bordes, H. Zhao, and T. Denoeux, "Information Fusion on Oversegmented Images: An Application for Urban Scene Understanding," *Mva*, pp. 2–6, 2013.

[72] M. Allodi, A. Broggi, D. Giaquinto, M. Patander, and A. Prioletti, "Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016-Augus, no. Iv, pp. 648–653, 2016.

[73] a. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[74] C. Coue, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application," *Int. J. Rob. Res.*, vol. 25, no. 1, pp. 19–30, 2006.

[75] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and Tracking in dynamic environments using a uniform evidential environment representation," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 6090–6095, 2014.

[76] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1331–1342, 2011.

[77] J. Cheng, Z. Xiang, T. Cao, and J. Liu, "Robust vehicle detection using 3D Lidar under complex urban environment," *2014 IEEE Int. Conf. Robot. Autom.*, pp. 691–696, 2014.

[78] C. H. Rodríguez-Garavito, A. Ponz, F. García, D. Martín, A. de la Escalera, and J. M. Armingol, "Automatic laser and camera extrinsic calibration for data fusion using road plane," pp. 1–6, 2014.

[79] M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Real-time object classification in 3D point clouds using point feature histograms," *2009 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 994–1000, 2009.

[80] D. O. Rubio, A. Lenskiy, and J. H. Ryu, "Connected components for a fast and robust 2D lidar data segmentation," *Proc. - Asia Model. Symp. 2013 7th Asia Int. Conf. Math. Model. Comput. Simulation, AMS 2013*, no. September 2015, pp. 160–165, 2013.

[81] M. Schreier, V. Willert, and J. Adamy, "Compact Representation of Dynamic Driving Environments for ADAS by Parametric Free Space and Dynamic Object Maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 367–384, 2016.

[82] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon, "Fast Multiple Objects Detection and Tracking Fusing Color Camera and 3D LIDAR for Intelligent Vehicles," pp. 234–239, 2016.

[83] M. Kusenbach, M. Himmelsbach, and H. J. Wuensche, "A new geometric 3D LiDAR feature for model creation and classification of moving objects," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016-Augus, no. Iv, pp. 272–278, 2016.

[84] B. Barrois, S. Hristova, C. Wohler, F. Kummert, C. C. C. Hermes, C. Wöhler, F. Kummert, and C. C. C. Hermes, "3D pose estimation of vehicles using a stereo camera," *Proc. IEEE Intell. Veh. Symp.*, pp. 267–272, 2009.

[85] D. Morris, R. Hoffman, and P. Haley, "A View-Dependent Adaptive Matched Filter for Ladar-Based Vehicle Tracking," 2009.

[86] M. Lehtomäki, A. Jaakkola, J. Hyyppä, J. Lampinen, H. Kaartinen, A. Kukko, E. Puttonen, and H. Hyyppä, "Mobile Laser Scanning Point Clouds in a Road Environment," pp. 2–3, 2015.

[87] F. Roos, D. Kellner, J. Dickmann, and C. Waldschmidt, "Reliable Orientation Estimation of Vehicles in High-Resolution Radar Images," *IEEE Trans. Microw. Theory Tech.*, vol. 64, no. 9, pp. 2986–2993, 2016.

[88] O. Aycard, "Laser-based detection and tracking moving objects using data-driven Markov chain Monte Carlo," *2009 IEEE Int. Conf. Robot. Autom.*, pp. 3800–3806, 2009.

[89] F. Nashashibi and A. Bargeton, "Laser-based vehicles tracking and classification using occlusion reasoning and confidence estimation," *IEEE Intell. Veh. Symp. Proc.*, no. 1, pp. 847–852, 2008.

[90] M. Darms, P. Rybski, and C. Urmson, "Classification and tracking of dynamic objects with multiple sensors for autonomous driving in Urban environments," *IEEE Intell. Veh. Symp. Proc.*, pp. 1197–1202, 2008.

[91] Y. Ye, L. Fu, and B. Li, "Object Detection and Tracking Using Multi-layer Laser for Autonomous Urban Driving," 2016.

[92] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe, D. Duggins, and J. Gowdy, "Moving object detection with laser scanners," *J. F. Robot.*, vol. 30, pp. 17–43, jan 2013.

[93] C. Tay, C. Pradalier, and C. Laugier, "Vehicle Detection And Car Park Mapping Using Laser Scanner," *Proc. IEEE-RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1761–1767, 2005.

[94] W. Zhang, Y. Guo, M. Lu, J. Zhang, A. A. Based, and D. Training, "Ground Target Detection in LiDAR Point Clouds using AdaBoost," *3Dv*, pp. 22–26, 2015.

[95] M. Braun, Q. Rao, Y. Wang, and F. Flohr, "Pose-RCNN : Joint Object Detection and Pose Estimation Using 3D Object Proposals," pp. 1546–1551, 2016.

[96] M. Himmelsbach, F. von Hundelshausen, and H. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," *Iv*, pp. 560–565, 2010.

[97] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans, *Fundamentals of object tracking.* 2011.

[98] R. Behringer, S. Sundareswaran, B. Gregory, R. Elsley, B. Addison, W. Guthmiller, R. Daily, and D. Bevly, "The DARPA grand challenge - development of an autonomous vehicle," *Intell. Veh. Symp. 2004 IEEE*, pp. 226–231, 2004.

[99] W. Xiao, B. Vallet, K. Schindler, and N. Paparoditis, "SIMULTANEOUS DETECTION AND TRACKING OF PEDESTRIAN FROM PANORAMIC LASER SCANNING DATA," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. III-3, pp. 295–302, jun 2016.

[100] K. Granstrom and M. Baum, "Extended Object Tracking: Introduction, Overview and Applications," pp. 1–17, 2016.

[101] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles," pp. 1–27, 2016.

[102] A. Zlocki, *Automated Driving*, vol. 2416. 2014.

[103] G. E. P. Box, "Science and Statistics," *J. Am. Stat. Assoc.*, vol. 71, no. 356, pp. 791–799, 1976.

[104] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, no. 1, p. 35, 1960.

[105] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association.* Mathematics in Science and Engineering Series, Academic Press, 1988.

[106] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proc. IEEE*, vol. 92, pp. 401–422, mar 2004.

[107] E. a. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *Technology*, vol. v, pp. 153–158, 2000.

[108] S. Thrun, "Particle Filters in Robotics," *Proc. Uncertain. AI*, vol. 1, pp. 511–518, 2002.

[109] B. Fortin, J. Noyer, and R. Lherbier, "A particle filtering approach for joint vehicular detection and tracking in lidar data," *2012 IEEE Int. Instrum. Meas. Technol. Conf. Proc.*, no. 3, pp. 391–396, 2012.

[110] N. Morales, J. Toledo, L. Acosta, and J. Sanchez-Medina, "A Combined Voxel and Particle Filter-Based Approach for Fast Obstacle Detection and Tracking in Automotive Applications," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–11, 2016.

[111] X. Li and V. Jilkov, "Survey of maneuvering target tracking: dynamic models," *Proc. SPIE Conf. Signal Data Process. Small Targets 2000*, no. April, pp. 212–235, 2000.

[112] Y.-s. Kim, "A Tracking Algorithm for Autonomous Navigation of AGVs in an Automated Container," vol. 19, no. 1, pp. 72–86, 2005.

[113] Y.-s. Kim and K.-S. Hong, "An IMM algorithm for tracking maneuvering vehicles in an adaptive cruise control environment," *Int. J. Control Autom. Syst.*, vol. 2, no. 3, pp. 310–318, 2004.

[114] N. Kaempchen and K. Dietmayer, "IMM Vehicle Tracking for Traffic Jam Situations on Highways," *Proc. 7th Intl. Conf. Multisens. Inf. Fusion*, vol. 1, no. 2, pp. 868–875, 2004.

[115] X. Wang, S. Challa, and R. Evans, "Gating techniques for maneuvering target tracking in clutter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 3, pp. 1087–1097, 2002.

[116] M. Schreier, *Bayesian environment representation, prediction, and criticality assessment for driver assistance systems*. PhD thesis, 2017.

[117] M. de Feo, A. Graziano, R. Miglioli, and A. Farina, "IMMJPDA versus MHT and Kalman filter with NN correlation: performance comparison," *IEE Proc. - Radar, Sonar Navig.*, vol. 144, no. 2, p. 49, 1997.

[118] Y. Bar-Shalom and X. R. Li, *Multitarget-multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995.

[119] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House radar library, Artech House, 1999.

[120] The QT Company, "Qt for Application Development," 2017.

[121] Nvidia Corp., "Autonomous Car Development Platform from NVIDIA DRIVE PX2."

[122] R. B. Rusu and S. Cousins, "3D is here: point cloud library," *IEEE Int. Conf. Robot. Autom.*, pp. 1 – 4, 2011.

[123] G. Bradski, "OpenCV Library," *Dr. Dobb's J. Softw. Tools*.

[124] G. Guennebaud, B. Jacob, and Others, "Eigen Library." http://eigen.tuxfamily.org, 2010.

[125] The MathWorks, "MATLAB and Simulink product families release 2016b."

[126] R. W. Johnson, J. L. Evans, P. Jacobsen, J. R. Thompson, and M. Christopher, "The changing automotive environment: High-temperature electronics," *IEEE Trans. Electron. Packag. Manuf.*, vol. 27, no. 3, pp. 164–176, 2004.

[127] S. Aksoy, "Binary Image Analysis," *Text*.

[128] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Commun. ACM*, vol. 18, pp. 409–413, jul 1975.

[129] A. H. A. Rahman, H. Zamzuri, S. A. Mazlan, and M. A. A. Rahman, "Model-Based Detection and Tracking of Single Moving Object Using Laser Range Finder," *2014 5th Int. Conf. Intell. Syst. Model. Simul.*, pp. 556–561, 2014.

[130] B. Kim, D. Kim, S. Park, Y. Jung, and K. Yi, "Automated Complex Urban Driving based on Enhanced Environment Representation with GPS/map, Radar, Lidar and Vision," *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 190–195, 2016.

[131] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2D lidar," *Int. J. Rob. Res.*, vol. 34, no. 7, pp. 1039–1063, 2015.

[132] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, USA: John Wiley & Sons, Inc., 2001.

[133] H. A. P. Blom and E. A. Bloem, "Interacting Multiple Model Joint Probabilistic Data Association avoiding track coalescence," *Aerospace*, 2002.

[134] R. Rajamani, *Vehicle Dynamics and Control*. Mechanical Engineering Series, Boston, MA: Springer US, 2012.

[135] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Eurasip J. Image Video Process.*, vol. 2008, 2008.

[136] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors," *Int. J. Comput. Vis.*, vol. 75, no. 2, pp. 247–266, 2007.

[137] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *Int. J. Comput. Vis.*, vol. 111, pp. 98–136, jan 2015.

[138] K. Bussemaker, "Sensing requirements for an automated vehicle for high- way and rural environments MSc thesis," 2011.

[139] A. Geiger, "The KITTI Benchmark Suite: Object Tracking Evaluation 2012," 2012.

[140] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOT17 Results," 2017.

[141] W. Zheng, A. Thangali, S. Sclaroff, and M. Betke, "Coupling detection and data association for multiple object tracking," *Comput. Vis. Pattern Recognit. (CVPR), 2012 IEEE Conf.*, pp. 1948–1955, 2012.

[142] X. Shi, Z. Yang, and J. Chen, "Modified Joint Probabilistic Data Association," *Ad-hoc Sens. Netw. Symp.*, no. December, pp. 6615–6620, 2015.

[143] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," 2015.

[144] W. Xiao, B. Vallet, K. Schindler, and N. Paparoditis, "Simultaneous Detection and Tracking of Pedestrian From Panoramic Laser Scanning Data," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. III-3, no. July, pp. 295–302, 2016.

[145] L. Spinello, M. Luber, and K. O. Arras, "Tracking people in 3D using a bottom-up top-down detector," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1304–1310, 2011.

[146] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, "Generative object detection and tracking in 3D range data," *2012 IEEE Int. Conf. Robot. Autom.*, pp. 3075–3081, 2012.

[147] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track : Online Multi-Object Tracking by Decision Making," pp. 4705–4713.

[148] J. H. Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon, "Online Multi-object Tracking via Structural Constraint Event Aggregation," *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1392–1400, 2016.

[149] A. Osep, W. Mehner, M. Mathias, and B. Leibe, "Combined image- and world-space tracking in traffic scenes," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1988–1995, 2017.

[150] E. a. Wan, R. van der Merwe, and A. Nelson, "Dual estimation and the unscented transformation," *Nips*, no. January 2000, pp. 1–7, 2000.

[151] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," *Inf. Fusion, 2008 11th Int. Conf.*, no. 1, pp. 1–6, 2008.

[152] G. Zhai, H. Meng, and X. Wang, "A constant speed changing rate and constant turn rate model for maneuvering target tracking," *Sensors (Switzerland)*, vol. 14, no. 3, pp. 5239–5253, 2014.

[153] The Linux Foundation, "Automotive Grade Linux Specification," 2015.

[154] M. Bojarski, "Explaining How End-to-End Deep Learning Steers a Self-Driving Car | Parallel Forall," 2017.

[155] H. Tehrani Niknejad, K. Takahashi, S. Mita, and D. McAllester, "Embedded multi-sensors objects detection and tracking for urban autonomous driving," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 1128–1135, 2011.

[156] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo," pp. 2181–2188, 2016.

[157] R. C. Luo and C. C. Lai, "Multisensor fusion-based concurrent environment mapping and moving object detection for intelligent service robotics," *IEEE Trans. Ind. Electron.*, vol. 61, no. 8, pp. 4043–4051, 2014.

[158] H. Cho, Y.-w. W. Seo, B. V. K. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," *2014 IEEE Int. Conf. Robot. Autom.*, pp. 1836–1843, may 2014.

[159] S. Budzan and J. Kasprzyk, "Fusion of 3D laser scanner and depth images for obstacle recognition in mobile applications," *Opt. Lasers Eng.*, vol. 77, pp. 230–240, 2016.

[160] G. Zhai, H. Meng, and X. Wang, "A constant speed changing rate and constant turn rate model for maneuvering target tracking," *Sensors (Switzerland)*, vol. 14, no. 3, pp. 5239–5253, 2014.

Glossary

List of Acronyms

CV Constant Velocity

CTRV Constant Turn Rate and Velocity

DA Data Association

EKF Extended Kalman Filter

GT Ground Truth

IDSW ID Switch

FM Fragmentation Number

FN False Negative

FP False Positive

IMM-UK-JPDAF Interacting Multiple Model Unscented Kalman Joint Probabilistic Association Filter

JPDAF Joint Probabilistic Data Association Filter

KF Kalman Filter

LIDAR Light Detection And Ranging

MAR Minimum Area Rectangle

MOT Multi Object Tracking

MOTA Multi Object Accuracy

MOTP Multi Object Precision

MHT Multi Hypothesis Tracking

ML	Mostly Lost
MT	Mostly Tracked
NN	Nearest Neighbour
PF	Particle Filter
TP	True Positive
RM	Random Motion
UKF	Unscented Kalman Filter

List of Symbols

α	UKF sigma point scaling parameter
β	UKF higher order scaling parameter
β_i	PDA association probabilities
χ	Matrix of sigma points in UKF
κ	UKF scalar tuning parameter
μ	IMM mode probability
ψ	Target object yaw
$\mathbf{F}$	System function
$\mathbf{H}$	Measurement equation
$\mathbf{K}$	Kalman Gain
$\mathbf{k}$	Time step
$\mathbf{P}$	Error covariance matrix
$\mathbf{Q}$	Process noise covariance matrix
$\mathbf{R}$	Measurement noise covariance matrix
$\mathbf{S}$	Innovation covariance matrix
$\mathbf{u}$	System input vector
$\mathbf{v}$	Measurement noise vector
$\mathbf{w}$	Process noise vector
$\mathbf{x}$	State vector
$\mathbf{z}$	Measurement vector
p	IMM mode transition probability
v	PDA innovation vector or target object velocity
W	UKF sigma points weight
x	x-position in cartesian coordinate
y	y-position in cartesian coordinate
z	z-position in cartesian coordinate

Index