

Spring 4-2021

A Comprehensive Mapping and Real-World Evaluation of Multi-Object Tracking on Automated Vehicles

Alexander Bassett

Embry-Riddle Aeronautical University, basseta4@my.erau.edu

Follow this and additional works at: <https://commons.erau.edu/edt>



Part of the [Robotics Commons](#), and the [Theory and Algorithms Commons](#)

Scholarly Commons Citation

Bassett, Alexander, "A Comprehensive Mapping and Real-World Evaluation of Multi-Object Tracking on Automated Vehicles" (2021). *PhD Dissertations and Master's Theses*. 579.

<https://commons.erau.edu/edt/579>

This Thesis - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in PhD Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

A COMPREHENSIVE MAPPING AND REAL-WORLD EVALUATION OF MULTI-
OBJECT TRACKING ON AUTOMATED VEHICLES

by

Alexander C. Bassett

A Thesis Submitted to the College of Engineering Department of Electrical Engineering
and Computer Science in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical and Computer Engineering

Embry-Riddle Aeronautical University

Daytona Beach, Florida

April 2021

A COMPREHENSIVE MAPPING AND REAL-WORLD EVALUATION OF MULTI-OBJECT TRACKING ON AUTOMATED VEHICLES

by

Alexander C. Bassett

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. M. Ilhan Akbas, Professor, Daytona Beach Campus, and Thesis Committee Members Dr. Patrick Currier, Professor, Daytona Beach Campus, and Dr. Eric Coyle, Professor, Daytona Beach Campus, and has been approved by the Thesis Committee. It was submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

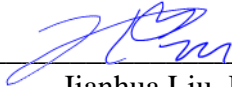
Thesis Review Committee:



Dr. M. Ilhan Akbas, Ph.D.
Committee Chair



Dr. Patrick Currier, Ph.D.
Committee Member

 4/23/2021

Jianhua Liu, Ph.D.
Graduate Program Chair,
Electrical and Computer Engineering



Dr. Eric Coyle, Ph.D.
Committee Member

Timothy Wilson, Ph.D.
Department Chair,
Electrical Engineering & Computer Science

Maj Mirmirani, Ph.D.
Dean, College of Engineering

Christopher Grant, Ph.D.
Associate Vice President of Academics

Date

Acknowledgments

I would like to thank the EcoCAR Mobility Challenge for funding my graduate school and continually challenging me throughout the years. Through this program, I have grown as an engineer and a leader and made fantastic friends along the way.

I would like to thank my committee chair, Dr. Ilhan Akbas, who has supported me continually throughout this thesis and guided me to become a better researcher.

I would like to thank my co-advisor, Dr. Patrick Currier, for guiding me to become a better engineer, writer, technical communicator, and leader throughout my entire graduate school education.

I would like to thank my committee member, Dr. Eric Coyle, for introducing me to Kalman filters in his course. This sparked my interest and ultimately led to this thesis.

I would like to thank the members of the ERAU EcoCAR Mobility Challenge team for aiding me in collecting data for this thesis. Specifically, I would like to thank Brandon Carrier, Devon Vail, and Devin Currie for their help with preparing the vehicle, testing components, and collecting data.

I would like to thank all my friends, family, and my girlfriend, Cassy, for their continued moral support throughout my education, especially during the creation of this thesis.

Table of Contents

Acknowledgments.....	iii
Table of Contents.....	iv
List of Tables.....	v
Abstract.....	1
1 Introduction.....	2
1.1 Motivation.....	4
1.2 Contributions.....	6
2 Related Work.....	7
2.1 Related Review/Surveys.....	7
2.2 Related Implementation Work.....	8
3 Comprehensive Mapping of MOT Field for Autonomous Vehicles.....	9
3.1 Comparison and Overview.....	10
3.2 Common AV Perception Sensor Types.....	12
3.2.1 Camera.....	12
3.2.2 Radar.....	14
3.2.3 Lidar.....	15
3.3 MOT Definitions and Considerations.....	15
3.3.1 Tracking Categories and Definitions.....	15
3.3.2 Shape Modeling.....	17
3.3.3 Dynamics Modelling.....	19
3.3.4 Object State.....	20
3.3.5 State Estimation.....	20
3.4 Conventional MOT Breakdown and Subcomponent Description.....	26
3.4.1 Sensor Processing to Detections.....	27
3.4.2 Track Gating.....	27
3.4.3 Observation-to-Track Data Association.....	28
3.4.4 Track Maintenance.....	33
3.5 Extended Object Data and Extended Object Tracking (EOT).....	34
3.5.1 Extended Object Tracking Overview.....	34
3.5.2 EOT Data Association.....	35
3.5.3 Extended Object Tracker Examples.....	35

3.5.4	POT to EOT	36
3.6	MOTs with Multiple Sensors	39
3.6.1	Track Fusion Methods	41
3.7	Metrics.....	41
3.7.1	Error Metrics.....	42
3.7.2	Assignment Metrics	42
3.7.3	OSPA	42
3.8	Chapter 3 Summary.....	43
4	Real-World Evaluation of an MOT Software Toolset on an AV	44
4.1	Vehicle Configuration	44
4.1.1	Bosch Radar	46
4.1.2	Intel Mobileye 6 Vision System	47
4.1.3	Ground Truth	48
4.2	Software Configuration.....	51
4.2.1	Software Toolsets.....	51
4.2.2	MathWorks Configuration.....	52
4.3	Issues and Considerations	54
4.3.1	Ground Truth Considerations	54
4.3.2	Time Alignment.....	54
4.3.3	Point Target Tracking Limitation	55
4.3.4	Real-World Complications	55
4.4	Experiment	56
4.4.1	Test Setup.....	56
4.4.2	Detection Evaluation.....	57
4.4.3	GNN Central Level Tracker Evaluation	65
4.5	Tracking Toolset Review	72
5	Conclusions and Future Work	73
	References.....	74

List of Tables

Table 1: Human vs AV Sensor Capabilities [3].....	2
Table 2: Max Speed Allowing For Minimum Stopping Distance [3]	3

Table 3: Example Association Matrix for Figure 9	30
Table 4: Example Association Matrix for Figure 9, Including Validation Gates	30
Table 5: Tracker Categorization	39
Table 6: Data Given From Sensors	53
Table 7: Detection Position RMSE.....	60
Table 8: Longitudinal Velocity RMSE.....	63
Table 9: Detection Lateral Velocity RMSE.....	65
Table 10: Tracker Position RMSE Comparison	67
Table 11: Tracker Longitudinal Velocity RMSE.....	70
Table 12: Tracker Lateral Velocity RMSE Comparison	72

List of Figures

Figure 1: Rolling vs Global Shutter [40]	13
Figure 2: General Motors Enhanced Night Vision [42].....	14
Figure 3: Relevant Tracking Categories	17
Figure 4: Relevant Shape Models	19
Figure 5: Kalman Filter Covariance Overlap [45].....	22
Figure 6: Illustration of Sensor Covariances. Dashed blue ellipse: vision system. Solid red ellipse: radar system.	23
Figure 7: MTT Breakdown (based on [9]).....	27
Figure 8: Validation Gates. A and B - detections associated with known tracks. C - unassociated detection representing a real object. D - unassociated detection representing noise.	28
Figure 9: Detection A can only be associated with the left vehicle. Detection B could represent either of the vehicles since it is within both of their gates. Detection C represents unassociated noise.	29
Figure 10. Incorrect Clustering of Sensor Data [54].....	37
Figure 11: Covariance ellipses of A (dotted red ellipse), B (dotted blue ellipse), and Covariance Intersection of both (green ellipse) [58].....	41
Figure 12: ERAU Test Vehicle.....	45
Figure 13: Component Layout	46
Figure 14. Bosch Radar [54]; Only The Circled Radar Is Used In This Thesis	47
Figure 15: Vision Setup; Mobileye 6 (Center) and Diagnostic Camera (Left).....	48
Figure 16: OxTS Ground Truth Unit	49
Figure 17: OxTS Ground Truth Data Before Corrections (above) and after (below)	50
Figure 18: MOT Pipeline	52
Figure 19: Scenario Description	57
Figure 20: Position Error from Detections.....	59
Figure 21: Radar Detections	61
Figure 22: Longitudinal Velocity Error From Detections	62
Figure 23: Lateral Velocity Error from Radar	64
Figure 24: Position Error from Central Level Tracker GNN.....	66
Figure 25: Tracker at Start	68
Figure 26: Longitudinal Velocity Error from Central Level Tracker GNN	69
Figure 27: Lateral Velocity Error from Central Level Tracker GNN.....	71

Abstract

Multi-Object Tracking (MOT) is a field critical to Automated Vehicle (AV) perception systems. However, it is large, complex, spans research fields, and lacks resources for integration with real sensors and implementation on AVs. Factors such those make it difficult for new researchers and practitioners to enter the field.

This thesis presents two main contributions: 1) a comprehensive mapping for the field of Multi-Object Trackers (MOTs) with a specific focus towards Automated Vehicles (AVs) and 2) a real-world evaluation of an MOT developed and tuned using COTS (Commercial Off-The-Shelf) software toolsets. The first contribution aims to give a comprehensive overview of MOTs and various MOT subfields for AVs that have not been presented as wholistically in other papers. The second contribution aims to illustrate some of the benefits of using a COTS MOT toolset and some of the difficulties associated with using real-world data. This MOT performed accurate state estimation of a target vehicle through the tracking and fusion of data from a radar and vision sensor using a Central-Level Track Processing approach and a Global Nearest Neighbors assignment algorithm. It had an 0.44 m positional Root Mean Squared Error (RMSE) over a 40 m approach test.

It is the authors' hope that this work provides an overview of the MOT field that will help new researchers and practitioners enter the field. Additionally, the author hopes that the evaluation section illustrates some difficulties of using real-world data and provides a good pathway for developing and deploying MOTs from software toolsets to Automated Vehicles.

1 Introduction

Vehicular accidents are one of the leading causes of death in the USA, with nearly 40k being killed annually [1]. One of the goals of Automated/Autonomous Vehicles (AVs¹) is to increase safety and lower the rates of injury and death from vehicular accidents. This is reflected in the description of many AV companies and even recognized in established automakers such as General Motors’ new motto “zero crashes, zero emissions, and zero congestion” [2]. It is not difficult to imagine that AVs have the potential to operate safer than humans. While humans have great sensory organs, they are limited to that FOV, number, range, and modality, whereas AVs can take advantage of combining multiple sensors for increased FOV, increasing the number of sensors as desired, using longer range sensors, and observing modalities imperceptible to humans, as shown in Table 1.

Table 1: Human vs AV Sensor Capabilities [3]

Performance aspect	Human	AV			CV	CAV
		<i>Radar</i>	<i>Lidar</i>	<i>Camera</i>	<i>DSRC</i>	<i>CV+AV</i>
Object detection	Good	Good	Good	Fair	n/a	Good
Object classification	Good	Poor	Fair	Good	n/a	Good
Distance estimation	Fair	Good	Good	Fair	Good	Good
Edge detection	Good	Poor	Good	Good	n/a	Good
Lane tracking	Good	Poor	Poor	Good	n/a	Good
Visibility range	Good	Good	Fair	Fair	Good	Good
Poor weather performance	Fair	Good	Fair	Poor	Good	Good
Dark or low illumination performance	Poor	Good	Good	Fair	n/a	Good
Ability to communicate with other traffic and infrastructure	Poor	n/a	n/a	n/a	Good	Good

¹ For the context of this thesis, Automated Vehicle and Autonomous Vehicle may be used interchangeably.

Also, AVs can recognize and react faster than humans [3]. Together, these factors result in AVs being able to outperform humans. This is shown in Table 2 as a comparison of the maximum allowed speed of a vehicle to react and stop in the necessary time to avoid an accident; this shows that AVs already have a strong advantage over humans in certain conditions such as nighttime driving. In fact, “at slow speeds, AV performance under degraded conditions may exceed human-driver performance under ideal conditions” [3].

Table 2: Max Speed Allowing For Minimum Stopping Distance [3]

Vehicle type (longest range sensor) [range limit]	Ideal conditions (dry, faster reaction)	Degraded conditions (wet, slower reaction)
Human driver (eyes) [night: 75 m]	85 km/h (53 mph)	60 km/h (37 mph)
AV (radar) [250 m]	210 km/h (130 mph)	145 km/h (90 mph)
CAV (DSRC) [500 m]	305 km/h (190 mph)	215 km/h (134 mph)
Human driver (eyes) [day: 1000 m]	405 km/h (252 mph)	285 km/h (177 mph)

Even factors as simple as music can have an influence on human driving; high volume music, fast tempo music, and the driver listening to non-favorite music all show tendencies of having a bad influence on driving [4]. Just increasing the volume of music from 53 dB to 95 dB prolonged braking response to red signals by 20% [5].

Autonomous systems operate under the methodology of “Sense, Plan, Act” where the system senses its environment, plans out its actions, and acts – or controls – itself to carry out those plans. Therefore, any amount of processing latency can have significant impacts on how the system views the world and reacts to it. For example, take a mid-size SUV with a 60 – 0 mph stopping distance of around 40m on dry roads [6]. This is the physical limitation or the best the system could

achieve if it reacted instantly. However, many operations compound to worsen this number. If a human is driving, it takes a significant amount of time to react to the situation, assuming the driver is even paying attention. Humans take an average of 2.3s to react (with a total range of 0.7s to 3s), including the time it takes from an event occurring to the driver physically lifting their foot off the accelerator and pressing the brakes to max [7]. Assuming the average reaction time of 2.3s, that delay adds 61m to the effective stopping distance, an increase of over 150%. Any decrease in those 2.3s could significantly decrease the stopping distance and, consequently, increase the safety of the vehicle.

To “see” the world around them, AVs use “stacks” of hardware and software to form their perception systems. These systems need to detect objects such as vehicles, pedestrians, and road signs, and track them over time using Multi-Object Tracking (MOT), aka Multi-Target Tracking (MTT), algorithms². The goal of MOT is to perform accurate state estimation of objects over time and provide the most up-to-date estimates of those objects in the systems’ world model. MOTs can also be an integral part of multi-modal multi-sensor fusion; that is, to combine the data from multiple sensors of the same and different modalities (e.g., radar, lidar, etc.). MOTs can also use prior target information to predict the current state of the world, just like humans. Therefore, the implementation of MOTs and their components is very important to decrease the processing time and increase AV safety.

1.1 Motivation

There are two key problems in the research area that this thesis intends to address:

² For the remainder of this thesis, “object” and “target” may be used interchangeably.

1. While MOTs are critical to perception, they are, for a variety of reasons, difficult to understand and learn about. They are complex and require a large breadth of knowledge to understand, create, iterate, tune, and test. They are also a heavily researched area with a wide variety of solutions for different operating environments. MOTs reside at the intersection of various fields, each of which having its operating environments, literature, and expectations; some areas include military use for threat tracking (missiles, drones, etc.), computer vision, robotics, and automated vehicles. Additionally, the literature typically only presents components of tracking algorithms rather than complete ones [8]. Factors such as these make it difficult for new researchers and practitioners to enter the MOT field as well as understand the various MOTs/MOT components and select the best one for their respective situation.
2. The second issue is that academic research tends to focus on individual components, performance, etc., and not implementation. Until recently, source code for tracking pipelines was not readily available, which presented a hinderance to new researchers. Progress has been made with regards to the availability of source code since the contributions from the Naval Research Lab's "Tracker Component Library" in 2017 and MathWorks' sensor fusion toolboxes in 2019. These contributions are further discussed in section 2 Related Work. However, those contributions focus on providing tools to create trackers and not necessarily support for executing them outside of simulation or case studies, details on best practices, etc. Very few have detailed how to implement MOTs on real platforms and discussed the associated issues that the effort presents.

1.2 Contributions

This thesis presents a practical approach by addressing the two problem areas previously discussed:

1. This thesis aims to address the difficulty in understanding and learning MOTs by providing a comprehensive mapping of the field through the lens of AVs. These areas are presented, discussed, and categorized together which, to the best of the authors' knowledge, has not been performed before.
2. An experimental evaluation of a Commercial Off-The-Shelf (COTS) MOT software toolset is performed on real-world data from an Automated Vehicle. Specifically, MathWorks toolsets are applied on the Embry-Riddle Aeronautical University (ERAU) EcoCAR Mobility Challenge (EMC) test vehicle. The results of those MOT systems are evaluated against ground truth data, also collected during those test drives, using MOT evaluation metrics. To the best of the authors' knowledge, this is the first evaluation of MOTs on multiple real-world automated vehicle sensors using COTS software toolsets.

2 Related Work

2.1 Related Review/Surveys

There have been many reviews of MOTs in the past, but all with shortcomings for various reasons, such as discussing only conventional or extended object tracking, lack of implementation or case study discussions, and lack of AV focus. The 1999 Blackman and Popoli book on tracking systems is an extensive textual reference cataloging multiple variations of object trackers, many of which are still commonplace today [9]. However, extended objects and extended object tracking are not covered, which is problematic considering many AV sensors produce extended objects. Another incredible resource is the 2017 review of extended object tracking methods by Granstrom [10]. This paper presents a general overview of EOT methods but not on AV specifics, implementation, or practical ways of using extended object sensors with POTs. Sensors like vision or lidar are technically classified as extended object types but have large research efforts dedicated to the detection and classification of objects, efforts that are unrelated to the tracking field. As such, it is ideal to find ways to integrate new state-of-the-art detection methods in the MOT pipeline. A taxonomy of many different MOTs is presented in [11] but does not have an AV focus or provide implementation details. A 2020 survey of autonomous driving [12] provides great background and overview of AV pipelines, but only briefly covers MOTs. It does benefit from mentioning different operating domains of MOTs, namely 3D space, image space, and point clouds, but fails to discuss POT vs EOT methods and implementation. Extensive research has been performed on visual MOTs, or Visual Object Trackers (VOTs), but that is outside the scope of this thesis; more can be read from [13]–[23].

2.2 Related Implementation Work

Previously in the MOT field, tracker source code was not readily available, which presented an obstacle for new researchers and practitioners. Source code from research papers tended to not be provided at all or provided only for the new component that was presented and not for the whole MOT that was used. This made it difficult for practitioners to implement new methods and for new researchers to enter the field. There were two large-scale attempts to alleviate this. In 2017, the Naval Research Laboratory created an open-source repository called the “Tracker Component Library” to aid researchers in experimenting with tracking algorithms by providing source code for many tracker components [8]. In 2019, MathWorks released its toolsets to support sensor fusion in autonomous systems, including complete pipelines for MOTs [24]. Both helped alleviate the issue of lack of source code presence but did not describe pathways for implementation on real systems like AVs. For instance, while MathWorks has excellent documentation and examples, they focus on the use of simulated trackers with little guidance on real-world usage such as integration, tuning, and analysis.

The literature also does not contain many case studies for MOT implementation on AVs. In [25], EOT methods are used to track vehicles with simulated automotive radar, however, this was not a broader paper such as a survey so it did not have the depth to cover conventional MOTs nor was real-world implementation performed. In [26], the real-time performance of EOTs was discussed and even implemented on real data, however, the method in which that was applied was not discussed. In [27], Mobileye presents their single camera system capable of performing Forward Collision Warning (FCW), but many technical and implementation details are omitted. In [28], a multi-sensor data fusion between a Mobileye camera and an external vehicle is presented but without discussing the implementation details or COTS software.

3 Comprehensive Mapping of MOT Field for Autonomous Vehicles

Multi-Object Tracking is a complex problem with a myriad of different configurations and components. This chapter aims to provide an overview of the field along with different MOT considerations researchers and practitioners may need to consider before implementing MOTs on automated vehicle systems. This chapter is broken into seven sections:

3.1 Comparison and Overview

- a. This section gives a short overview of MOTs, points of confusion, and assumptions made for the rest of this paper.

3.2 Common AV Perception Sensor Types

- b. Exteroceptive sensor types commonly used in AVs are discussed.

3.3 MOT Definitions and Considerations

- c. This section defines the characteristics and considerations of MOTs and how they are modeled.

3.4 Conventional MOT Breakdown and Subcomponent Description

- d. The layout of conventional MOTs is broken down and each subcomponent is discussed. Examples of conventional MOTs are given.

3.5 Extended Object Data and Extended Object Tracking (EOT)

- e. The background is given on Extended Object Tracking and other ways to track extended objects.

3.6 MOTs with Multiple Sensors

- f. The other sections dealt with single sensor MOTs. This section describes methods for using MOTs with multiple sensors.

3.7 Metrics

- g. This section describes metrics to evaluate the performance of MOTs.

3.1 Comparison and Overview

Before explaining the background of each subsection, it is important to note the variety of ways object tracking and sensor fusion are treated, discussed, and referenced in the literature. These methods are often similar and related but evolved separately over time and are referred to differently with their own set of assumptions and connotations, adding to the difficulty in understanding object tracking and fusion. Some terms include:

1. Object Tracking [9]
 - a. The overarching term for tracking objects.
2. Single Target Tracking (STT) [9]
 - a. Object trackers that track a single object.
3. Multi-Object Tracking (MOT) [12], [16], [19]; Multi-Target Tracking (MTT) [9], [10], [16], [29], [30]; Multi-Detection Tracking [10]
 - a. Object trackers that track multiple objects.
4. Multi-Target Filtering [30]
 - a. Filtering for multiple objects, which is separate from tracking them in an MOT.
5. Point Object Tracking (POT); Point Target Tracking (PTT) [31]
 - a. Object trackers assuming a point model (no extent information).
6. Extended Object Tracking (EOT) [10]; Extended Target Tracking (ETT)

- a. Object trackers assuming an extended object model (extent information available).
- 7. Visual Object Tracking (VOT) [32]; Video Tracking [31]
 - a. Object tracking performed on images/video data.
- 8. Detection And Tracking of Multiple Objects (DATMO) [12]; Detector And Tracker of Moving Obstacles (DATMO) [33]; Moving Objects Tracker (MOT) [33]
 - a. Object tracking performed on Lidar data.
- 9. Single Sensor Tracking [34]
 - a. Object trackers that operate using a single sensor.
- 10. Multi-Target Multi-Sensor Tracking (MTMST) [35]; Multi-Sensor Multi-Object Tracking [25]
 - a. Object trackers that track multiple objects with multiple sensors.
- 11. Automatic Target Tracker (ATT) [31]
 - a. Type of tracking used since before WWII.

The different terms alone can make understanding this topic a daunting task, much less when each topic has certain connotations associated with them. For instance, MOT has entirely different contexts and research bases when viewed from an object tracking perspective vs a Visual Object Tracking (VOT) perspective. For clarity, this thesis assumes the following:

1. “Object” and “Target” are used interchangeably. Consistent with [10].
2. “Object” refers to the real-world object while “track” or “object track” refers to the state estimate of those real-world objects.
3. “Association” refers to associating detections with tracks (no-commitment) where as “Assignment” refers to assigning detections to tracks (with commitment).

4. “Observation”, “return”, and “detection” can all refer to feedback from a sensor and can be used interchangeably. However, observations and returns contextually tend to imply generic sensor feedback while detections tend to imply a sensor detecting an object. In EOT, the more generic sensor feedback terms tend to refer to the “raw” data (which could have many observations/returns per object) while a “detection” tends to imply single sensor feedback from one object (often after some form of processing like clustering).

Generically, the purpose of object tracking is to perform object state estimation based on measurement data from sensors.

3.2 Common AV Perception Sensor Types

A variety of sensors and detection methodologies have been utilized within the AV sector. Three of the most common AV sensors are camera, radar, and lidar, but their exact specifications can vary wildly. The most common AV perception setup fuses data from the camera(s) and radar(s) [36]–[38] and represents nearly 70% of the field when compared to camera-lidar, radar-lidar, and radar-camera-lidar combinations [39]. The need for multiple sensor modalities to achieve baseline human performance is shown in Table 1.

3.2.1 Camera

Cameras are ubiquitous sensors and are now practically inextricable from usage in AVs. One reason for this is that they provide features that other sensors have difficulty with, such as traffic sign classification. They have many factors to consider such as cost, resolution, color vs grayscale, frame rate, shutter type, sensor size, and optic parameters [38]. Cameras are excellent in providing more detailed information of the environment such as shape, size/orientation, target recognition, classification, mapping, lane detection, and sign recognition [38], [39]. As opposed

to lidar and radar, single cameras cannot directly measure the distance to targets, but they can estimate it and can directly measure the angle to the target.

An especially important consideration for AVs is the type of camera shutter. Depending on the type of shutter, the entire image can be detected at the same time (global shutter) or rows of pixels can be captured in quick succession (rolling shutter) [40]. Rolling shutter sensors are widespread in consumer devices but global shutters are generally considered the best for properly sensing motion [40], [41]. This is because in high speed, or high accuracy, environments, rolling shutters can cause distortion effects known as wobble or jello effect [40] and can be seen in Figure

1.

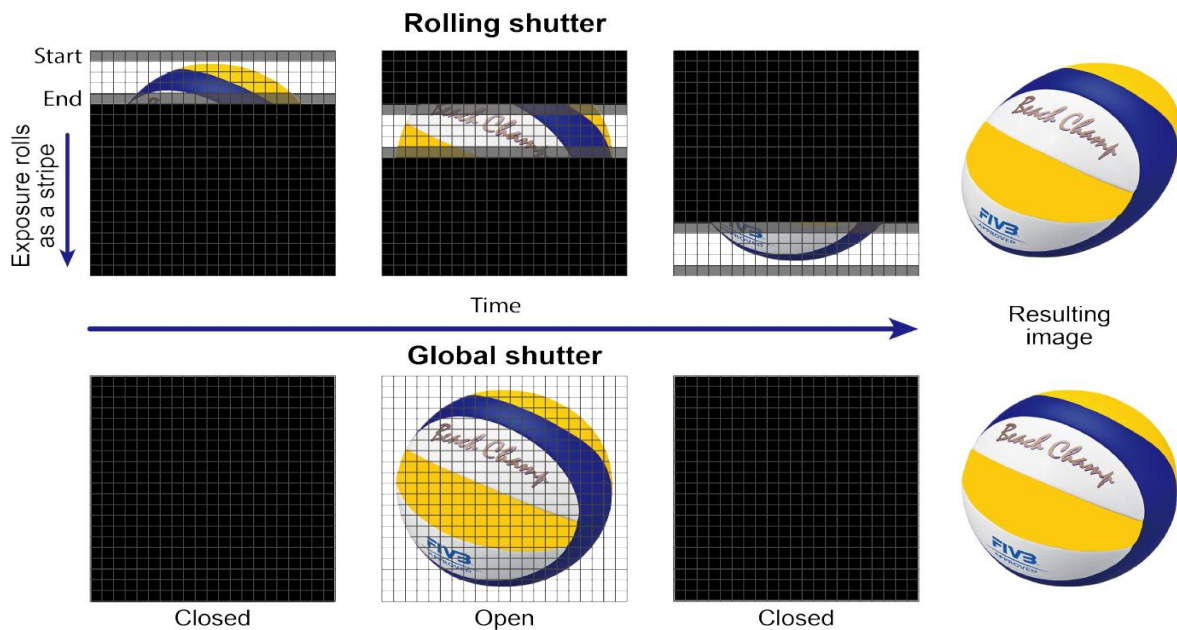


Figure 1: Rolling vs Global Shutter [40]

Other considerations include the type of light the sensor is detecting. Most cameras can detect visible light. Far infrared (FIR) cameras have the benefit of being able to view pedestrians

and animals in the dark and through environmental conditions like dust and smoke [38]. There are examples of these types of cameras already in use such as the General Motors 2016 Cadillac CT6 with an “Enhanced Night Vision” system which detects humans and animals at night and displays the results in the vehicle’s Driver Information Center (DIC), as shown in Figure 2 [42].



Figure 2: General Motors Enhanced Night Vision [42]

3.2.2 Radar

Millimeter Wave (MMW) Radar (RADio Detection And Ranging) is another popular sensor used in many AVs today. Automotive radars typically operate at 77 GHz, are based on Frequency-Modulated Continuous Wave (FMCW) technology, and use digital beamforming to control the direction of the wave [38], [39]. They can accurately measure the distance to objects using round-trip time and frequency shift as well as measuring relative velocity through the Doppler effect [38]. Some of the best arguments for them are their independence to light and weather conditions along with the longest detection range of the three discussed sensors [38]. However, they have downsides such as false positives/false negatives relative to different materials, difficulty detecting stationary objects, and low horizontal resolution/accuracy [38], [39].

3.2.3 Lidar

Lidar (LIght Detection And Ranging) is an active sensor that measures distance by measuring round-trip time from a laser light pulse [38]. They can measure distances up to 200 m, have an accuracy of as good as a few millimeters, often cover 360-degree FOV, and often have rows of lasers to produce several vertical layers and a denser 3D point cloud [38]. They have downsides such as low vertical resolution, sparse measurements (meaning small objects like wire fences may be difficult to see), poor detection of low reflectivity objects, and being impacted by weather more than radar (but less than cameras) [38].

3.3 MOT Definitions and Considerations

MOTs are defined by a variety of factors that must be considered when designing a system. Widely used factors include Tracking Categories and Definitions, Shape Modeling, Dynamics Modelling, Object State, and State Estimation and are discussed in their respective sections of this thesis [10].

3.3.1 Tracking Categories and Definitions

There are a few categories of tracking problems and they are generally differentiated by the number of observations per object per sensor per timestep. Sensors can be thought of as having “resolution cells” which refer to the smallest area the sensor can detect an object within. Objects occupying no more than one resolution cell can have a maximum of one observation while objects spanning over multiple resolution cells could have multiple observations. For instance, Air Traffic Control radar might be able to detect the presence of an object with a certain cross-section and return a single detection per object per timestep. An automotive radar may have a higher resolution and return multiple detections per object per timestep. Lidars have extremely high resolution and may return tens to hundreds of returns per object per timestep. It could also be the case that at large

ranges an object occupies a single resolution cell but as it gets closer it spans multiple cells [10]. Note that this is all relative to the sensors and not the object, even though the object often has domain-specific implications. For instance, an aircraft may return hundreds of Lidar points per timestep, but the object would still be considered a point object if it were being sensed with the fictional ATC radar. This also means new higher resolution sensors could change an object that was conventionally thought of as a point object to an extended object with the sensor upgrade.

The three tracking categories most relevant to Automated Vehicles (AVs) are shown in Figure 3 and are as follows:

1. *Point Objects*

Point Objects refer to objects that only occupy a single sensor resolution cell thus producing a single sensor return per object per timestep.

2. *Extended Objects*

Extended Objects refer to objects that occupy multiple resolution cells thus producing multiple observations per object per timestep.

3. *Group Objects*

Group Objects refer to a grouping of two or more sub objects that share a common motion, such as a group of pedestrians walking in the same direction on a crosswalk. As such, the group object occupies multiple resolution cells per timestep, but the sub objects may occupy single or multiple resolution cells. In this type of tracking, the whole group object is tracked rather than the sub objects being tracked individually [10].

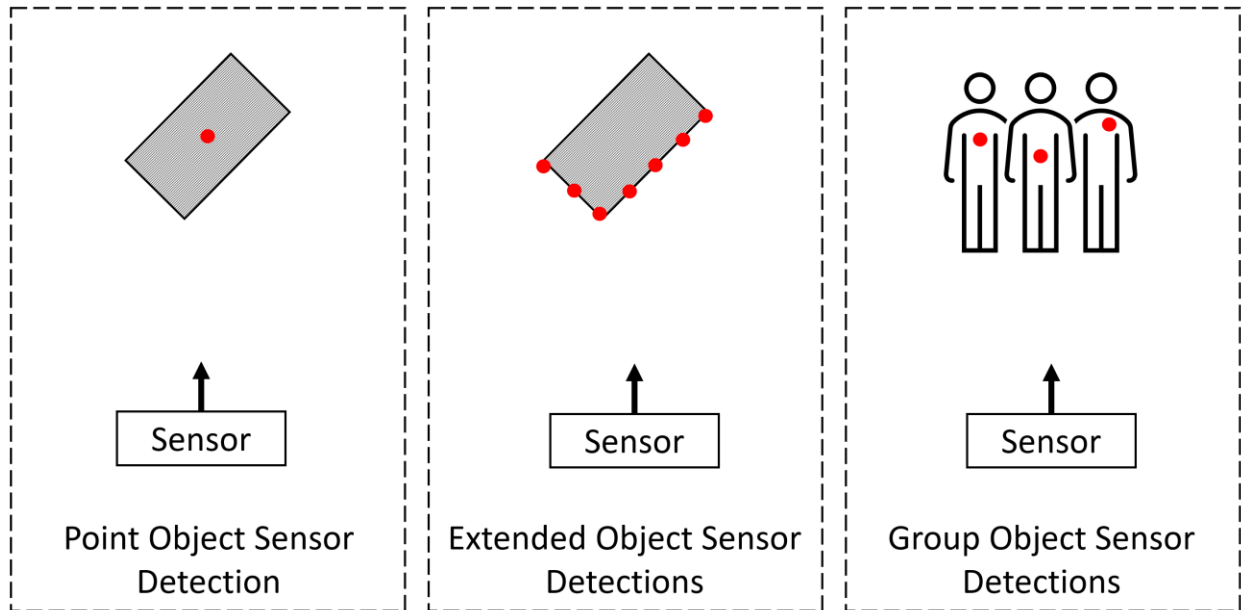


Figure 3: Relevant Tracking Categories

Point objects are the simplest but have the distinct disadvantage of not being able to directly measure the extent of the object, such as width, length, angle, etc. Even though the extent cannot be directly measured with point objects, some extent parameters, such as heading angle, can be estimated over time through motion modeling techniques. For instance, an object may be assumed to travel along its vector from the previous track information which could be used to estimate its heading. The obvious advantage of extended objects is being able to directly measure these parameters. For instance, being able to directly measure the heading angle, or even the steering angle, could be used in the motion model to better perform state estimation. However, the biggest disadvantage of tracking extended objects is the added computational complexity/calculation time.

3.3.2 Shape Modeling

Shape modeling refers to the assumption of shape used within the trackers. While this may seem like the tracking categories, it is different in that rather than being relative to the sensor

resolution cells, shape modeling refers to the shape assumed internal to the trackers. The two shape categories relevant to this paper are shown in Figure 4 and are as follows:

1. *Without Shape (a point assumption)*

Using no shape means a point type is assumed and only the non-extent variables are used. This is often used in conjunction with a point object model where the object only occupies one sensor resolution cell per timestep, so no extent information is present.

2. *With Shape (an Extended Object)*

Using a shape implies there is some form of extended object data present. The model can be broken into two subcategories:

- a. *Basic Geometric Shape*

Basic geometric shapes are used to measure the extent of the objects without needing a high-fidelity model of the exact shape of every individual object being tracked. Two basic shapes are rectangles and ellipses, which are commonly used in automotive Extended Object Tracking (EOT) for tracking vehicles and/or pedestrians [9], [10], [43]. In 2D space, these are represented with three parameters – two-dimensional parameters and one angular/heading parameter. The dimensional parameters for a rectangle would likely be length and width whereas the ellipse would likely be A and B axis length. If 3D estimation is required, the 2D geometric shapes can be extended to 3D shapes such as extending ellipses to ellipsoids.

- b. *Complex Shape*

The complex shape category can handle a variety of shapes and measurement appearances. It can be the most accurate but is also the most difficult to compute. This is also called an arbitrary shape model and can be constructed as a combination of ellipses or a curve with parameterization [10].

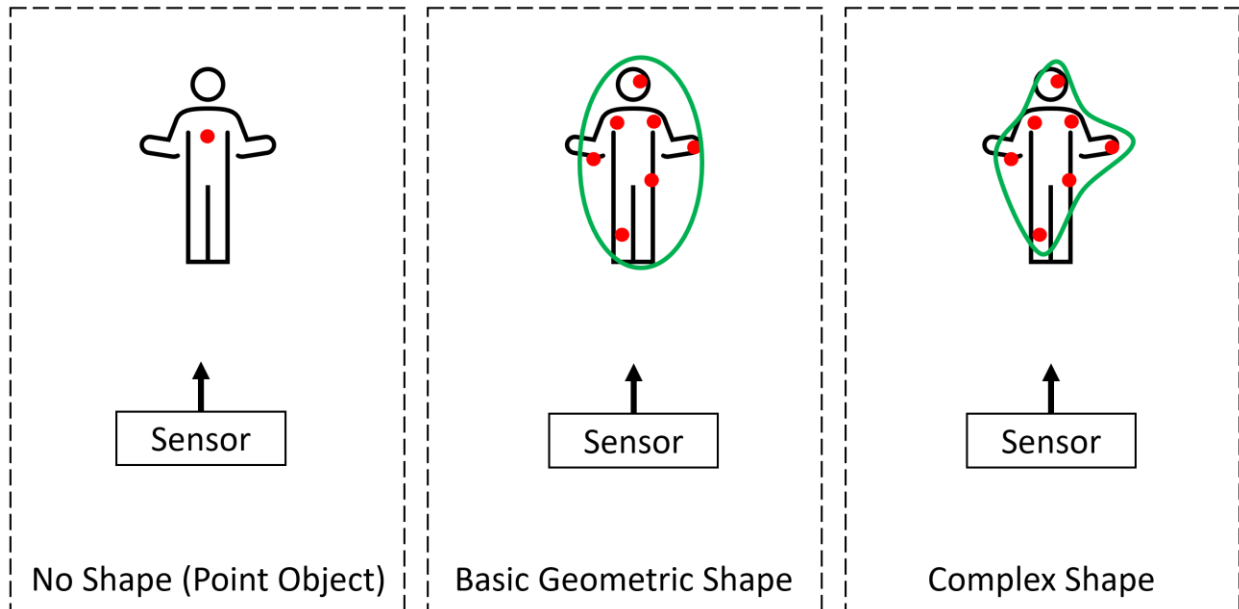


Figure 4: Relevant Shape Models

3.3.3 Dynamics Modelling

Dynamics Modelling refers to the underlying dynamic model of the objects being tracked, which will be used in the state estimation algorithms. This modeling can include all parameters in the object state: position, kinematic, and extent (such as rotation of the shape). Additionally, it can describe how the number of measurements changes such as closer objects occupying more sensor resolution cells thus having more detections [10].

Some common dynamics models include Constant Velocity (CV), Constant Acceleration (CA), and Constant Turn (CT) [44]. These are used in the prediction step of state estimation to determine how to update the appropriate states. Another model that could be used is the bicycle

model, however, it is limited to object like bicycles or vehicles and requires additional knowledge such as the wheel turn angle [10].

3.3.4 Object State

Object State simply refers to the state parameters being tracked for state estimation. They include position, kinematic, and extent measurements. The use of position and kinematic states are common and can be included in any of the tracking categories. The position is the 2D XY location or 3D XYZ position whereas kinematic state can refer to a variety of dynamic motion parameters such as velocity, acceleration, heading, and turn rate [10]. An example of this may look like this:

$$State = [x \ y \ \dot{x} \ \dot{y}]^T$$

Extent, on the other hand, is specific to non-point objects and refers to the shape, size, and orientation of the object. The potential shapes of objects are detailed in Shape Modeling. One example is a rectangular assumption with parameters for length, width, and angle. Sometimes the extent angle will be omitted by assuming it to be equivalent to the kinematic heading angle [10]. An example of this state may look like this:

$$ExtendedState = [x \ y \ \dot{x} \ \dot{y} \ \theta \ \dot{\theta} \ L \ W]^T$$

3.3.5 State Estimation

The last component of an MOT is the state estimation algorithm. State estimators are generally Bayesian estimators, defined as estimators which work to minimize the posterior expected value of a loss function. These estimators are frequently implemented as a type of Kalman filter, which works to minimize the error, or loss, of the estimate over time.

3.3.5.1 Kalman Filter Conceptually

Kalman filters are, in essence, a dynamic weighted average filter between two steps:

1. Prediction Step

The prediction step uses the prior state estimate to estimate the current state based on a process model. The process model is a representation of the dynamics of the system and how the states update over time. In addition to the process model, there is process noise, or how much trust to put into the dynamic model of the system. If there was a perfect model of the system, there would be no noise. However, since this operates in the real world, uncertainty is added to account for unmodelled or extremely complex elements such as wind gusts, speed bumps, different vehicle suspensions, etc.

2. Update Step (aka Correction Step)

The update step uses new sensor measurements to estimate the current state. It also has a model, called the measurement model, but it is simply a way to represent sensor measurements in the format of object state. This also has a noise matrix, measurement noise, which represents the uncertainty of the sensor measurements. Some sensors, such as radar, provide noise estimates that can be fed directly into the measurement noise matrix. Others can be estimated or recorded over time.

The prediction state estimate and measurement state estimate are combined like a dynamic weighted average based on the models' error rates. This is done to form the current state estimate and is done in such a way as to reduce the overall precision error or loss. Since the process and measurement model both have noise, or error, the true state of an object can never be known. However, the combination of these models/estimates results in higher accuracy than either of the models individually; in other words, by incorporating the physics, or dynamics, of the system, higher accuracy can be achieved than raw measurement alone.

This can be more intuitive by overserving Figure 5. In it, assume μ_0 is the mean and σ_0 is the covariance of process model and μ_1 is the mean and σ_1 is the covariance of the measurement model. By multiplying the Gaussian curves together, a new one is formed which includes information from both and at a much higher accuracy, as illustrated by the blue curve in Figure 5.

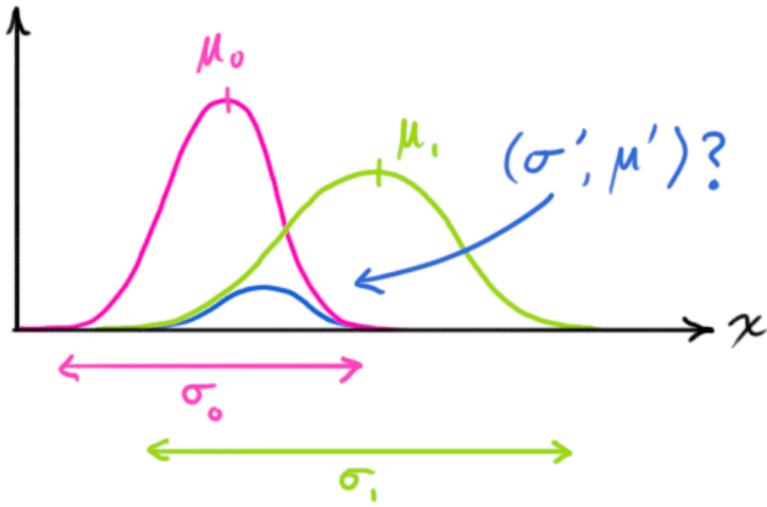


Figure 5: Kalman Filter Covariance Overlap [45]

While this is already impressive, Kalman filters become extraordinarily powerful when they combine estimates from multiple sensor modalities. For instance, common automated vehicle setups use radar and vision systems. Radar is very accurate at measuring the range and range rate to objects, but they tend to be less accurate at estimating the azimuth angle and lateral range rate to a target. Vision systems detect the azimuth angle with ease as the object location in the image has a strong correlation to angle. However, the system struggles with range because it can only be estimated via the number of pixels wide an object is. This effect is compounded at larger distances since as the target vehicle gets further away from the host, the change in range corresponds to a smaller and smaller change in pixels.

A visual example of the covariance for a radar and vision system is shown in Figure 6. The dashed blue ellipse illustrates the covariance of the vision system and shows how there is less certainty in the longitudinal direction but more in the lateral direction. The red ellipse illustrates the radar system and shows a higher confidence in the longitudinal range but less in the lateral.

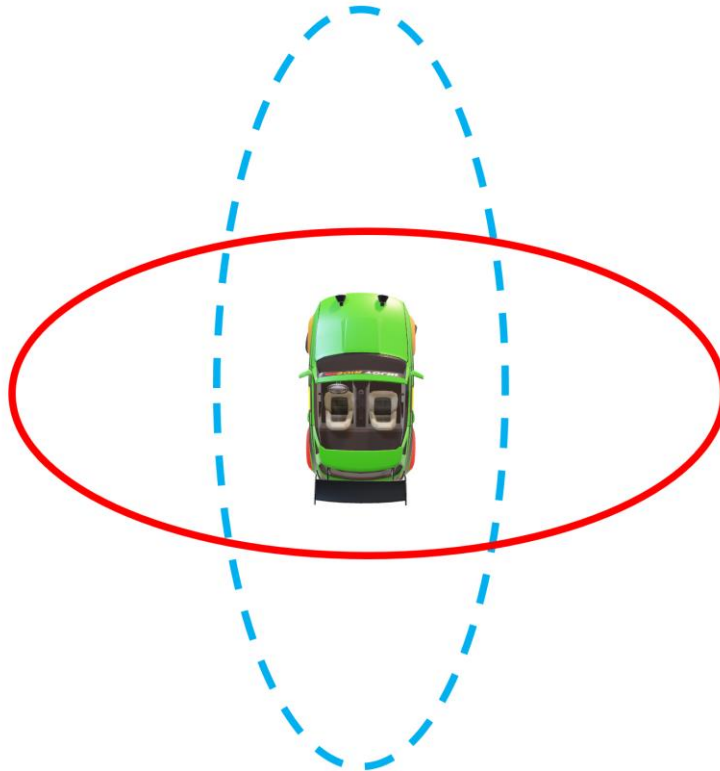


Figure 6: Illustration of Sensor Covariances. Dashed blue ellipse: vision system. Solid red ellipse: radar system.

In Figure 6, there is a smaller, overlapping region of covariances. This region is an area with a higher probability of the true state being in. By combining the two measurements, there is a greater certainty of the true position even before performing any filtering. Combine this with the Kalman filters' ability to incorporate process model predictions and the overall setup becomes significantly more powerful than any individual measurement or sensor could provide.

3.3.5.2 Linear Kalman Filter Mathematically

As mentioned in the previous section, Kalman filters have two main steps – prediction and update. The mathematics behind the steps will be overviewed in the following section, but for a deeper understanding, readers should reference other sources such as [46].

First, the current state at time k is predicted from the previous state at $k-1$ by multiplying it by the state transition matrix, F_k .

$$\hat{x}_k = F_k \hat{x}_{k-1}$$

Similarly, the covariance matrix is updated by multiplying the $k-1$ covariance with the state transition matrix. However, the transition alone does not account for outside disturbances in the update so the process noise, Q_k , is added to the updated covariance.

$$P_k = F_k P_{k-1} F_k^T + Q_k$$

In cooperative systems, some additional information, such as inputs to the system, can be included in state estimation using a control matrix and control vector. However, state estimation for autonomous vehicles generally assumes that vehicles/objects are uncooperative, meaning the host vehicle does not know how the objects are trying to control themselves, so the control matrix and vector are omitted.

The output of the prediction step is the best state prediction and covariance at time k . These are fed into the update (or correction) step to combine with the best state estimate from measurements. The two estimates are combined using a weighted average where the weight of each is determined by K - the Kalman gain.

Like how the process model is updated with the prediction matrix, the measurement model is updated with the sensor matrix, H .

$$K' = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

$$\hat{x}'_k = \hat{x}_k + K' (\bar{z}_k - H_k \hat{x}_k)$$

$$P'_k = P_k - K' H_k P_k$$

3.3.5.3 Other Filters

The filter described in the previous section, Linear Kalman Filter Mathematically, is a linear Kalman filter, meaning only linear process models can be represented. However, real systems are often not linear. The Extended Kalman Filter (EKF) was developed to handle this non-linearity. Details of the EKF and the Unscented Kalman Filter (UKF) are outside the scope of this thesis but are described further in [46]. Another common filter is the particle filter, described further in [47]. Regardless of the implementation, all these filters seek to better estimate the current state of an object using mathematical tools.

3.3.5.4 Improved Modelling with Interacting Multiple Model (IMM)

As discussed, Extended Kalman Filters (EKFs) can handle nonlinear dynamics whereas standard, or linear, Kalman filters only represent linear dynamics. However, both filters still need to make certain modelling assumptions during their respective prediction steps. For instance, a simple process model may have two states, position, and velocity, and uses the previous velocity to calculate the current position. However, it must assume some sort of model to determine the current velocity. As mentioned in Dynamics Modelling, some common assumptions include Constant Velocity (CV), Constant Acceleration (CA), and Constant Turn (CT) [39], [44].

These filters work well when the model is valid (or close to it) but have larger error rates when the model is violated. The Interacting Multiple Model (IMM) is a technique that runs multiple models concurrently and dynamically switches between them based on which has the lowest estimated error [35]. This is more computationally expensive as multiple models need to be run simultaneously, but it can result in higher overall accuracy than any single model.

3.4 Conventional MOT Breakdown and Subcomponent Description

Conventional MOTs are “assignment-based”, referring to setups where sensor observations are first assigned to existing object tracks before state estimation is performed between the existing object track and the new measurement. Note that the assignment process does not guarantee all tracks and detections will be assigned. Dependent on the assignment method, it also may not guarantee one-to-one pairings between tracks and detections. The breakdown of assignment-based MOTs is shown in Figure 7. Note that in practice the steps can be far less discernable; this is just a convenient breakdown for explanation [9]. MOTs are comprised of five main components which are covered in the following sections: 1) Sensor Processing to Detections 2) Track Gating 3) Observation-to-Track Data Association 4) Track Maintenance and 5) State Estimation (discussed in the previous section).

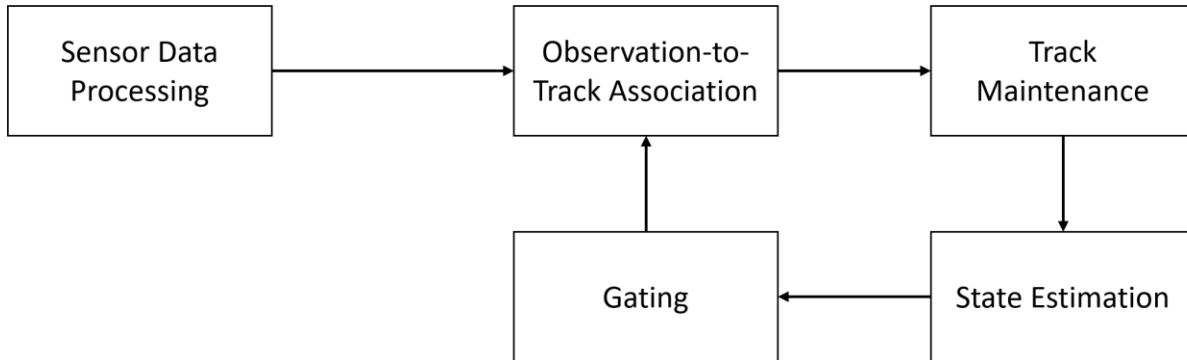


Figure 7: MTT Breakdown (based on [9])

3.4.1 Sensor Processing to Detections

Sensor data processing is needed for converting sensor data into observations usable by MOTs. This is frequently used with extended objects, which are covered later in section 3.5 Extended Object Data and Extended Object Tracking (EOT). However, sub-sections under section 3.4 describe conventional MOTs and assume both a point object category and a point object shape model. This means all observations from sensors represent either an object or noise, so no preprocessing is needed.

3.4.2 Track Gating

Observation-to-track association can be a computationally complex task so, before performing it, track gating is performed to eliminate any unreasonable pairings and reduce the needed computations. This is done by taking previous tracks and forming “validation gates” around the tracks, often in the form of a statistical distance or covariance [10]. For a new observation to even be associated with the track it must fall inside that validation gate.

In a simple example, such as tracking commercial aircraft, the track gates could be so far apart that they never overlap, as shown in Figure 8. This is the simplest case since the not

overlapping implies that all detections are either within the gates of a single known track, meaning the detection could only be assigned to that track, or they are outside all track gates and either represent noise or a new untracked object. That could increase the performance of the system as it eliminates the need to compare all pairings.

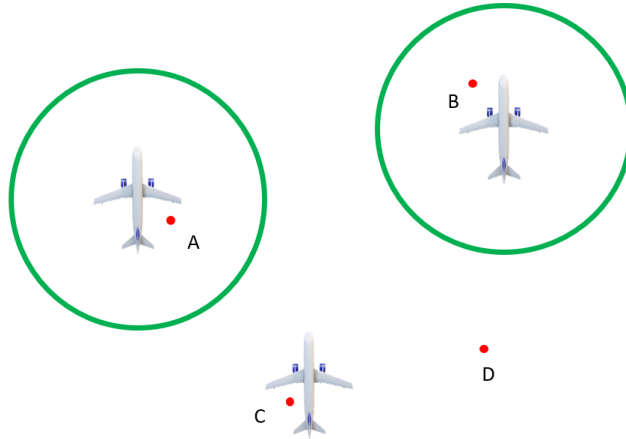


Figure 8: Validation Gates. A and B - detections associated with known tracks. C - unassociated detection representing a real object. D - unassociated detection representing noise.

3.4.3 Observation-to-Track Data Association

After receiving sensor observations, the observations must be associated and then assigned to the objects/tracks they represent before state estimation can be performed. Note that there is a subtle difference between association and assignment; data association refers to determining how well the observations match the tracks whereas the assignment refers to deciding which association(s) to formally select [9]. Once the assignments are made, the observations are passed directly into the state estimator of the respective tracks. Note that this step is called by different names such as track-to-detection [48], detection-to-track [31], [48], observation-to-track [9], measurement-to-track [31], and measurement-to-object [25] association.

Assigning tracks and observations is not always a clear task, as shown in Figure 9. In that example, detection B could reasonably be assigned to either vehicle. In fact, with some sensors such as radar, detection B may even represent both objects.

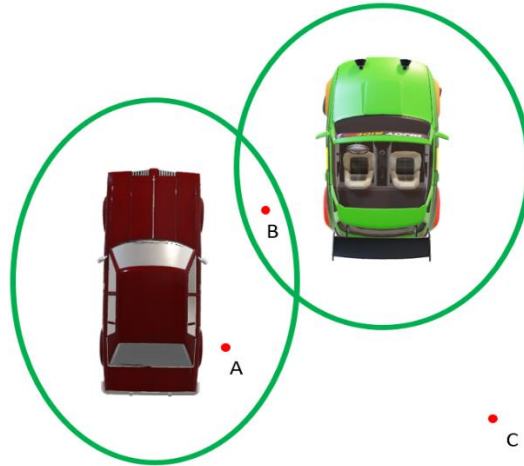


Figure 9: Detection A can only be associated with the left vehicle. Detection B could represent either of the vehicles since it is within both of their gates. Detection C represents unassociated noise.

The first step of determining the observation-to-track associations is by determining the cost of assigning all tracks to all observations, forming an $M \times N$ association, or cost, matrix where M is the number of observations and N is the number of tracks [9]. Specifically, each element in this matrix represents the cost of assigning a specific observation with a specific track. Observations close to tracks will have a low cost, observations far from tracks will have a high cost, and observations between multiple tracks may have very similar costs. Table 3 shows a possible association matrix for the example in Figure 9.

Table 3: Example Association Matrix for Figure 9

	Track 1 (Red Car)	Track 2 (Green Car)
Observation A	2	10
Observation B	3	4
Observation C	10	13

Note that trackers typically limit the association matrix to only consider pairings within the validation gates; this is done by removing the invalid values entirely or by setting them with an unreasonably high cost. The updated example is reflected in Table 4.

Table 4: Example Association Matrix for Figure 9, Including Validation Gates

	Track 1 (Red Car)	Track 2 (Green Car)
Observation A	2	INF
Observation B	3	4
Observation C	INF	INF

Note that these data association methods typically are what differentiate MOTs and, as such, are typically what MOTs are referred to by. For example, a GNN Multi-Object Tracker is an MOT that is using a GNN data association algorithm.

The following sections compare categories of data association techniques, data association examples, and assignment algorithms.

3.4.3.1 Unique-Neighbors vs All-Neighbors Data Association

Unique-Neighbor’s data association is an approach that determines a one-to-one observation-to-track pairing, meaning that a track may only be updated by a single observation and an observation can only be used to update a single track. All-Neighbors data association, on

the other hand, allows for all neighbors within the gated region of a track to be used together to update the state [9]. In the Figure 9 example, detection B could only be assigned to one of the tracks if a unique neighbor's approach were used but would partially represent both tracks if an all-neighbors approach were used. Note that this is different from EOT techniques since EOT assumes multiple data points represent a single object.

3.4.3.2 Sequential vs Deferred Decision Data Association

Sequential decision data association, also known as single-hypothesis association, means that at every timestep a single decision, or hypothesis, must be made for each observation-to-track assignment and the assignment is not reversible. Deferred decision data association, also known as multi-hypothesis association, allows for multiple predictions of observation-to-track assignments existing at each timestep and allows for the final decision to be made at a later time [9]. In the Figure 9 example, the proper assignment of detection B is ambiguous; with deferred decision data association, the tracker could keep two possible hypotheses of its assignment and defer the official assignment until more data is available.

3.4.3.3 Data Association and Assignment Examples

This section presents three common data association techniques [9].

3.4.3.3.1 Global Nearest Neighbor (GNN)

One of the simplest methods of data association is Global Nearest Neighbor (GNN). It is a type of unique-neighbor and sequential data association approaches. It is sometimes referred to as sequential most probable hypothesis tracking [9]. Some examples of algorithms used for GNN are listed in 3.4.3.3.4 Unique-Neighbors Assignment Algorithms. GNN uses the association, or cost, matrix to determine which pairings of tracks and detections result in the best overall score. Since it is a unique-neighbor approach, there can only be 1-to-1 pairings. Since it is a sequential decision

approach, the assignment it decides is final and cannot be changed in future iterations. GNN is a very simplistic algorithm so while it is the most computationally efficient of the three in this section, it does not perform as well in cluttered environments. The following methods can have much better performance in those environments at the cost of computational efficiency.

3.4.3.3.2 Multi Hypothesis Tracker (MHT)

The Multi Hypothesis, or Multi Hypothesis Tracker (MHT), is a data association method that is a type of unique-neighbor and deferred decision approach that works well in cluttered environments [9]. Like GNN, it is unique-neighbor approach meaning that it can only perform 1-to-1 pairings between single tracks and detections. Unlike GNN, it is deferred decision meaning it can maintain multiple assignment hypotheses and later decide on one when more data becomes available. Each hypothesis is still only 1-to-1 pairings, but multiple hypotheses allow for multiple “guesses” on which pairing is most correct; this is what allows it to perform better than GNN in cluttered or noisy environments. For example, in one iteration an incorrect detection may appear like a better pairing than the true detection. In GNN the incorrect detection would be paired for that iteration. In MHT, two hypotheses could be made: one where the incorrect detection is paired and one where the true detection is paired. In future iterations, the MHT could gain enough evidence to remove the incorrect detection hypothesis.

3.4.3.3.3 Joint Probabilistic Data Association (JPDA)

JPDA is an all-neighbors data association approach that can use multiple observations to update a single track. It can be implemented as a sequential decision or a deferred decision approach [9]. It performs the best of the three in cluttered environments but is also more complex. Since it uses an all-neighbors approach, it does not need to form 1-to-1 pairings. It can instead take

information from all detections within its validation gate to update the track. The information from the different detections is combined by using a soft assignment.

3.4.3.3.4 Unique-Neighbors Assignment Algorithms

After obtaining the association matrix, an assignment algorithm determines which combination of assignments results in the lowest overall cost to the system. Many assignment algorithms were initially developed for applications other than MOTs, such as finance and labor distribution, but have since been used in MOTs [9]. These algorithms are unique-neighbors assignment algorithms. Some common algorithms include ‘MatchPairs’, ‘Munkres’, ‘Jonker-Volgenant’, and ‘Auction’ and are all options in the MathWorks GNN tracker [49]. Another option common in VOT is the Hungarian Method [50]. The goal of these algorithms is to use the association, or cost, matrix to determine which combination of 1-to-1 pairings result in the best overall cost to the system. The best combination of pairings are used as the final assignments.

3.4.4 Track Maintenance

Track maintenance is an overarching term for track creation, deletion, and confirmation. When a new object comes into the sensing FOV, the object is represented as detection(s), but the MOT does not know if the detection represents a real object or noise. During this time, it creates a tentative track and waits for more updates which would provide enough evidence to confirm it as a real track. Once the object has left the FOV, the MOT processes it as missed detections but does not know if that means there is a temporary occlusion, or the object is leaving the FOV. Once enough evidence is present that the object is gone, the MOT deletes the track. The track maintenance step makes MOT robust to things such as noisy sensors or temporary occlusions.

There are two main ways of confirming/deleting tracks which are based on track history and track score [9]. The track history method evaluates the recent history of sensor updates to see

how many times detections have been matched to the track in question. This is also referred to as the *M-out-of-N* method, referring to the M number of matched detections required to be in the last N updates for the track to be confirmed. Alternatively, score-based methods calculate the likelihood ratio and, for convenience, the log-likelihood ratio that the track is valid before confirming it [9]. Once the track is confirmed, it is held as confirmed until it matches enough criteria to be deleted. In the case of *M-out-of-N*, this could be dropping below the M needed detections. For score-based methods, this is dropping below a specified deletion score.

3.5 Extended Object Data and Extended Object Tracking (EOT)

The following section describes using data from extended object sensors and Extended Object Trackers (EOT).

3.5.1 Extended Object Tracking Overview

An effective approach for utilizing high-resolution sensors is through Extended Object Tracking (EOT), a general term for techniques that directly account for extended objects (multiple detections per object per scan) and the shape, or extent, of those objects rather than a point assumption [10]. Note that the difficulty here is that there is an unknown number of objects represented by an unknown number of detections per object plus noise. This means there is no direct correlation between the number of detections and the number of objects as there is with point object tracking. This means that in addition to properly associating multiple observations with their respective objects/tracks, the algorithm also must determine how many objects are present to try and associate the observations.

Rather than assuming a point object, EOT's make use of the discrete returns to provide increased accuracy, decreased misclassification, and additional object state data, such as target dimension and orientation [10]. EOTs use predefined bounds and kinematic models to associate

groups of detections with shapes, typically simple geometric shapes like rectangles or ellipses [9], [10], [43]. Because EOTs can measure the shape and heading directly, they use that to improve their measurements by fitting data to those shapes. In the case of a vehicle, the EOT uses the shape from the previous estimate in the current one so a better estimate can be made even with fewer data points. For example, if a lidar goes from having data on both the back and the side of the vehicle to only one side, the rectangular outline is propagated from the previous timestep, so the size does not dramatically change. This shape awareness is a key EOT advantage for situations where objects are close to each other since it is better equipped to determine the proper associations.

3.5.2 EOT Data Association

In EOT data association, measurements are modeled as either being an object measurement or noise measurement. This means the data association can be split into two parts, partitioning and cell association. Partitioning is done such that each group of detections or “cell” that is identified contains detections that all represent either the same object or no object at all (noise) [10]. After partitioning into cells, the cells are associated with existing tracks and the detections are processed to determine the measurement for the object.

3.5.3 Extended Object Tracker Examples

The exact details of these EOTs are outside the scope of this thesis but are commonly based on Random Finite Sets (RFS) and Finite Set Statistics (FISST). Some examples are Probability Hypothesis Density (PHD), CPHD, CB-MeMber, Generalized Labeled Multi-Bernoulli (GLMB), Labeled Multi-Bernoulli (LMB), Poisson Multi-Bernoulli Mixture (PMBM), and Variational Multi-Bernoulli (VMB) [51].

3.5.4 POT to EOT

The previously discussed, EOTs take all the sensor data into account then determine the number of objects and how the observations are associated with those objects before moving onto state estimation. Arguably, these EOTs are acting as a form of object detector by converting the extended data into a format that represents individual objects that can be tracked. However, there are other methods of converting extended object data into a format compatible with MOTs, specifically with conventional MOTs. Some of these methods are discussed in the following sections.

Even though the extended object data cannot be used directly, it may still be desired to use a Point Object Tracking (POT) method. Conventional MOTs, such as POT, are heavily researched and are simpler, more efficient, and more flexible when compared to EOTs. This makes POTs a very attractive option, especially when the desired use case is a real-time environment with processing and time constraints. Some methods of using conventional MOTs (or POTs) with extended object data are discussed in the following sections.

3.5.4.1 Clustering

One method of using POTs with extended object sensor data is to cluster the returns into a single point then treat that point as a single detection that can be put through standard MOTs. One example of clustering is Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which clusters data based on its proximity to other data points (aka the density of the data in that region) [52]. The downside of any clustering is that information is lost when the data is reduced to a point object since the clustering algorithms do not account for the geometry of the object they are detecting. This results in the center point of the cluster being located at the centroid of the detections, which is not necessarily the center of the object [53]. In addition to the resulting

decrease in positional accuracy, this method is also susceptible to returns being incorrectly clustered into nearby detections, as illustrated in Figure 10. While it has its downsides, clustering data into a point object is still a simple and effective way of using extended object data in conventional MOT, provided the desired operating environment does not interfere with clustering such as a cluttered or closely spaced environment.

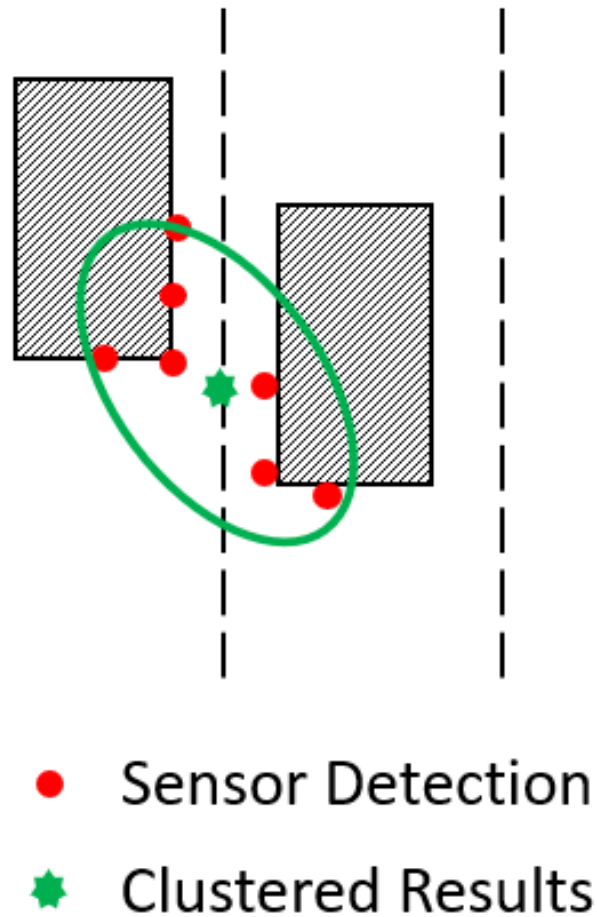


Figure 10. Incorrect Clustering of Sensor Data [54]

3.5.4.2 Other Detections/Processing Techniques

Recently there have been significant performance advancements of sensor detectors using methods such as deep learning for camera data [39]. These detectors are specialized in their data types and provide impressive accuracies, making them more conducive to using over raw EOT

methods. Additionally, sometimes these detectors need to be running anyways for reasons other than detection, such as the classification of traffic signs. Some sensors, such as radar, are often sold as integrated systems that output processed detections rather than raw observations, also reducing the AV integrator's need to use EOT methods.

There are many types of detectors and they are often entirely unrelated areas of research. Deep learning vision processing techniques have witnessed an explosion in research since AlexNet was presented in 2012 and “shattered the ImageNet image recognition challenge” [12], [55]. Lidar processing techniques are also being heavily researched and have resulted in many effective algorithms such as VoxelNet, SECOND, PointRCNN, and PointPillars [12].

No matter the exact detector being used, once it provides some state measurement the observations can be fed directly into conventional MOTs. The conventional MOTs can even be used as a pseudo-EOT method if the object extent is added to the object state.

3.5.4.3 Summary

In summary, there are eight overall types of tracking when it comes to these distinctions. These eight types are from the permutations between:

1. The tracker type – whether the tracker is conventionally POT or conventionally EOT.
2. The shape model – whether the tracker uses point object or extended object representation.
3. The tracking category – whether the sensor observes point or extended objects.

This categorization is summarized in

Table 5 and, to the best of the authors' knowledge, is a novel contribution.

Table 5: Tracker Categorization

Tracker Type	Shape Model	Tracking Category	Description
POT	PO	PO	This category represents conventional MOTs with PO sensors.
POT	PO	EO	This category reduces an extended object to a point object, either through clustering or a detector without extent info. It can be efficient but loses information and possibly accuracy.
POT	EO	PO	N/A. This would suggest a point object can gain extent information, which it cannot. Entry kept for completeness.
POT	EO	EO	In this category, a detector is used to process sensor data into a point object with extent estimates to mimic extended objects. The tracked object state includes extent data, but it is not used to aid accuracy in future iterations like with EOTs.
EOT	PO	PO	In this category, point objects are tracked with trackers traditionally categorized as EOT. One instance of this is using GM PHD to track point targets in dense clutter as it “provides better estimation of objects as it handles clustering and data association simultaneously” [56].
EOT	PO	EO	To the authors’ knowledge, this category also has no example or use case. It implies that extended object data is reduced to point object form before being run through a complex EOT that does not know its extent.
EOT	EO	PO	N/A. This would suggest a point object can gain extent information, which it cannot. Entry kept for completeness.
EOT	EO	EO	This category represents conventional EOTs with EO sensors.

3.6 MOTs with Multiple Sensors

Up until this point, MOTs have been discussed with the assumption of using a single sensor. However, AVs are almost certain to contain more than one sensor and more than one sensor modality. This necessitates the discussion of two types of MOT fusion, central-level track processing and sensor-level track processing [9]. Central-level track processing refers to

information fusion before tracking, such as concatenating and clustering radar data from different sensors before tracking. Sensor-level track processing refers to tracking before fusion. Sensor-level track processing performs object tracking at the sensor level (e.g., each sensor has an output of object tracks) then fuses the tracks. This is also referred to as track-to-track fusion.

Central level tracking has the potential to have better performance as all the raw sensor measurements are available at the time of fusion and tracking. When performing sensor level tracking, the information is condensed into tracks which can result in lower performance [9].

Sensor level tracking does condense information into tracks first, but that does provide some benefits, such as modularity, complexity reduction, and reusability of the algorithms [34]. However, this comes at the expense of running multiple trackers, which could be computationally expensive with many separate sensors being processed on a single computer. SLT allows the individual trackers to be tuned independently and relative to each sensor before any fusion takes place. It also hides implantation details of the sensor level trackers, which might be useful in production environments such as sensor manufacturers that output tracks rather than raw detection information. In the case of some automotive sensors using the limited data rates of CAN buses, it may also be impractical to send raw sensor information. Having the individual sensors run the trackers also alleviates the increased computing needs if all tracking was running on one machine. Even though the discussion has been focused on an individual vehicle running SLT, the same concepts can be applied to multiple vehicles. In this scenario, each vehicle would have its tracking system that condenses all sensor information into tracks that can be fused with tracks from other vehicle tracks.

3.6.1 Track Fusion Methods

Sensor-level track processing relies on fusing tracks from multiple sources. To do so, it needs a strategy to fuse the covariance of the tracks. However, this often requires either independence or prior knowledge of the cross-covariance, which is often unknown due to fusion data being correlated [57].

There are two methods of interest for covariance comparison, the cross-covariance and covariance intersection. Both are covered in [58] and covariance intersection is shown in Figure 11.

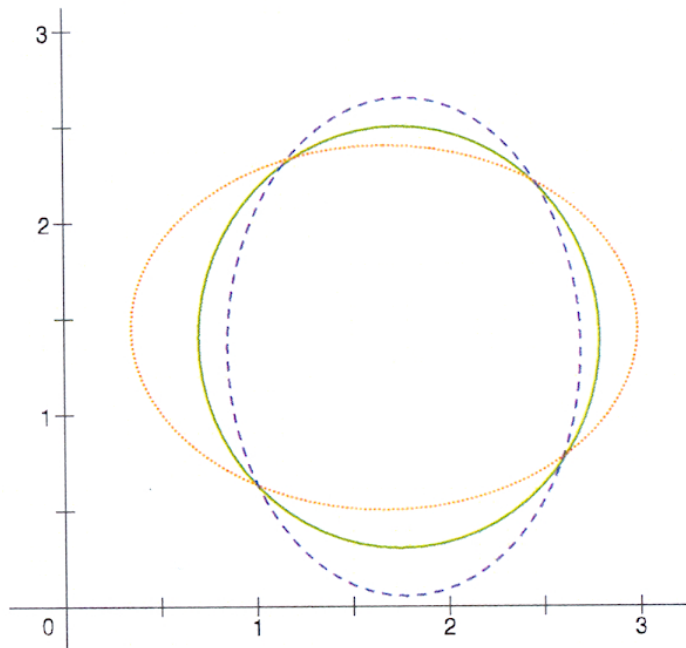


Figure 11: Covariance ellipses of A (dotted red ellipse), B (dotted blue ellipse), and Covariance Intersection of both (green ellipse) [58]

3.7 Metrics

Standard metrics are needed to compare performance across various tracker configurations. Conventional error metrics calculate the error of a single element such as the position error of one

track over time. Since MOTs track multiple objects, there are other metrics to evaluate how well the tracks were assigned, created, etc. OSPA is a metric that combines multiple sub-metrics into one value to represent the whole tracker. These metrics are discussed in their subsections below.

3.7.1 Error Metrics

One popular error metric is the Root Mean Squared Error (RMSE) [10]. This calculates the RMS over a list of errors so it is very flexible on how it can be applied. Normalized Estimation Error Squared (NEES) is another standard performance metric and is used for Gaussian assumed state estimates that also incorporates the covariance matrix [10].

A widely used computer vision metric is Intersection-over-Union (IoU) “which is defined as the area of the intersection between the estimated shape and the ground truth shape, divided by the area of the union of the two shapes” [10]. While traditionally used in computer vision, it is also used for comparing the IoU of extended objects and their shapes [43].

3.7.2 Assignment Metrics

Standard error metrics like RMSE account for the accuracy of a single track but are not sufficient to describe the overall performance of a complete MOT. This is because MOTs also need metrics for categorizing a variety of track factors related to the observation-to-track assignment. Some examples of these metrics include the number of true and false tracks, number of track ID swaps, and number of redundant tracks [59].

3.7.3 OSPA

The Optimal Sub-Pattern Assignment (OSPA) metric is a single value that illustrates the overall error of each tracker by incorporating multiple types of Measures of Effectiveness (MoE) such as position errors, unassigned tracks, and incorrect assignments [19], [30], [60], [61].

3.8 Chapter 3 Summary

This chapter has provided a comprehensive mapping of MOTs through the lens of Automated Vehicles through six sections. Section 3.1 gave a short overview of the sections along with some common MOT names and assumptions used in this thesis. Section 3.2 discussed common AV sensor types and the need for AVs to include multiple sensor modalities. Section 3.3 defined and discussed different considerations researchers and practitioners must think about when designing or selecting a Multi-Object Tracker. Section 3.4 presented a breakdown of conventional MOTs and examples of common MOTs, such as GNN, JPDA, and TOMHT. Section 3.5 dove into Extended Object Tracking and methods used for processing extended object data with conventional MOTs. Section 3.6 discussed central-level and sensor-level track processing for creating tracking systems with multiple sensors and potentially multiple sensor modalities. The final section, section 3.7, presented some metrics used for evaluating and comparing MOT configurations against one another. The following chapter will build on this by implementing an MOT on an Automated Vehicle using software toolsets.

4 Real-World Evaluation of an MOT Software Toolset on an AV

This chapter presents a case study for the implementation of Multi-Object Trackers on an Automated Vehicle (AV) using Commercial Off-The-Shelf (COTS) software toolsets. Specifically, MathWorks toolsets were used to analyze data and deploy MOT models to a “mule” vehicle in conjunction with the EcoCAR Mobility Challenge (EMC). Sensor data and ground truth data were first calibrated, recorded, and analyzed. Later, MOT configurations were tuned and evaluated offline using ground truth comparison metrics from the Metrics section. While the tracker performance is evaluated, it is just one example of an MOT system and its performance is not the sole focus of this evaluation. The primary purpose of this evaluation is to evaluate the toolset and the process of integrating the MOT with real-world AV sensor data. This chapter is broken into five sections:

4.1 Vehicle Configuration – discusses the physical setup of the vehicle and sensors.

4.2 Software Configuration – discusses the software tools used in this evaluation.

4.3 Issues and Considerations – discusses known issues and considerations that needed to be accounted for.

4.4 Experiment – discusses the setup, operation, and evaluation of the case study.

4.5 Tracking Toolset Review – discusses the overall review of the MathWorks tracking toolset.

4.1 Vehicle Configuration

The AV from this case study is used in conjunction with Embry-Riddle Aeronautical University’s (ERAU) EcoCAR Mobility Challenge (EMC) team. The EcoCAR Mobility Challenge “will challenge 11 university teams to apply advanced propulsion systems, as well as connected and automated vehicle technology to improve the energy efficiency, safety and

consumer appeal of the 2019 Chevrolet Blazer” [62]. The specific vehicle used in this case study is the teams’ “mule” vehicle used to test sensors, perception algorithms, and perception system integration without the need of the final vehicle. This vehicle is equipped with a Bosch radar, Intel Mobileye 6 vision system, a webcam (for diagnostic purposes), a Linux computer (the Intel AIOT TANK) equipped with a Kvaser PCI-E CAN (Controller Area Network) card to communicate with the devices, and a ground truth system from Oxford Technical Solutions (OxTS). The radar, Mobileye, and TANK were all sponsored by their companies as a part of the EMC competition.



Figure 12: ERAU Test Vehicle

Figure 13 illustrates the component layout, coordinate frames, and vehicle coordinate frame of the system.

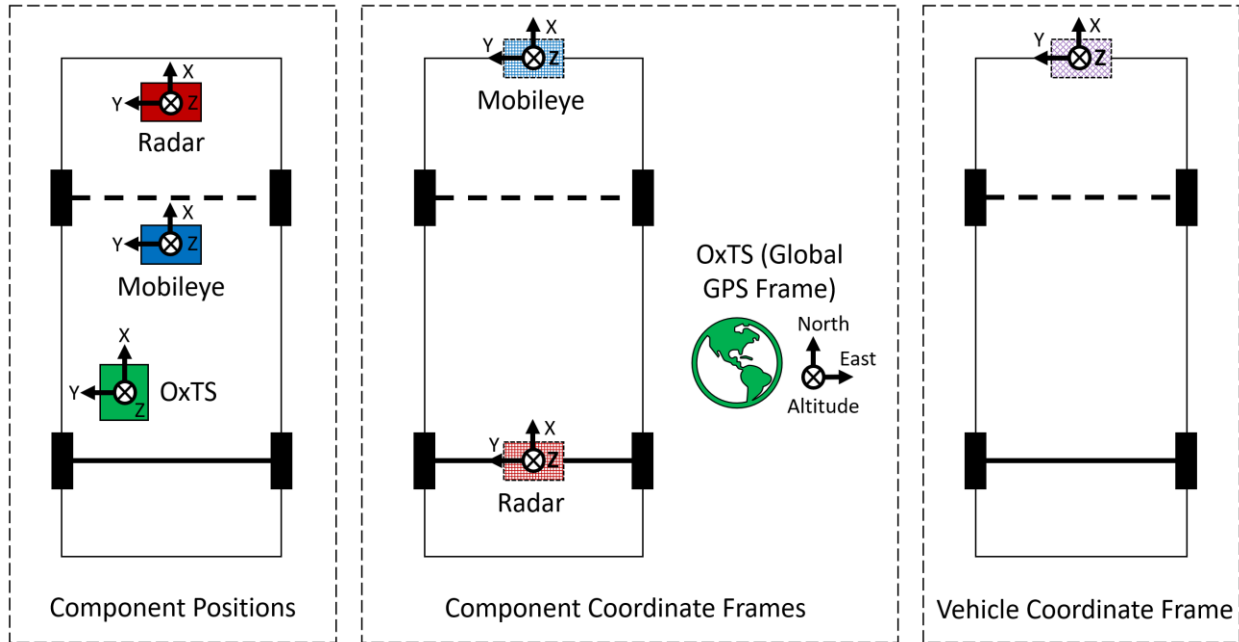


Figure 13: Component Layout

4.1.1 Bosch Radar

The Bosch radar is an automotive radar designed to enable AV features such as Adaptive Cruise Control (ACC) and Automatic Emergency Braking (AEB). It is a Frequency-Modulated Continuous Wave (FMCW) radar operating at 77 GHz and uses digital beam forming to allow for flexible antennae use and high accuracy results [63]. It has a max range of 160 m and a max FOV of 84 degrees. It can detect up to 32 objects and returns their measurements over CAN bus. For this case study, one forward-facing radar was used, and its mounting is shown in Figure 14. Bosch does not have a ROS driver available so one was developed in C++ including custom ROS message definitions.

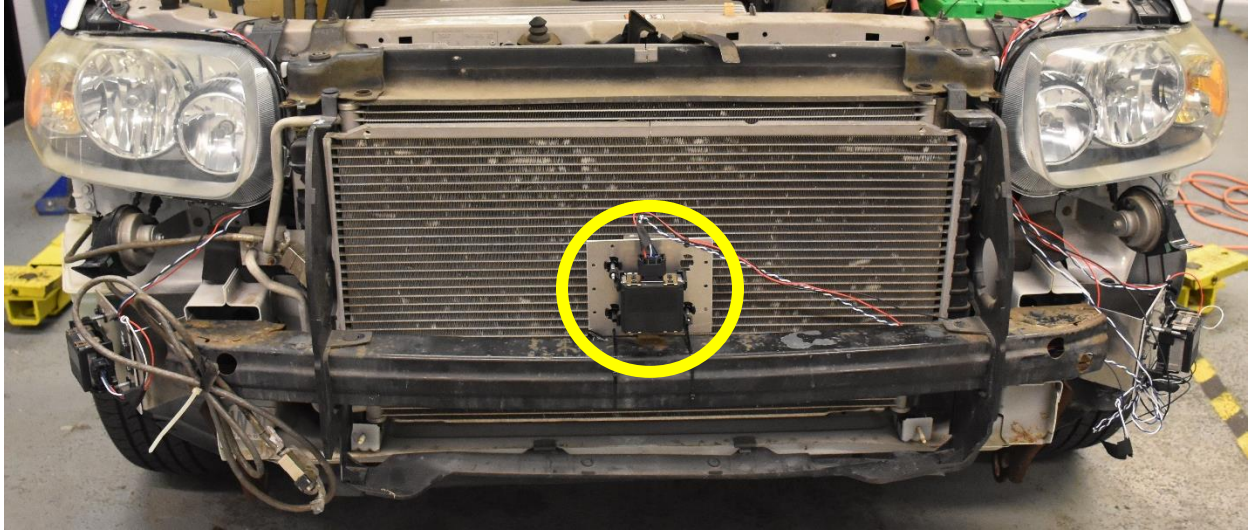


Figure 14. Bosch Radar [54]; Only The Circled Radar Is Used In This Thesis

The radar requires status messages at certain intervals which contain information such as vehicle speed, turn rate, and the position of the sensor on the vehicle. In part, it uses this information to determine the conversion between the sensor frame and the rear axle frame, the frame in which it reports all detections in. The detections are later converted to be in the front bumper frame both to match the Mobileye and to be more intuitive with zero distance between vehicles being shown as zero distance relative to the front bumper.

4.1.2 Intel Mobileye 6 Vision System

The Mobileye 6 Vision system is an automotive collision avoidance system designed to provide drivers with auditory notifications for events such as Forward Collision Warning (FCW) and Lane Departure Warning (LDW) [64]. More on the original Mobileye design can be read in [27]. While the system is designed to be installed directly in vehicles for collision avoidance, the Mobileye Development Kit provides access to the underlying detections for vehicles, lanes, pedestrians, and traffic signs for developers to use [65]. The max vehicle detection range is 150m, and the max pedestrian detection range is 40m. The Mobileye is mounted alongside the webcam

and is shown in Figure 15. The Mobileye system had an existing ROS driver and ROS messages available through the distributor AutonomouStuff.



Figure 15: Vision Setup; Mobileye 6 (Center) and Diagnostic Camera (Left)

The Mobileye is calibrated either by using a calibration board or using self-calibration by driving around street environments for a certain amount of time. After the calibration is complete, detections are reported on the CAN line and are reported in the front bumper coordinate frame.

4.1.3 Ground Truth

Evaluating the efficacy of the tracking solution was done by comparing the fused results to ground truth data. This ground truth data was collected at the same time as sensor data through an RT3003G unit from Oxford Technical Solutions (OxTS), pictured in Figure 16. This device uses high accuracy dual-antenna GPS, Accelerometers, and Gyroscopes to achieve up to 0.01m Circular Error Probable (CEP) position accuracy, 0.05 km/h velocity accuracy, and 0.1-degree

heading accuracy [66]. Before recording data, this unit goes through its warm-up and calibration phase while being driven around.



Figure 16: OxTS Ground Truth Unit

After recording ground truth data, the matching GPS base station data is downloaded from the Florida Permanent Reference Network (FPRN) [67]. The OxTS vehicle data along with the closest FPRN station data is fed into the OxTS software for post-processing, GPS corrections, then exported to an accessible format such as a CSV file. Post-processing improves the final location data by correcting the GPS for local errors such as weather and fusing the various sensors. An example of location data before and after post-processing is shown in Figure 17.

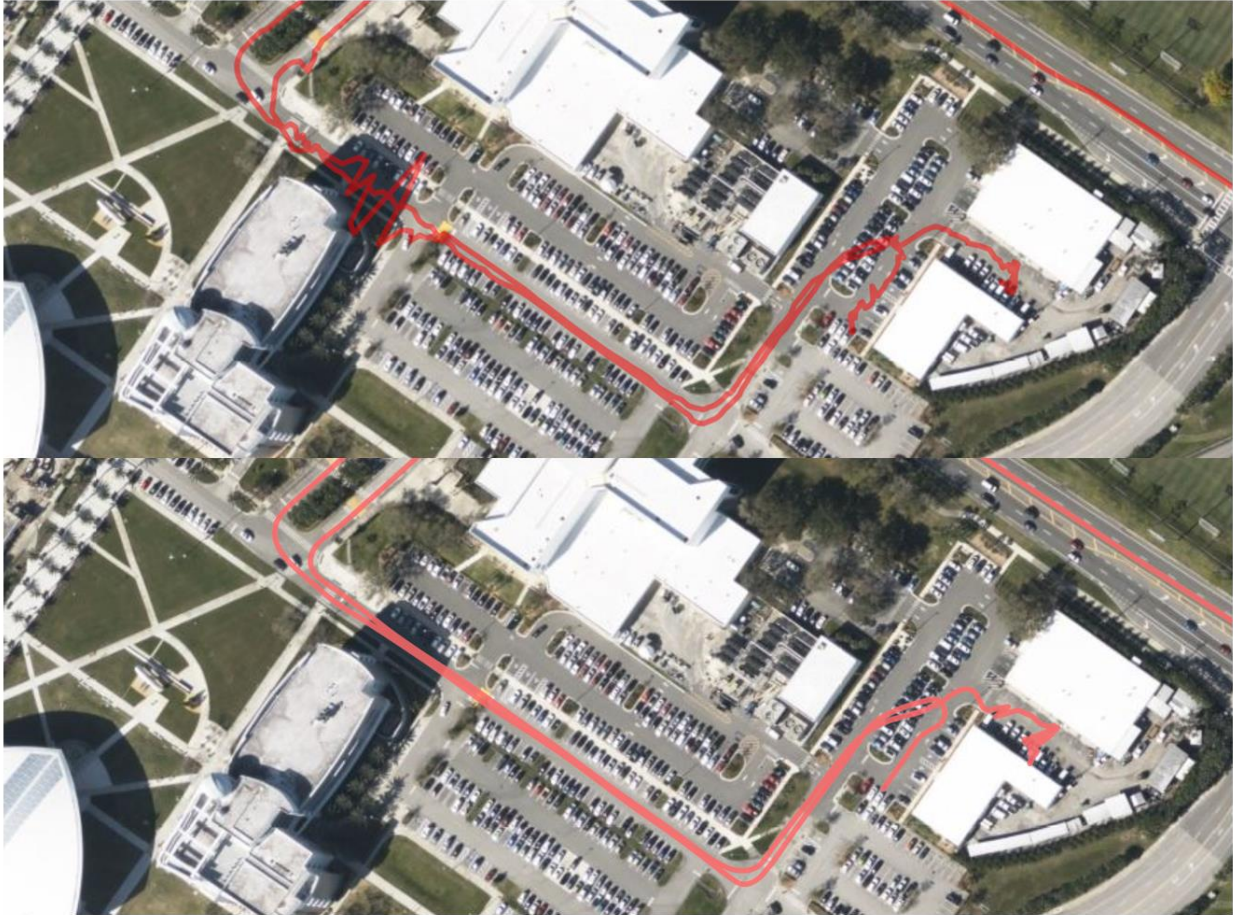


Figure 17: OxTS Ground Truth Data Before Corrections (above) and after (below)

The high-accuracy ground truth data is provided in an absolute GPS reference frame, which cannot be used directly. Instead, it is first converted to a local vehicle coordinate frame for direct comparison with sensor fusion. This process consists of two transforms:

1. Transform the GPS locations of the host vehicle into a relative ENU (Easting-Northing-Up) frame using the target vehicle location as the origin. This is done with MATLABs “latlon2local” function, which converts a GPS location into a relative ENU frame in meters based on an GPS origin position.

2. Transform that ENU frame to a local vehicle FLU (Front-Left-Up) frame relative to the front bumper of the host vehicle by applying the following transform:

$$FLU = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ -\cos\theta & -\sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Easting \\ Northing \\ Up \end{bmatrix}$$

These steps transform the absolute GPS ground truth data into a relative frame suitable for direct use with comparison of MOT/fusion results.

During initial testing, only a single OxTS unit was available so no direct comparison between the two vehicles was possible. To alleviate this, a scenario was set up with the target vehicle in a static location (shown later in Figure 19). The host vehicle is placed as close as possible to the target vehicle and its position is recorded in place of the target vehicle. That location is later used as the target location while the host vehicle location is polled from OxTS data. These locations can then be used alongside one another as if they were two separate units.

4.2 Software Configuration

4.2.1 Software Toolsets

This case study relied on two main software toolsets: Robot Operating System (ROS) for drivers, communication, and data recording, and MathWorks to analyze data and create, tune, and deploy Multi-Object Trackers (MOTs).

Robot Operating System (ROS) [68] organizes algorithms and packages into modular packages called “nodes”. These nodes communicate with each other in a peer-to-peer, multi-lingual, and abstracted way through ROS. This allows for modular components that can be easily replaced and upgraded as needed. ROS toolsets also provide ways to record a synchronized log of data, called ROS bags, to play back at later times. Since the communication is abstracted, the other

ROS nodes (the algorithms) operate as if the data were coming live from sensors. This leaves users with a clean and flexible interface to develop their software. Due to the widespread usage of ROS in robotics, many low-level drivers are already provided online either from individual contributors or direct from sensor manufacturers. For this evaluation, the CAN communication, Data Base CAN (DBC) parsing/conversion, Mobileye driver, Webcam driver, and GPS driver were all already available through ROS, which significantly sped up the development process.

Due to the rapid growth of automated and autonomous systems, there became a need for perception toolsets to prevent “reinventing the wheel” for every new autonomous system [8], [24]. While the exact implementation of perception pipelines varies wildly, the overall structure remains the same. This structure, along with a variety of MOT/fusion components, is what MathWorks has developed in their toolsets for Automated Driving and Sensor Fusion. These tools were envisioned to be “highly configurable and extensible and that will make it easy to share and reuse perception algorithms across researchers and developers, teams, and even organizations” [24].

4.2.2 MathWorks Configuration

The following sections describe the MathWorks model that was developed to convert, fuse, and track objects, as shown in Figure 18.

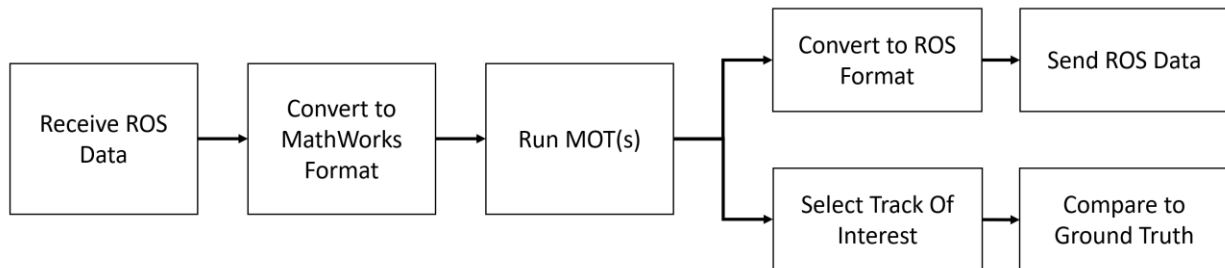


Figure 18: MOT Pipeline

To receive custom ROS messages in MathWorks, the first step is to set up the MATLAB/Simulink environment to recognize those custom messages using the steps from the MathWorks documentation in [69]. Next, the sensor data must be converted into the MathWorks “objectDetection” format, which is compatible with their Autonomous Driving and Sensor Fusion toolboxes. Once this is complete, theoretically any MOT/fusion element within the toolboxes should be able to be used, tested, and tuned without other changes to the model.

The “objectDetection” format has built-in support for position and velocity in XYZ along with a noise matrix to be populated with variances from the position and velocity measurements.

Table 6 shows which parameters are present on the sensors in this evaluation:

Table 6: Data Given From Sensors

	Position			Position Std Dev, σ			Velocity			Velocity Std Dev, σ		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
	Radar	✓	✓	✓	✓	✓	-	✓	✓	-	✓	-
Mobileye	✓	✓	-	-	-	-	✓	-	-	-	-	-

The “objectDetection” format expects measurement updates in the format: $[D_x D_y D_z V_x V_y V_z]$.

The sensor detections were then converted into that format. Unmeasured states were populated with 0’s in addition to large variances in the corresponding parts of the noise matrix, signifying a high degree of uncertainty. Variances for measured states without a sensor-reported variance were set to educated guesses after visual observation of the data. The sensors were each assigned a

unique index to comply with MathWorks format. This conversion also includes any additional calibration or offsets that need to be performed; in this case, there were only differences in angular calibration between the two sensors. This completed the conversion to the objectDetection format and allowed the detections to be fed into a Global Nearest Neighbors (GNN) Central-Level Track processing approach.

4.3 Issues and Considerations

A variety of issues were faced when trying to obtain good results in the final multi-object tracker. Some key issues are discussed in the following sections.

4.3.1 Ground Truth Considerations

As mentioned, OxTS ground truth is currently only available for one vehicle so dynamic two-vehicle testing is not available. Additionally, since only one target vehicle has location data, the full ground truth of the scenario is not available. This makes it difficult to compare the tracked data to ground truth to check for missed detections, track identification flipping, etc. It specifically precludes the use of track metrics that account for track switching, etc., and all-encompassing metrics like OSPA. OSPA is composed of three components: localization error, cardinality error, and labeling error which each operate over a list of tracks and truths to find the overall OSPA value of the system at that time. However, in this evaluation, only one truth track is known which means OSPA wrongly assumes other tracks are missed or false tracks [61]. This precludes the use of these metrics and means the testing and evaluation of the current setup is wholly based on single-track performance.

4.3.2 Time Alignment

Comparing ground truth data to sensor data requires proper time alignment. However, the OxTS system uses GPS atomic time while the sensor data is recorded using the local computer

time, which is unlikely to be synchronized. To alleviate this, a GPS unit was connected to the TANK computer and a software driver was created to record the time offset for corrections to be applied in the analysis phase. In this evaluation, it averaged a 0.3 s offset, which would have resulted in over 1 m of incorrectly attributed error even when driving slowly at 10mph.

4.3.3 Point Target Tracking Limitation

Both the vision and radar sensors perform some sort of internal pre-processing before reporting their detections; this is known since they report point targets even though the sensors receive extended object data. Presumably, this is both for convenience and to conserve bandwidth since reporting all data over CAN bus is infeasible. Since the objects are only reported as point objects, it precluded the use of Extended Object Tracking (EOT) methods. Therefore, conventional MOT methods were used instead.

4.3.4 Real-World Complications

When testing in the real world, there are sources of error that may not be accounted for, sources of error that present more error than anticipated, or even unknown sources of error. Some examples are imperfect calibration, the impact of material type on reflections, the impact of location on reflections, preprocessing techniques proprietary to the sensor manufacturer, weather effects, non-flat roads pitching and rolling the vehicle & sensors, EMF, etc. For instance, during some test runs when preparing for this evaluation the radar would return multiple observations per vehicle even though it never returned more than one when the target vehicle was in motion. This implies the issue is related to the speed of the vehicle, which matches the understanding that radar has difficulty detecting static objects. Either way, this caused the tracker to have difficulties due to its point object assumption, thus causing jumps in position between the extraneous observations for the same target vehicle. This is not believed to be an issue on the road since vehicles will

primarily be in motion; additionally, while multiple detections from a stopped vehicle would decrease tracker performance it should not be a safety concern as the vehicle controller should be observing the track that presents the most danger (i.e., the closest vehicle in that lane).

Another issue that was observed in testing was an offset between the sensor detections and ground truth. The ground truth assumed a reference point of the back of the target vehicles' rear bumper, however, the sensors did not. The radars' reference point was further in and is most likely because the rear bumper was plastic, which does not show up well in radar reflections. The Mobileye data was also further in and is likely because the rear bumper was not the only factor in the system estimating distance. Since the evaluation was intended for the tracker and not for the sensor detections, this offset was removed to make the ground truth line up with the sensor data.

4.4 Experiment

The following section analyzes test data from a real-world automated vehicle using MOTs from the MathWorks toolsets. Note that since this analysis is only based on single track accuracy, the proper detection or track needs to be selected for comparison against ground truth. This was done by filtering the sensor data to only the measurement closest to the ground truth at each timestep.

4.4.1 Test Setup

The test setup is shown in Figure 19. Due to the ground truth constraints discussed in Ground Truth and Ground Truth Considerations, the setup required a stationary target vehicle with a long driving corridor to approach it. To accommodate this, the setup consisted of a target vehicle placed at one end of a parking lot and the host vehicle being placed 40m away at the other end. The host vehicle accelerates to 10 mph before stopping behind the target vehicle. Before the test, the OxTS is calibrated, warmed up, and the static position of the target vehicle is noted in the data.

As the test runs, the sensor data is recorded into a ROS bag file, which allows the data to be played back later in a time-synchronized manner. The ground truth data and sensor data files are then transferred to other computers for detailed analysis and experimentation with the MathWorks tools.

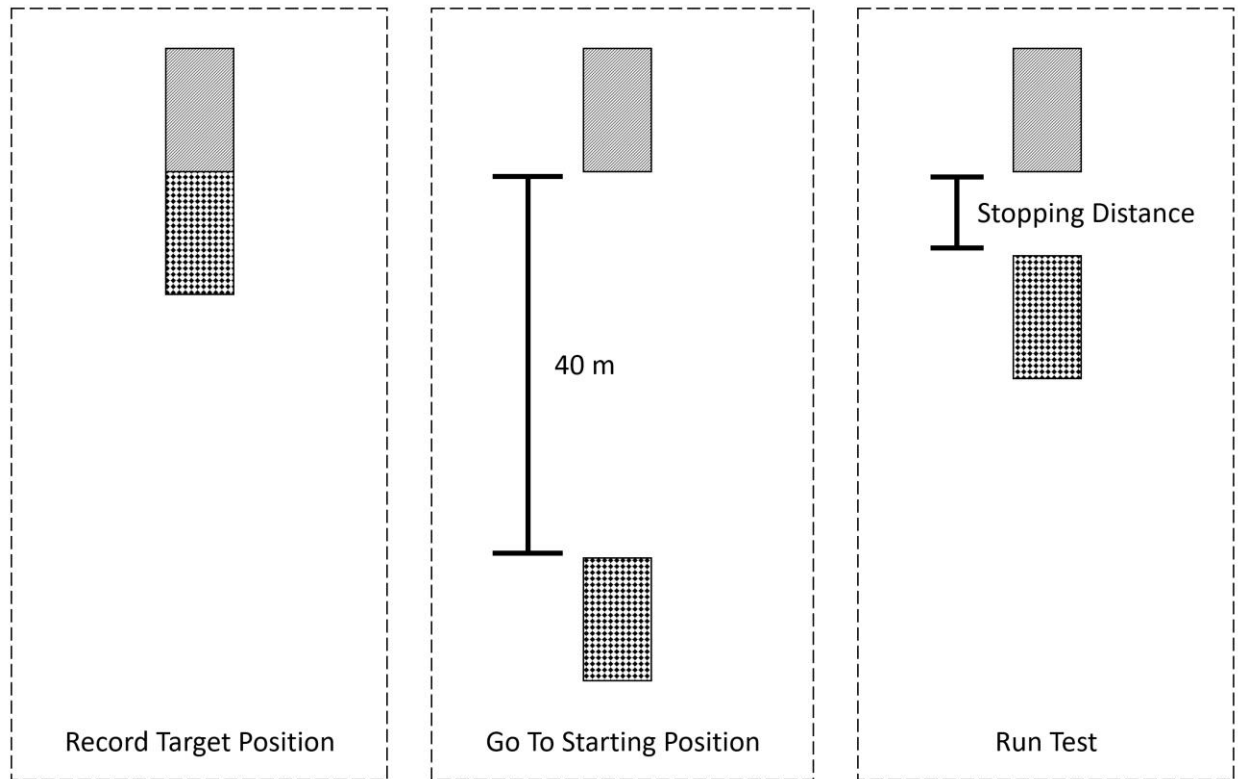


Figure 19: Scenario Description

4.4.2 Detection Evaluation

Before any analysis or comparison can occur, coordinate transforms must be applied to the detections (illustrated in Figure 13) and to the ground truth (discussed in section 4.1.3 Ground Truth) to allow for direct comparison of results. This includes angular offsets between the sensors and position offset between the ground truth and target vehicle. The position offset is applied, in part, to compensate for the final stopping distance between the vehicles. This is done to make a 0 m offset represent the end location for ground truth. If this were not performed, the positional error

would be inflated by the same amount as the stopping distance, which does not represent the true accuracy of the tracking solution. After coordinate transforms, the ground truth and sensor data are time-aligned so the data can be compared directly. After those corrections, the raw detections vs ground truth is shown in Figure 20, Figure 22, and Figure 23. As a reminder, the Mobileye does not measure lateral velocity, so it is excluded from that comparison.

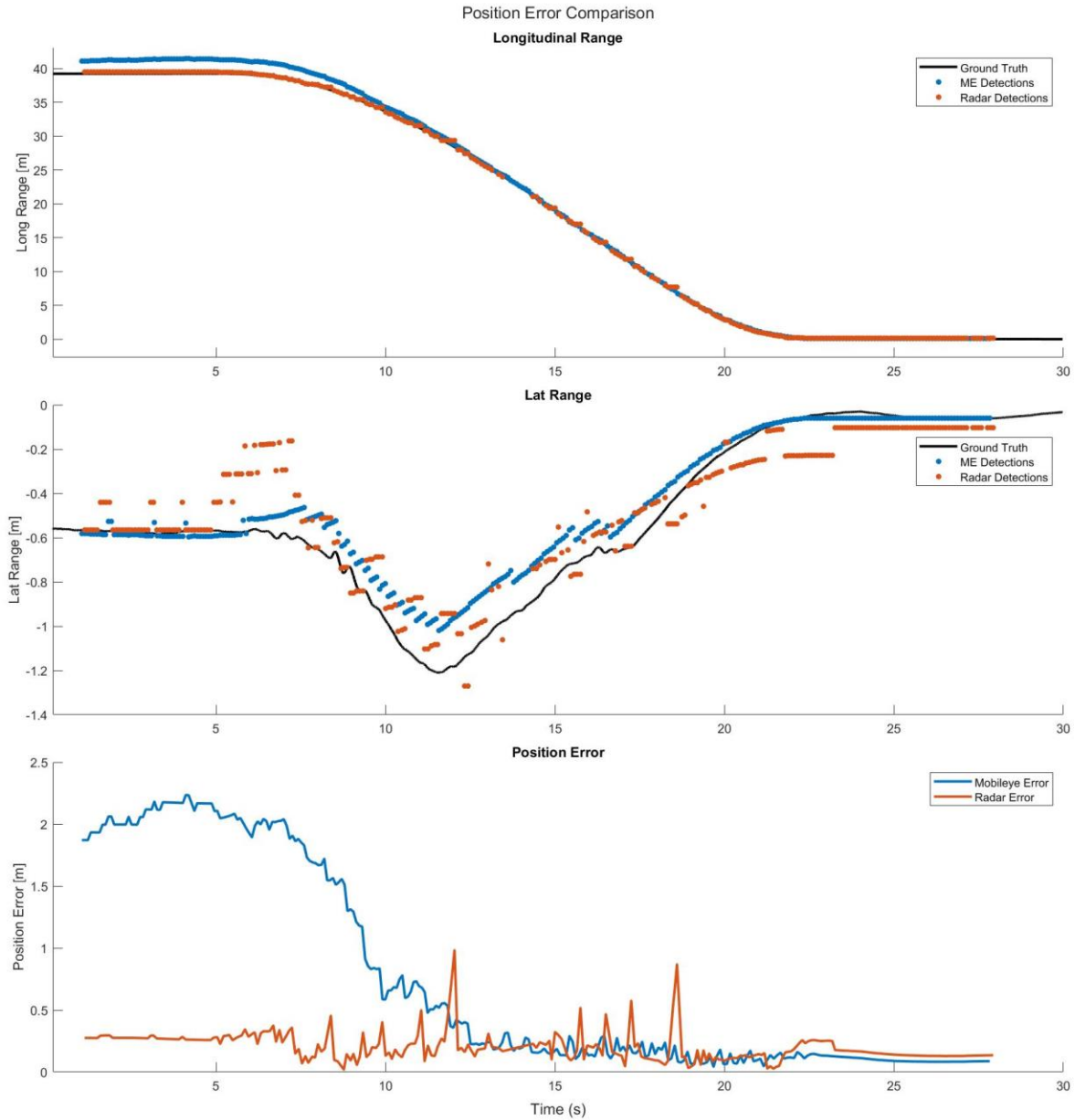


Figure 20: Position Error from Detections

Figure 20 compares sensor data and ground truth data of the relative range between the host and the target vehicle. The positional RMSE values are shown in Table 7. The results match expectations. The radar performed significantly better than the Mobileye for longitudinal position RMSE (1.14m vs 0.20m, respectively). The Mobileye performed much worse at the beginning of the scenario with over 2 m of error then dropped below 0.5 m as the scenario progressed. This is

expected since vision sensors can only estimate the vehicle distance rather than measure it directly; at larger distances, fewer pixels represent the entirety of an object, so a higher error rate is to be expected. The Mobileye performed better at measuring the lateral position of the target vehicle than the radar. This is visually apparent in Figure 20 whereas numerically the RMSE is 0.11 m for the Mobileye vs 0.14 m for the radar.

Table 7: Detection Position RMSE

	ME	Radar
Long Pos RMSE	1.14 m	0.20 m
Lat Pos RMSE	0.11 m	0.14 m
Total Pos RMSE	1.14 m	0.24 m

There are a few points of interest in this data. First, in the lateral range section of Figure 20, the radar jumps between values, which is especially apparent in the first 10 seconds of the test run. Even though the radar is likely processing the results at a higher resolution than this, the resolution of the sensor appears low (or “jumpy”) due to the limited resolution of the CAN bus. While lower accuracy could be due to radar having difficulties observing lateral range, this jumping is due to the Bosch CAN bus protocol only supporting this limited resolution.

Figure 21 isolates the radar data for clarity. At 14 seconds, there is a gap in the radar data caused by consistency bit errors for receiving the radar message. The error is believed to be due to a temporary CAN bus issue and is later shown to not impact the tracking performance. Next, there are spikes of radar position error shown in Figure 20 from around 7 to 17 seconds which are caused by radar detections holding the same value over multiple iterations, as shown in Figure 21. This could be the result of the radar latching the previous detection value when having difficulty detecting the vehicle due to it being stationary.

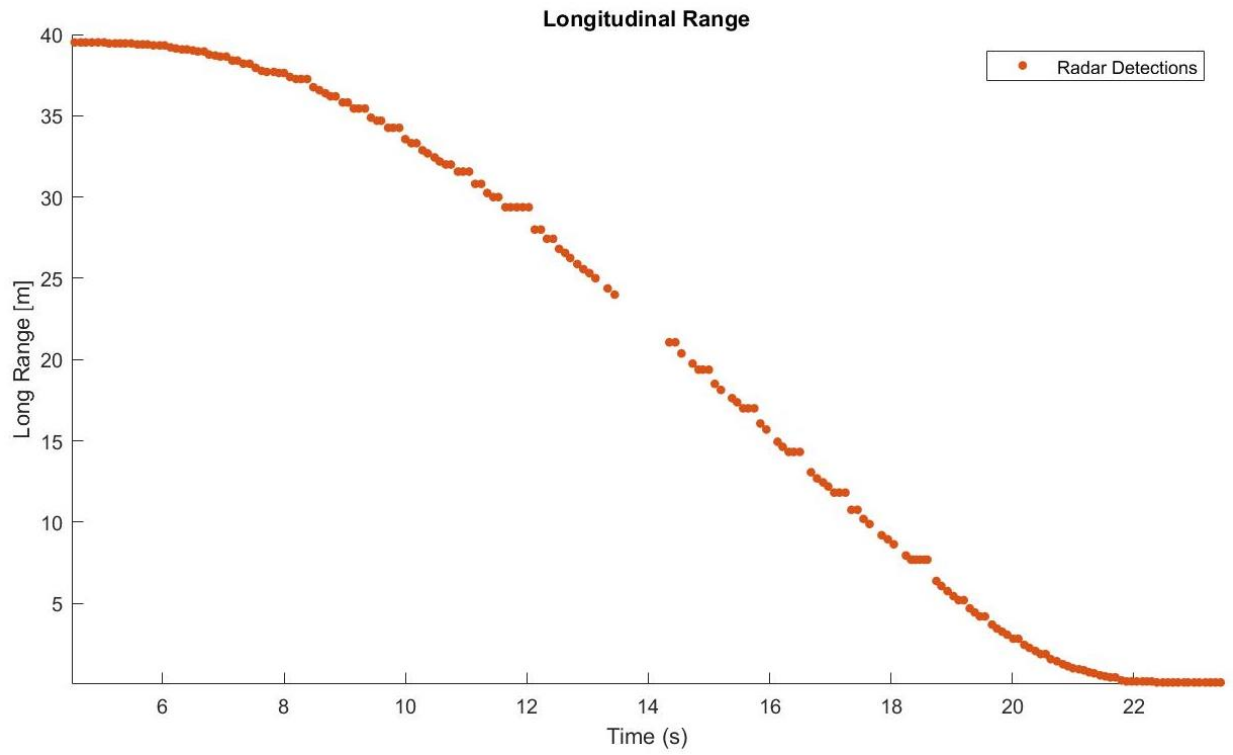


Figure 21: Radar Detections

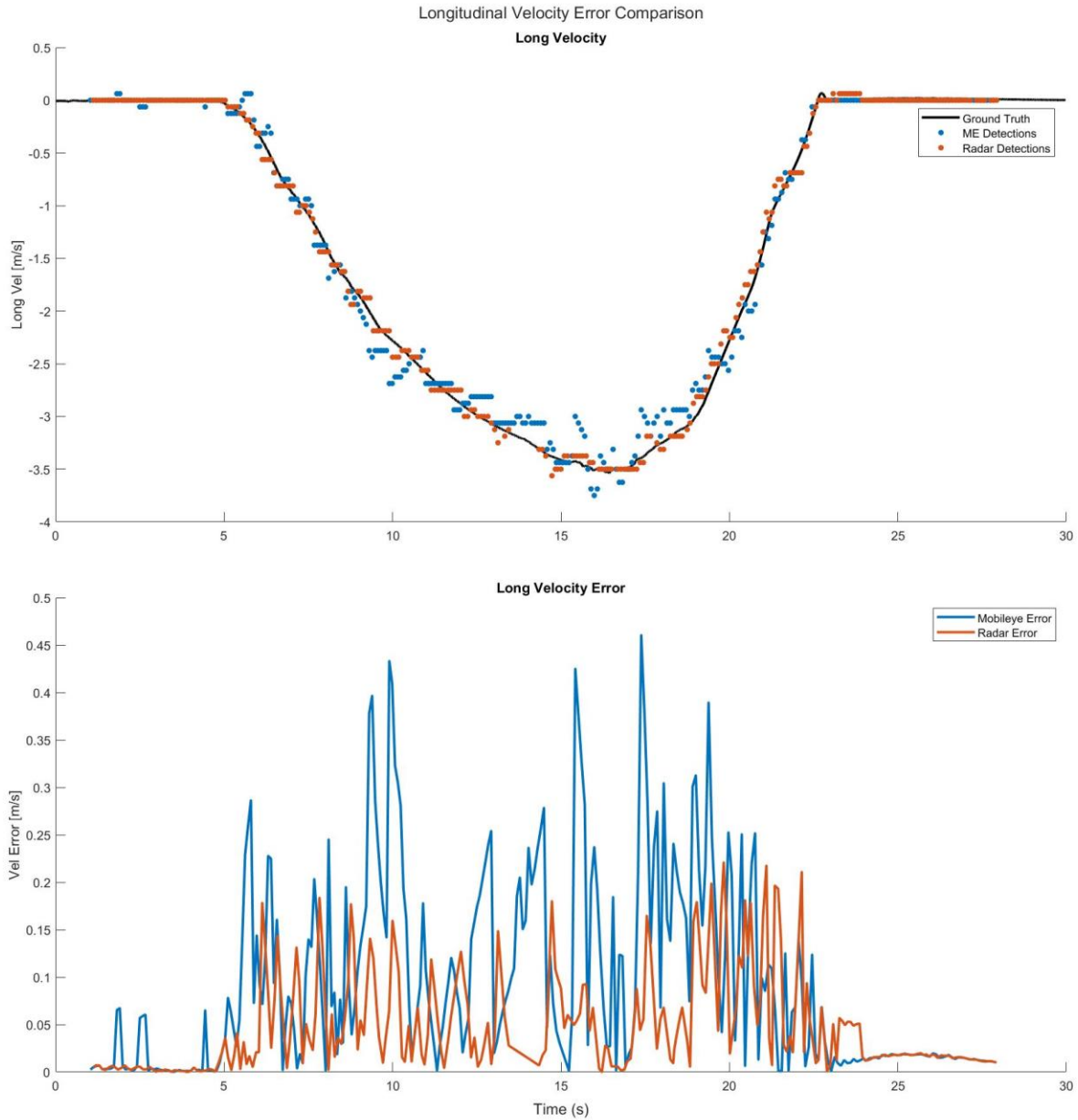


Figure 22: Longitudinal Velocity Error From Detections

Figure 22 compares the relative velocity of the target vehicle between the sensor measurements and the ground truth. Table 8 shows the relative longitudinal velocity RMSE values. As expected, the radar is much more accurate than the Mobileye since it actively measures the velocity of the object. The radar has half the RMSE of the Mobileye at 0.07 m/s vs 0.14 m/s. Note that the velocity values, and error, appear to be jumping between values. Applying a filter, such as

a Kalman filter within an MOT, should smooth this out and result in a lower error and is discussed later in this section.

Table 8: Longitudinal Velocity RMSE

	ME	Radar
Long Vel RMSE	0.14 m/s	0.07 m/s

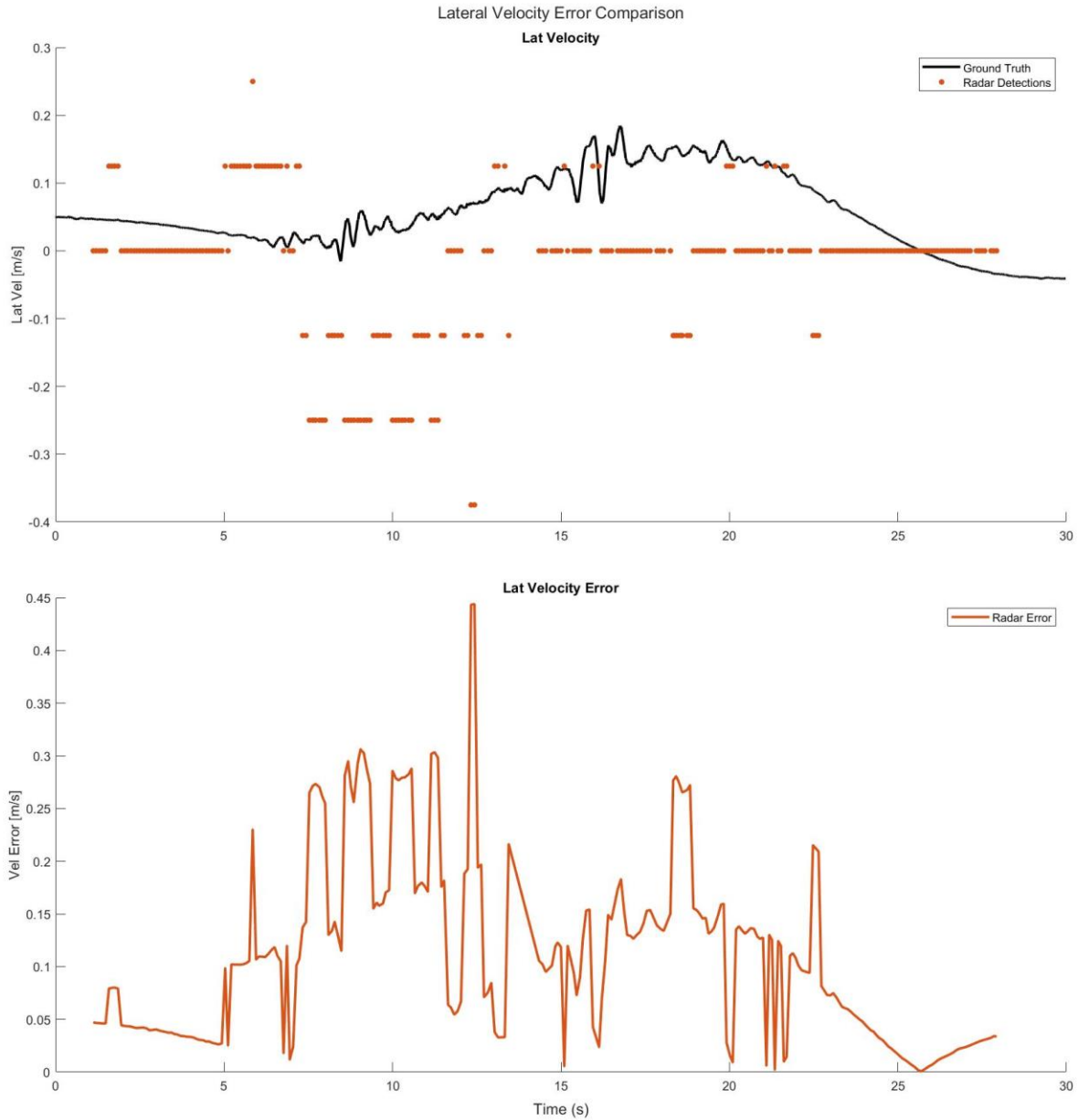


Figure 23: Lateral Velocity Error from Radar

Figure 23 compares the lateral position of the sensor measurements vs ground truth. Mobileye does not measure lateral velocity, so it is excluded from this discussion. Like the previous lateral range discussion, the lateral velocity also jumps between values caused by limited CAN bus resolution. The RMSE of the radar lateral velocity is 0.14 m/s but should have a strong benefit from filtering to smooth out the larger value jumps.

Table 9: Detection Lateral Velocity RMSE

	ME	Radar
Lat Vel RMSE	N/A	0.14 m/s

4.4.3 GNN Central Level Tracker Evaluation

A Central Level Track processing methodology was used with a Global Nearest Neighbors (GNN) assignment algorithm. This was able to assign the radar and vision detections to the same track to result in better state estimation than if only a single modality was used. The overall results and error comparison is shown in Figure 24, Figure 26, and Figure 27.

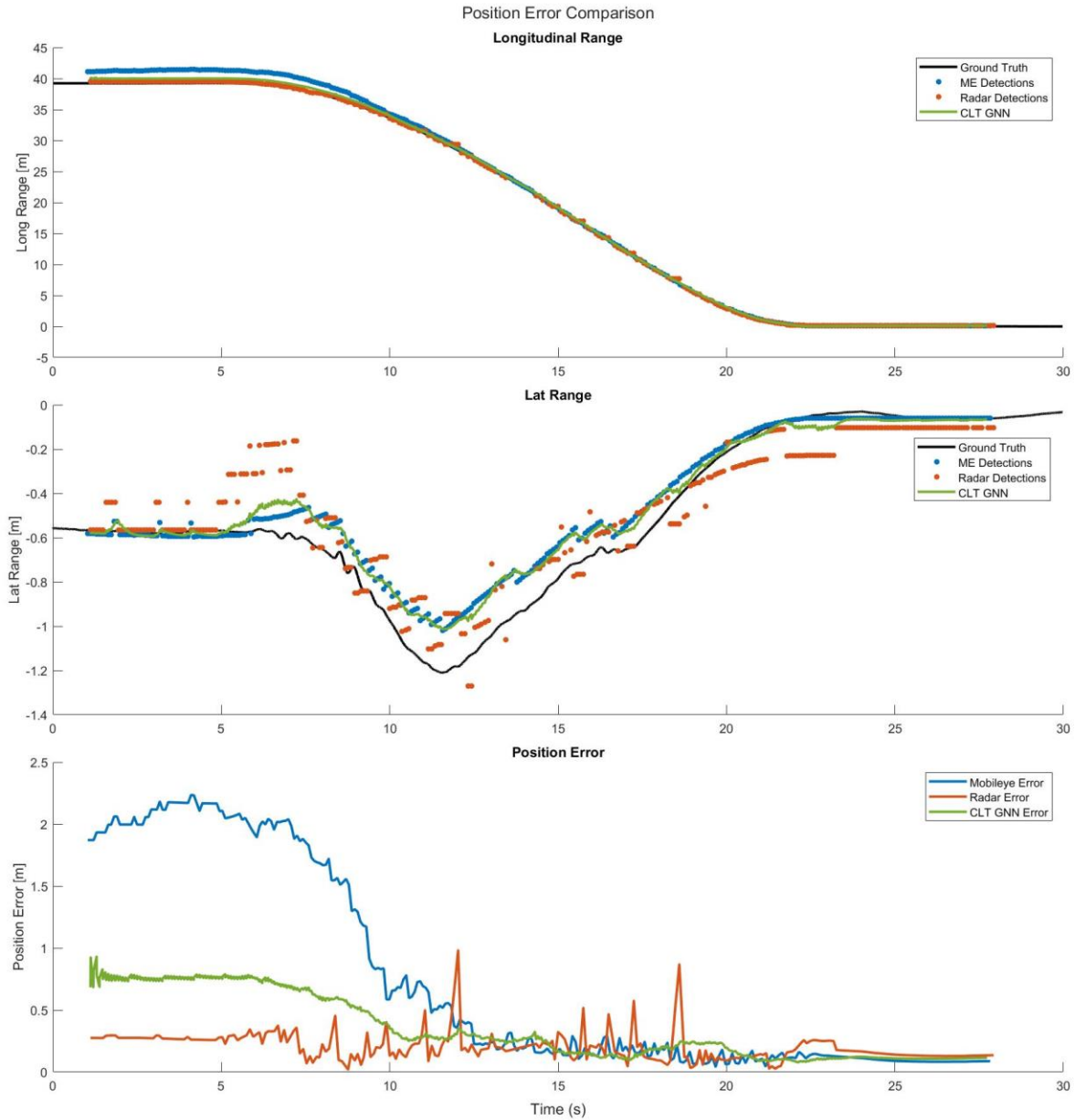


Figure 24: Position Error from Central Level Tracker GNN

Figure 24 compares the position performance of the tracker vs sensor data and ground truth. The RMSE comparison is shown in Table 10. Figure 25 illustrates the tracker forming the best estimate based on the two detections; it specifically illustrates how the lower longitudinal position noise of the radar results in the tracker assigning it a higher weight than the Mobileye. This is numerical confirmed as the tracker has 0.42 m longitudinal position RMSE, which is between the

0.20m of the radar and 1.14m of the Mobileye. The tracker performed better than both sensors for lateral position RMSE with 0.10m vs 0.11m and 0.14m. Visually, the tracker looks like it followed the lateral Mobileye data more closely than the radar and filtered out some of the spikes. Overall, the tracker is shown to have a worse RMSE than radar, 0.44m vs 0.24m, but with most of that error coming from the beginning of the test. The tracker is visually more consistent than the radar and Mobileye and contains no large spikes.

Table 10: Tracker Position RMSE Comparison

	ME	Radar	CLT GNN
Long Pos RMSE	1.14 m	0.20 m	0.42 m
Lat Pos RMSE	0.11 m	0.14 m	0.10 m
Total Pos RMSE	1.14 m	0.24 m	0.44 m

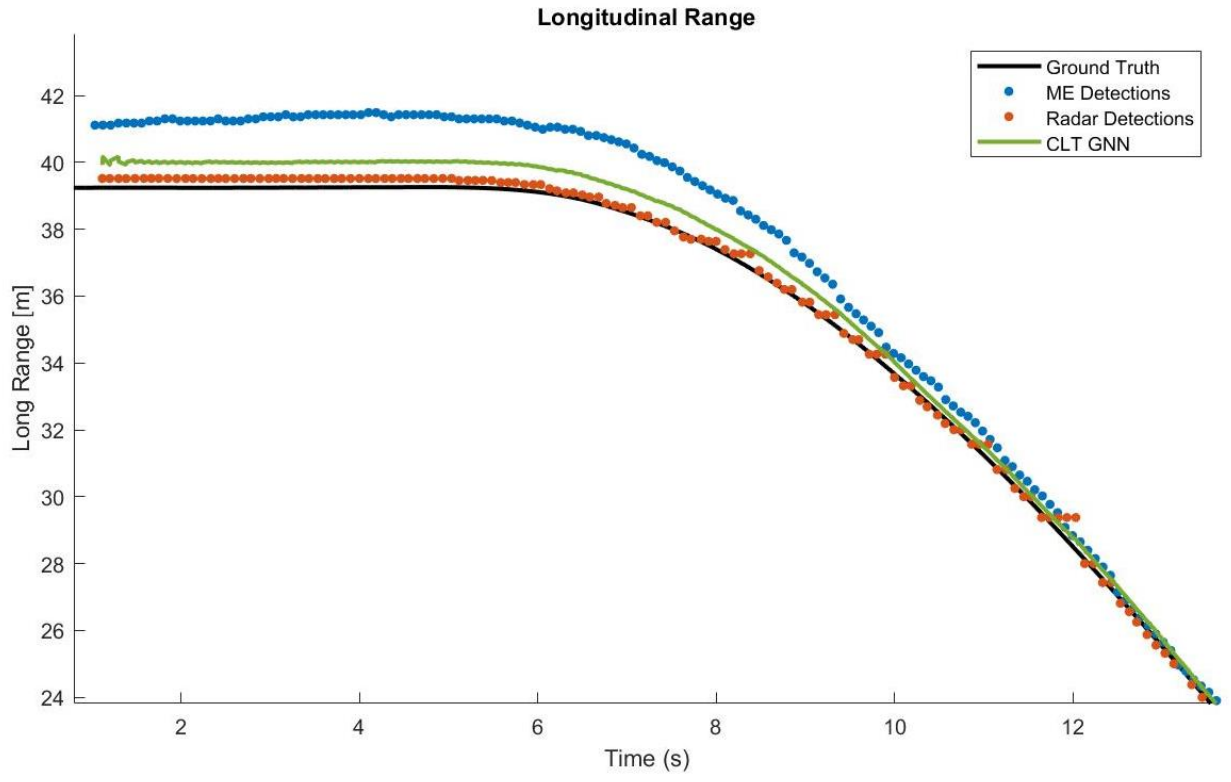


Figure 25: Tracker at Start

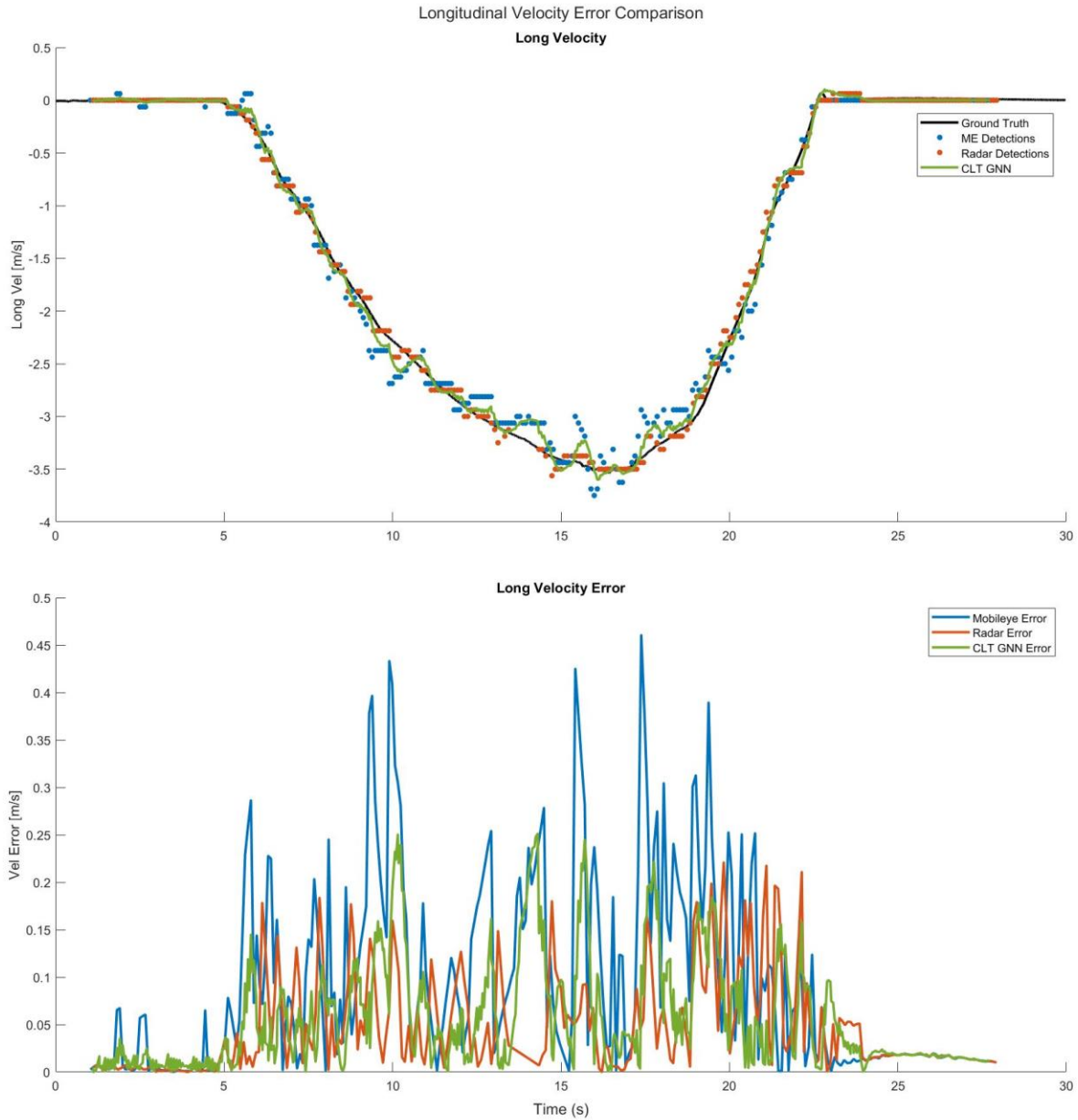


Figure 26: Longitudinal Velocity Error from Central Level Tracker GNN

Figure 26 compares the relative velocity performance of the tracker vs the sensor data and ground truth. Visually, it appears to account for both sensors and follows the overall trend of the ground truth. While the tracker error still “swings”, the swings are smoother than the abrupt jumps in the sensor data. The RMSE comparison is shown in Table 11. The radar has half the error of the

Mobileye and the tracker remains very close to the radar RMSE. There is only a 1 cm/s RMSE difference between the two as the tracker accommodates the Mobileye measurements.

Table 11: Tracker Longitudinal Velocity RMSE

	ME	Radar	CLT GNN
Long Vel RMSE	0.14 m/s	0.07 m/s	0.08 m/s

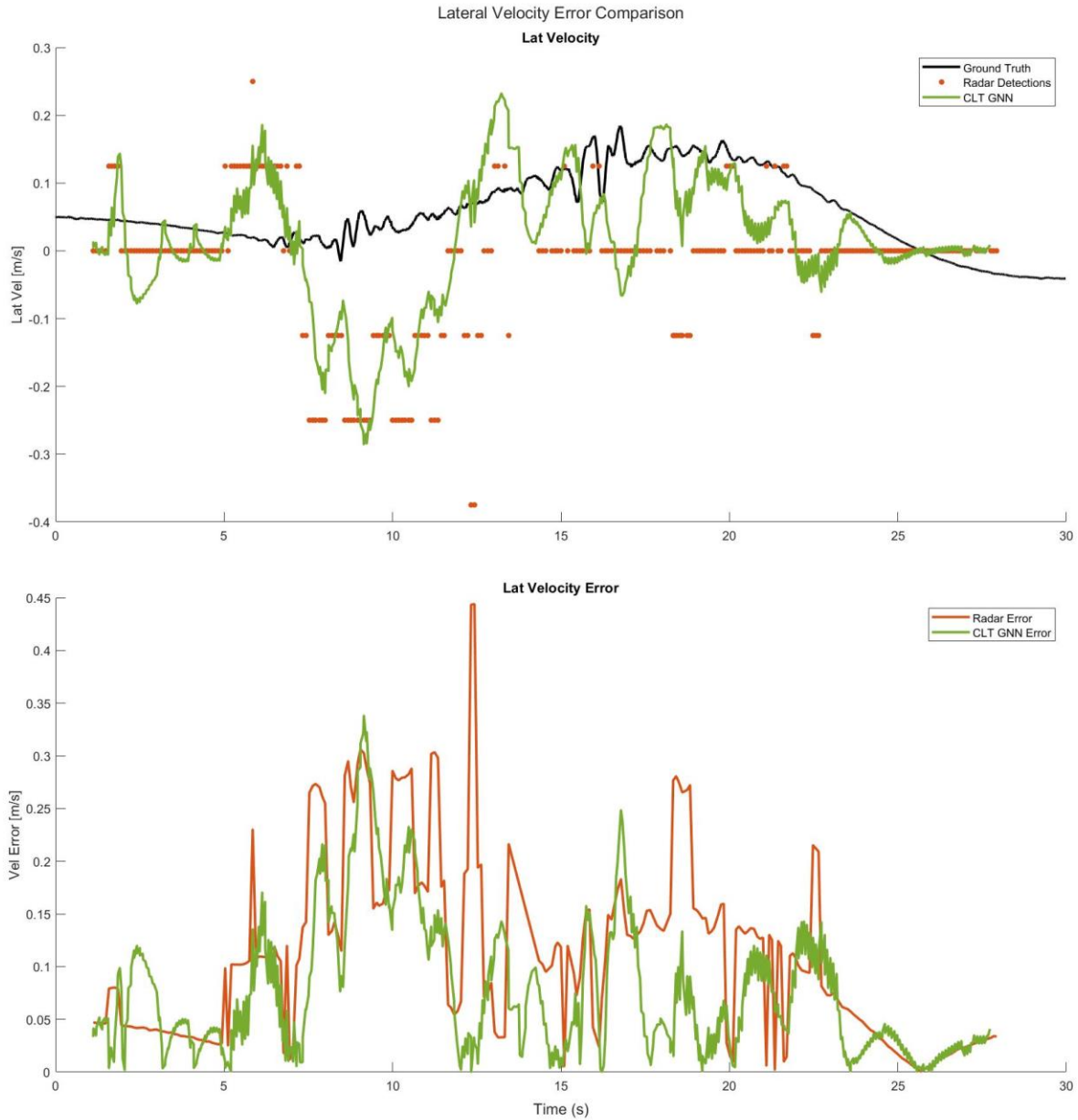


Figure 27: Lateral Velocity Error from Central Level Tracker GNN

Figure 27 compares the lateral velocity between the tracker, radar data, and ground truth. The tracker smooths out the jumping from the raw data and results in a lower overall RMSE of 0.10 m/s compared to 0.14 m/s for the radar, as shown in Table 12.

Table 12: Tracker Lateral Velocity RMSE Comparison

	ME	Radar	CLT GNN
Lat Vel RMSE	N/A	0.14	0.10

4.5 Tracking Toolset Review

Using a COTS toolset such as this has some distinct advantages and disadvantages. First, it has a variety of premade trackers and tracker components that can run out-of-the-box. Due to the interfaces MathWorks developed, these trackers and components can be easily swapped out to test a variety of configurations with a single set of data making it great to prototype and tune, especially when coupled with the various plotting and display tools available in MATLAB. The added benefit of code-generation support allows users to get the benefits of a good prototyping and analysis environment while also being able to directly deploy that model to the final device. If the AV group is using MathWorks tools elsewhere, like with vehicle controls, it could be more straightforward to integrate the different systems and share knowledge. MathWorks tools are paid products which is a financial con but has the benefit of support plus good documentation and resources. Another con of any framework is that users are forced to match their setup. MathWorks tools provide options for custom functions such as association, cost, and state transition, but it is not always practical or even feasible to change underlying code to suit a different need/purpose or fix a bug if one were found. Another downside of the MathWorks toolset is that it requires users to know or learn MATLAB and/or Simulink. Additionally, the full system must either be written in those languages or interface layers must be written to convert the data between different languages.

5 Conclusions and Future Work

This thesis has presented two main contributions: 1) a comprehensive mapping for the field of Multi-Object Trackers (MOTs) with a specific focus towards Automated Vehicles (AVs) and 2) a real-world evaluation of an MOT developed and tuned using COTS software toolsets. The mapping section provided a wide background on the use of MOTs in AVs, including definitions, considerations, breakdowns, common examples, extensions to EOT, MOTs with multiple sensors, and common metrics used to compare MOT performance. The second contribution used data from a real-world AV and ground truth system to evaluate the performance of an MOT using MathWorks toolsets, namely the Sensor Fusion and Tracking toolbox. These toolsets are designed to solve implementation difficulties with MOTs such as providing well-designed interfaces and a library of various trackers, tracker components, and tracker metrics. This setup allows trackers, components, and evaluation methods to be seamlessly swapped out to try new configurations. Additionally, the inclusion of Simulink blocks that support code-generation allows the entire tracking and fusion pipeline to be compiled and deployed directly to a target device.

Future work could include extension into a variety of areas. A trade study of different tracking toolsets and their capabilities could provide novel insight for researchers and practitioners. Evaluation of more complex trackers, such as Probability Hypothesis Density (PHD), or implementing trackers MathWorks toolsets do not currently support, such as Visual Object Trackers (VOT), could also provide value to the field. While Multi-Object Tracking is heavily researched, further surveys and literature reviews spanning the individual research fields could aid prospective researchers and practitioners in understanding the complexities of this large field. Further case studies using real-world data and reproducible/non-proprietary tracking toolsets could also aid in future development by providing perspective and reusable and consistent frameworks.

References

- [1] National Highway Traffic Safety Administration, “2018 Fatal Motor Vehicle Crashes Overview,” *Natl. Cent. Stat. Anal.*, p. 10, Oct. 2019.
- [2] “Zero Congestion With Self-driving Vehicles | General Motors.” <https://www.gm.com/> (accessed Mar. 05, 2021).
- [3] B. Schoettle, “Sensor Fusion: A Comparison of Sensing Capabilities of Human Drivers and Highly Automated Vehicles,” *Sustain. Worldw. Transp.*, p. 47, 2017.
- [4] K. Mizoguchi and S. Tsugawa, “Influence of in-vehicle music on driving: experimental results with a driving simulator,” in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, Jul. 2012, pp. 117–121, doi: 10.1109/ICVES.2012.6294296.
- [5] B. Dalton, D. Behm, and A. Kibele, “Effects of sound types and volumes on simulated driving, vigilance tasks and heart rate,” *Occup. Ergon.*, vol. 7, pp. 153–168, Jan. 2007.
- [6] J. S. Bartlett, “Cars, SUVs, and Trucks With the Best and Worst Braking Distances,” *Consumer Reports*. <https://www.consumerreports.org/car-safety/best-and-worst-braking-distances/> (accessed Jan. 19, 2021).
- [7] D. V. McGehee, E. N. Mazzae, and G. H. S. Baldwin, “Driver Reaction Time in Crash Avoidance Research: Validation of a Driving Simulator Study on a Test Track,” *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 44, no. 20, pp. 3-320-3–323, Jul. 2000, doi: 10.1177/154193120004402026.
- [8] D. F. Crouse, “The tracker component library: free routines for rapid prototyping,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 5, pp. 18–27, May 2017, doi: 10.1109/MAES.2017.160215.
- [9] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.
- [10] K. Granstrom, M. Baum, and S. Reuter, “Extended Object Tracking: Introduction, Overview and Applications,” *ArXiv160400970 Cs Eess*, Feb. 2017, Accessed: Oct. 29, 2020. [Online]. Available: <http://arxiv.org/abs/1604.00970>.
- [11] G. W. Pulford, “Taxonomy of multiple target tracking methods,” *Sonar Navig. IEE Proc. - Radar*, vol. 152, no. 5, pp. 291–304, Oct. 2005, doi: 10.1049/ip-rsn:20045064.
- [12] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A Survey of Autonomous Driving: Common Practices and Emerging Technologies,” *ArXiv190605113 Cs Eess*, Apr. 2020, doi: 10.1109/ACCESS.2020.2983149.
- [13] K. Cannons, “A Review of Visual Tracking,” p. 242.

- [14] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and Deep Trackers: Recent Visual Object Tracking Approaches and Trends," *ArXiv181207368 Cs*, Feb. 2019, Accessed: Jun. 19, 2020. [Online]. Available: <http://arxiv.org/abs/1812.07368>.
- [15] J. Jeong, T. S. Yoon, and J. B. Park, "Mean shift tracker combined with online learning-based detector and Kalman filtering for real-time tracking," *Expert Syst. Appl.*, vol. 79, pp. 194–206, Aug. 2017, doi: 10.1016/j.eswa.2017.02.043.
- [16] W. Luo *et al.*, "Multiple Object Tracking: A Literature Review," *ArXiv14097618 Cs*, May 2017, Accessed: Sep. 08, 2020. [Online]. Available: <http://arxiv.org/abs/1409.7618>.
- [17] S. Murray, "Real-Time Multiple Object Tracking - A Study on the Importance of Speed," *ArXiv170903572 Cs*, Oct. 2017, Accessed: Aug. 29, 2020. [Online]. Available: <http://arxiv.org/abs/1709.03572>.
- [18] "Short-Term Visual Object Tracking in Real-Time," p. 127.
- [19] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," *2016 IEEE Int. Conf. Image Process. ICIP*, pp. 3464–3468, Sep. 2016, doi: 10.1109/ICIP.2016.7533003.
- [20] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *ArXiv170307402 Cs*, Mar. 2017, Accessed: Aug. 28, 2020. [Online]. Available: <http://arxiv.org/abs/1703.07402>.
- [21] A. S. Jalal and V. Singh, "The State-of-the-Art in Visual Object Tracking," *Inform. Ljubl.*, vol. 36, no. 3, pp. 227–247, Sep. 2012.
- [22] M. Kristan *et al.*, "The Visual Object Tracking VOT2014 Challenge Results," Sep. 2014, pp. 191–217, doi: 10.1007/978-3-319-16181-5_14.
- [23] "Visual Tracking: An Experimental Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014, doi: 10.1109/TPAMI.2013.230.
- [24] E. H. Kivelevitch *et al.*, "Sensor Fusion Tools in Support of Autonomous Systems," presented at the AIAA Scitech 2019 Forum, San Diego, California, Jan. 2019, doi: 10.2514/6.2019-0384.
- [25] A. Scheel, C. Knill, S. Reuter, and K. Dietmayer, "Multi-sensor multi-object tracking of vehicles using high-resolution radars," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2016, pp. 558–565, doi: 10.1109/IVS.2016.7535442.
- [26] S. Reuter, B. Wilking, J. Wiest, M. Munz, and K. Dietmayer, "Real-Time Multi-Object Tracking using Random Finite Sets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 4, pp. 2666–2678, Oct. 2013, doi: 10.1109/TAES.2013.6621844.
- [27] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, "Forward collision warning with a single camera," in *IEEE Intelligent Vehicles Symposium, 2004*, Jun. 2004, pp. 37–42, doi: 10.1109/IVS.2004.1336352.

- [28] M. Obst, L. Hobert, and P. Reisdorf, "Multi-sensor data fusion for checking plausibility of V2V communications by vision-based multiple-object tracking," in *2014 IEEE Vehicular Networking Conference (VNC)*, Dec. 2014, pp. 143–150, doi: 10.1109/VNC.2014.7013333.
- [29] H. Zhang, Jinlong Yang, Hongwei Ge, and Le Yang, "An improved GM-PHD tracker with track management for multiple target tracking," in *2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*, Oct. 2015, pp. 185–190, doi: 10.1109/ICCAIS.2015.7338659.
- [30] M. Beard, B. T. Vo, and B. Vo, "OSPA(2): Using the OSPA metric to evaluate multi-target tracking performance," in *2017 International Conference on Control, Automation and Information Sciences (ICCAIS)*, Oct. 2017, pp. 86–91, doi: 10.1109/ICCAIS.2017.8217598.
- [31] B. J. Schachter, "Unification of automatic target tracking and automatic target recognition," in *Automatic Target Recognition XXIV*, Jun. 2014, vol. 9090, p. 909002, doi: 10.1117/12.2048595.
- [32] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, pp. 1–1, 2016, doi: 10.1109/TIP.2016.2520370.
- [33] C. Badue *et al.*, "Self-Driving Cars: A Survey," *ArXiv190104407 Cs*, Oct. 2019, Accessed: Mar. 13, 2021. [Online]. Available: <http://arxiv.org/abs/1901.04407>.
- [34] R. Mobus and U. Kolbe, "Multi-target multi-object tracking, sensor fusion of radar and infrared," in *IEEE Intelligent Vehicles Symposium, 2004*, Jun. 2004, pp. 732–737, doi: 10.1109/IVS.2004.1336475.
- [35] E. Mazor, A. Averbuch, Y. bar-shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *Aerosp. Electron. Syst. IEEE Trans. On*, vol. 34, pp. 103–123, Feb. 1998, doi: 10.1109/7.640267.
- [36] V. Lekic and Z. Babic, "Automotive radar and camera fusion using Generative Adversarial Networks," *Comput. Vis. Image Underst.*, vol. 184, pp. 1–8, Jul. 2019, doi: 10.1016/j.cviu.2019.04.002.
- [37] Tesla, "Upgrading Autopilot: Seeing the World in Radar," *Tesla*, Sep. 11, 2016. <https://www.tesla.com/blog/upgrading-autopilot-seeing-world-radar%20> (accessed Mar. 04, 2021).
- [38] E. Marti, M. A. de Miguel, F. Garcia, and J. Perez, "A Review of Sensor Technologies for Perception in Automated Driving," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 94–108, winter 2019, doi: 10.1109/MITS.2019.2907630.
- [39] Z. Wang, Y. Wu, and Q. Niu, "Multi-Sensor Fusion in Automated Driving: A Survey," *IEEE Access*, vol. 8, pp. 2847–2868, 2020, doi: 10.1109/ACCESS.2019.2962554.
- [40] B. Pueo, "High speed cameras for motion analysis in sports science," *J. Hum. Sport Exerc.*, vol. 11, pp. 53–73, Dec. 2016, doi: 10.14198/jhse.2016.111.05.

- [41] D. Schubert, N. Demmel, L. v Stumberg, V. Usenko, and D. Cremers, “Rolling-Shutter Modelling for Direct Visual-Inertial Odometry,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 2462–2469, doi: 10.1109/IROS40897.2019.8968539.
- [42] “General Motors Night Vision Technology,” *GM Authority*. <https://gmauthority.com/blog/gm/general-motors-technology/gm-safety-technology/gm-active-safety-technology/gm-night-vision-technology/> (accessed Mar. 05, 2021).
- [43] C. Lundquist and U. Orguner, “Tracking Rectangular and Elliptical Extended Targets Using Laser Measurements,” in *Proceedings of the International Conference on Information Fusion*, 2011, pp. 592–599.
- [44] X. R. Li and V. P. Jilkov, “Survey of maneuvering target tracking. Part I. Dynamic models,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1333–1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.
- [45] “How a Kalman filter works, in pictures | Bzarg.” <https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/> (accessed Jan. 19, 2021).
- [46] Q. Li, R. Li, K. Ji, and W. Dai, “Kalman Filter and Its Application,” in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Nov. 2015, pp. 74–77, doi: 10.1109/ICINIS.2015.35.
- [47] H. R. Künsch, “Particle filters,” *Bernoulli*, vol. 19, no. 4, pp. 1391–1403, Sep. 2013, doi: 10.3150/12-BEJSP07.
- [48] “Introduction to Assignment Methods in Tracking Systems - MATLAB & Simulink.” <https://www.mathworks.com/help/fusion/ug/introduction-to-assignment-methods-in-tracking-systems.html> (accessed Mar. 13, 2021).
- [49] “Multi-sensor, multi-object tracker using GNN assignment - MATLAB.” https://www.mathworks.com/help/fusion/ref/tracker_gnn_system_object.html (accessed Mar. 17, 2021).
- [50] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955, doi: 10.1002/nav.3800020109.
- [51] R. P. S. Mahler, “Multitarget Bayes filtering via first-order multitarget moments,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1152–1178, Oct. 2003, doi: 10.1109/TAES.2003.1261119.
- [52] M. Ester, H.-P. Kriegel, X. Xu, and J. Sander, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” 1996, p. 6.
- [53] K. Granström, L. Svensson, S. Reuter, Y. Xia, and M. Fatemi, “Likelihood-Based Data Association for Extended Object Tracking Using Sampling Methods,” *IEEE Trans. Intell. Veh.*, vol. 3, no. 1, pp. 30–45, Mar. 2018, doi: 10.1109/TIV.2017.2788184.

- [54] A. Bassett, D. Cicotte, and P. Currier, “Object Tracking Comparison for Automated Vehicles Using MathWorks Toolsets,” Apr. 2021, doi: 10.4271/2021-01-0110.
- [55] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” *ArXiv151202325 Cs*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.
- [56] “Track Point Targets in Dense Clutter Using GM-PHD Tracker in Simulink - MATLAB & Simulink.” <https://www.mathworks.com/help/fusion/ug/track-point-targets-in-dense-clutter-using-gm-phd-tracker-in-simulink.html> (accessed Mar. 17, 2021).
- [57] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, “Multisensor data fusion: A review of the state-of-the-art,” *Inf. Fusion*, vol. 14, no. 1, pp. 28–44, Jan. 2013, doi: 10.1016/j.inffus.2011.08.001.
- [58] S. Matzka and R. Altendorfer, “A comparison of track-to-track fusion algorithms for automotive sensor fusion,” in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Aug. 2008, pp. 189–194, doi: 10.1109/MFI.2008.4648063.
- [59] “Introduction to Tracking Metrics - MATLAB & Simulink.” <https://www.mathworks.com/help/fusion/ug/introduction-to-tracking-metrics.html> (accessed Apr. 01, 2021).
- [60] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, “A Consistent Metric for Performance Evaluation of Multi-Object Filters,” *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3447–3457, Aug. 2008, doi: 10.1109/TSP.2008.920469.
- [61] B. Ristic, B. Vo, D. Clark, and B. Vo, “A Metric for Performance Evaluation of Multi-Target Tracking Algorithms,” *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3452–3457, Jul. 2011, doi: 10.1109/TSP.2011.2140111.
- [62] “» EcoCar Mobility ChallengeAVTC I Advanced Vehicle Technology Competitions.” <https://avtcseries.org/ecocar-mobility-challenge/> (accessed Mar. 21, 2021).
- [63] “Mid-range radar sensor (MRR).” [https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/automatic-emergency-braking/mid-range-radar-sensor-\(mrr\)/](https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/automatic-emergency-braking/mid-range-radar-sensor-(mrr)/) (accessed Mar. 21, 2021).
- [64] “Mobileye Collision Avoidance System | Mobileye for Fleets,” *Mobileye*. <https://www.mobileye.com/us/fleets/products/mobileye-6-collision-avoidance-system/> (accessed Mar. 21, 2021).
- [65] “Mobileye Camera Development Kit.” <https://autonomoustuff.com/products/mobileye-camera-dev-kit> (accessed Mar. 21, 2021).
- [66] “RT3000 v3 » GNSS-aided inertial navigation system for automotive testing,” *OxTS*. <https://www.oxts.com/products/rt3000-v3/> (accessed Mar. 21, 2021).

[67] “Florida Permanent Reference Network (FPRN),” *FDOT*.
<https://www.fdot.gov/geospatial/fprn.shtm> (accessed Mar. 05, 2021).

[68] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” 2009, vol. 3, p. 6.

[69] “ROS Custom Message Support - MATLAB & Simulink.”
<https://www.mathworks.com/help/ros/ug/ros-custom-message-support.html> (accessed Mar. 21, 2021).