DETERMINING SPATIAL RELEVANCY OF OBJECTS FOR IMPROVED DEVELOPMENT OF MULTI-OBJECT TRACKING IN AUTONOMOUS DRIVING SYSTEMS

by

MAEN HAMMOD

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2023

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Osamah Rawashdeh, Ph.D., Chair
László Lipták, Ph.D.
Brian Dean, Ph.D.
Guangzhi Qu, Ph.D.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Osamah Rawashdeh, for his guidance, support, and mentorship throughout the process of writing this dissertation. His insights and expertise were invaluable in shaping the direction and content of my research. I would also like to thank my committee members, Professor László Lipták, Dr. Brian Dean, and Dr. Guangzhi Qu, for their valuable feedback and guidance. I am very grateful to my colleagues at Argo AI, Dr. Brent Tweddle (coauthor for the patent), Dr. Peter Car, Joseph Poczatek, Adam Sylvester, and Dr. Ratnesh Kumar, for their support and for many hours of discussions and valuable feedback on the ideas presented in this dissertation. I am deeply grateful to my mother for her love and support throughout my life and academic journey. I would also like to thank my brother, Dr. Sayah Hammod, for his love, support and encouragement. I am grateful to my wife Sarah and my children, Emily, and Daniel, for their unwavering love and support. I would also like to thank my brothers and sisters, and my in-laws, for their kindness and encouragement. Finally, I would like to dedicate this work to my late father, Hammod Hammod, who inspired me to pursue my passions and dreams. His guidance and support will always be with me.

ABSTRACT

DETERMINING SPATIAL RELEVANCY OF OBJECTS FOR IMPROVED DEVELOPMENT OF MULTI-OBJECT TRACKING IN AUTONOMOUS DRIVING SYSTEMS

By

Maen Hammod

Adviser: Osamah Rawashdeh, Ph.D.

The perception system for autonomous vehicles (AVs) typically outputs all the objects it can observe in a scene. This is significantly more objects than what the AV would interact with, and far more than any human driver focus on during a driving task. Validating the perception system on all the observed objects could penalize its performance based on objects that will never interact with the AV or affect its planned trajectory. This dissertation outlines a strategy for identifying a subset of objects, referred to as Spatially Relevant Objects (SRO), that the perception system must perform exceptionally well on. This is valuable for several reasons and has many applications. For example, it can be used to determine the set of objects that should be included in the verification dataset for the perception system and thereby have a more efficient development cycle. Additionally, when evaluating the perception system on the SRO subset, the computed metrics not only evaluate the performance but also consider the real-world safety of the perception system, and the results would heavily support the safety case arguments. Finally, determining spatially relevant objects using a representative dataset then plotting their footprints relative to the AV can help determine

and confirm the necessary sensing requirements and field of view coverage by prioritizing areas where SROs are most likely to appear.

This is done without ignoring the fact that the world contains objects of different classes with different kinematics and could behave in a non-compliant way. The preliminary finding of applying our system showed that we can measure the performance of the perception system using a subset that averages about 11% of the observed objects without compromising the safety of the AV.

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

TABLE OF CONTENTS - Continued

LIST OF FIGURES

LIST OF FIGURES - Continued

LIST OF FIGURES - Continued

LIST OF ABBREVIATIONS

SRO	Spatially Relevant Objects
LiDAR	Light ranging and detection.
DDT	Dynamic Driving Task
ODD	Operational Design Domain
MOT	Multi-Object Tracking
SAE	Society of Automotive Engineering
ADS	Autonomous Driving system
OEDR	Object and Event Detection Response
ABS	Anti-lock Braking System
SDS	Self-Driving System
FOV	Field Of View
IMU	Inertial Measurement unit
GPS	Global Positioning Systems
AVS	Autonomous Vehicle System
CAVS	Complimentary Autonomous Vehicle System
AVP	Autonomous Vehicle Platform
CNN	Convolutional Neural Network
NTSB	National Transportation Safety Board
IOU	Intersection Over Union
DA	Drivable Area

LIST OF ALGORITHMS

LIST OF TABELS

CHAPTER 1: AUTONOMOUS DRIVING SYSTEMS

1.1 Introduction

The way we commute, and travel is changing. For the first time since automobiles were massively produced for personal use, we are at the beginning of a significant change in the automotive and transportation systems. We are shifting to mobility services based on self-driving cars, which will be paid for per trip or membership. In the near future, many of us who do not like our commutes will no longer need to own or drive a car. Instead, we will rely on services that safely and conveniently use autonomous vehicles to take us where we want to go. Our transportation experience will be managed by partnerships between automotive manufacturers and technology companies, and we will no longer be required to shop for, finance, and maintain our own vehicles, or waste our time driving, looking for parking space, or filling gas. There will be less traffic congestion. And we will be able to choose between carpooling with others or paying more to have a customized autonomous vehicle depending on our trip needs.

Reserving a ride will happen with the touch of an app. The vehicles that arrive won't have a steering wheel or gas and brake pedals. They are enabled with 360-degree field of view sensing and technologies that do not get distracted or tired. This will guarantee the safety of not only their occupants but also other road users. All of this and the transportation cost us a fraction of what we ever did before [1].

Although in the past few decades there has been a huge effort by legislators, cities, and automotive manufacturers to reduce traffic deaths and integrate human-failing technologies [3] through active and passive safety, in 2016 alone there were more than 34,000 fatal crashes in the United States, resulting in 37461 fatalities (1.2 fatalities per 100 million miles traveled), not to mention the economic impacts. About 95% of them were estimated to be caused by a driver's error [2]. Despite the fact that the number of fatalities per mile on the road has fallen five-fold since the 1960s, it stayed quite the same over the previous decade. Improvement in road safety is being proposed by taking humans completely out of the loop and replacing the driving task with automation where sensors, computers, and software are combined into what we call a "virtual driver system". For this potentially transformative technology to be accepted by society, its first generation must be safer than (or at least as safe as) the human drivers of today.

The virtual driver's operating environment is highly unpredictable, with other nearby vehicles on the road potentially exhibiting inattention, fatigue, or other behaviors that violate traffic rules and driving etiquette. This makes it difficult to confidently assert that the system has successfully passed all validation gates and achieved safety goals. To address this challenge, we propose a data-driven framework that aids in evaluating the multi-object tracking (MOT) pipeline of self-driving systems. This framework helps to identify areas of the perception system that require testing, as well as the specific objects that the perception system must perform exceptionally well on. By triaging and correcting failures, the framework contributes to the overall safety and reliability of the self-driving system.

The rest of this chapter provides an overview of automated driving systems, including their hardware, software, and the background material necessary to understand the contribution of this dissertation. Chapter 2 summarizes related literature on multi-object tracking (MOT) metrics. The problem statement is discussed in Chapter 3, while Chapter 4 delves into the proposed framework for identifying spatially relevant objects, including the algorithms and the model details, the system architecture, and the implementation details. Chapter 5 presents the dataset used to evaluate our system and the evaluation results. Finally, Chapter 6 summarizes our contribution and suggests potential avenues for future improvement.

1.2 Automated Driving Systems

The Society of Automotive Engineering (SAE) J3016 [4] divides automated driving systems that perform part of the dynamic driving task (DDT) into six levels, ranging from no automation (Level 0) to full automation (Level 5), also known as autonomous driving systems (ADS) or self-driving systems (SDS). Additionally, it explains the relationship between these levels and the operational design domain (ODD), which define the operating range of the ADS by setting technical, geographic, and environmental parameters (e.g., maximum operating speed, weather condition, geonet, etc.), as well as the response of both the driver and the ADS for executing the DDT fallback in order to reduce the risk of crash when a trip should not or can't be completed (Minimal Risk Condition). Figure 1-1 shows Society of Automotive Engineers (SAE) levels of road vehicle autonomy [5]

1. **Level 0 (No Driving Automation)**: The driver performs the entire DDT including the Object and Event Detection Response (OEDR) as well as the DDT fallback, obviously for this level the Operational Design Domain is not applicable.
2. **Level 1 (Driver Assistance):** In limited ODD the automation driving system execute ODD-specific subtask of the DDT by providing either lateral (steering) or longitudinal (acceleration/deceleration) but not simultaneously, the driver is expected to perform the rest of the DDT, the Object and Event Detection Response subtask as well as the DDT Fallback. This includes systems like cruise control and anti-lock braking systems (ABS)
3. **Level 2 (Partial Driving Automation):** In limited ODD the automated driving system execute ODD-specific subtask of the DDT by providing both lateral (steering) or longitudinal (propulsion, and braking) under the supervision of the driver who also responsible of the Object and Event Detection Response subtask, as well as immediately performing DDT Fallback. Examples of this system are General Motors super cruise and Tesla Autopilot [2].
4. **Level 3 (Conditional Driving Automation):** In limited ODD the ADS executes - the entire DDT including Object and Event Detection Response subtask. The driver is responsible for performing the DDT fallback (within a defined time frame) upon receiving a notification issued by the ADS to take over as well as

when failures in other vehicle systems occur. Example of Level 3 is traffic jam system that performs the driving task in dense traffic on a freeway but requires the driver to take over at ODD exit for example upon exiting the freeway.

5. **Level 4 (High Driving Automation):** In limited ODD the ADS executes the entire DDT including OEDR subtask as well as DDT fallback without any expectation that the user will need to take any action. Typically, this type of system requires supporting infrastructure (e.g., maps, remote guidance, connectivity), examples of this system are Waymo Driver [7] and Argo AI SDS system [6].
6. **Level 5 (Full Driving Automation):** In unconditional (unlimited) ODD the ADS executes the entire DDT including OEDR subtask as well as DDT fallback without any expectation that the user will need to take any action.

1.3 The Significance of the Operational Design Domain

As mentioned above, levels 1 through 4 are ODD-Specific and expressly contemplate ODD limitations. The ODD captures the complexity of the driving environment by defining widely ranging variables and attributes that include road types, operating weather conditions, lighting conditions, geographical constraints, and certain road features like the existence of lane markings and maximum road curvature, not to mention many. The ADS under these levels that are performing part of or the complete DDT is highly dependent on the ODD [5]. For example, a specific implementation of a level 4 SDS may be intended to operate in the city and on roads with a maximum speed of 45 mph, but its sensor suite will not be enough for higher speeds.

Although the framework we present in this dissertation can be used in the verification process for any perception-capable ADS (regardless of the autonomy level), some of the ODD attributes highly impact the configuration of this framework. For example, lane and road geometry affect our world model, and maximum road speed affects our assumptions about the behavior of the road users. Figure 1-2 illustrates the orthogonality of ODD relative to levels of driving automation.

1.4 Autonomous Driving System Hardware Architecture

For Level 4 autonomous systems, safety should be the number one priority and the driver of the self-driving system (SDS) architecture. To achieve the safety goals, independence and redundancy in the hardware, software, and sensing suite are required. Figure 1-3 illustrates a high-level diagram of the Argo self-driving system architecture (4th generation) [6], which is the same system used to collect the dataset used for developing and proofing the technology presented in this dissertation, the following section provides a quick overview of the main part of this architecture.

1.4.1 Multi-modal Sensor Technology

To safely navigate the world, an autonomous vehicle is required to constantly see 360 degrees around itself. To achieve this, it relies on different types of sensors that each detect the world in different ways (systematic diversity) and also on more than one sensor observing the same FOV (property of independence). This diversity and redundancy in the sensor architecture are necessary for handling different edge cases introduced by the

environment. For example, the dynamic range of a camera sensor could be insufficient to handle exiting a dark tunnel on a sunny day.

A comparison between different types of perception sensors is provided in [8]. Also, a summary of the main features of sensors used in perception systems of the AV is provided in Figure 1-5. Both illustrate how systematic diversity overcomes individual sensor type weaknesses to achieve safety goals. For localization and perception, Argo SDS uses the following sensors: short- and long-range LiDAR, far-field high-resolution cameras, near field cameras, radars, global positioning systems (GPS), inertial measurement units (IMU), wheel speed sensors, and finally microphones for siren detection.

1.4.2 The On-board Computers

Redundancy and independence in the computing systems of autonomous vehicles are also required. Therefore, at least two independent on-board computers will be used in level 4 autonomous systems. One is responsible for performing the DDT and the second is responsible for performing the DDT fallback [5]. In Figure *1-3* [6], you can see that these correspond to the Autonomous Vehicle System (AVS) and the Complimentary Autonomous Vehicle System (CAVS). The AVS hardware is designed to satisfy the needs of the self-driving software subsystems, which include functions such as collecting and processing data from sensors, localization and mapping, object detection, tracking and prediction, motion planning and control, and diagnostics, as well as the infrastructure (OS, communication layer, etc.) that is required to run these functions. On the other hand,

the CAVS, which runs at the same time with the AVS as a backup system, is ready to take over if the AVS enters a degraded state or if the AVS is not able to avoid a potential impact. In that situation, the CAVS intervenes and executes the fallback trajectory to bring the vehicle safely to a stationary state. Both the AVS and the CAVS communicate with the Autonomous Vehicle Platform (AVP) through different communication channels (CAN/Ethernet) for the purpose of reading the health status and readiness of different parts of the AVP as well as providing the steering, propulsion, and braking commands.

1.5 Self-driving Systems Software

The software that performs the DDT and DDT fallback consists of multiple subsystems across different abstraction layers. This includes software infrastructure (HW abstraction layer, operating system, filesystem, communication, etc.), embedded software, vehicle interface, safety monitors, and autonomy software, among others. SEA J3131 [10] provides a reference architecture and describes the functional components for Level 4 and Level 5 ADS features (autonomy software) as well as the relationship between them, which is agnostic to the specific implementation (Figure *1-6*). For the purpose of this dissertation, we assume a simplified high-level architecture as per Figure 1-7.

1.5.1 Motion Planning and Control

The main goal of the autonomy software is to provide the control commands to the AVP (Autonomous Vehicle Platform) that safely drive the AV from point A (current location) to point B (destination) again and again throughout the day. To do so, the

proposed architecture (Figure *1-6*) divides the software that performs the DDT vertically into multiple layers (motion planning and control layers). Each layer is executed at a different frequency and uses different data from the perception, prediction, localization, and mapping subsystems. The Mission Layer uses the static world model represented by a pre-recorded vector map to perform route planning, which represents a sequence of lane segments the AV should traverse to reach the destination. The Tactical Behavior Layer uses the perception and prediction data as well as the vector map data to produce a trajectory. On the other hand, the control command layer checks potential hazards and the drivability of the provided trajectory, and finally, the control actuation layer provides a closed loop (e.g., Model Predictive Control) steering, propulsion, and braking control.

1.5.2 Maps

Self-driving cars require pre-knowledge of the world where they will operate to aid localization, perception, prediction, and motion planning. This prior knowledge is recorded in high-definition maps (3D maps) that contain point cloud representations of the static world used for localization, as well as the vector maps used throughout the autonomy stack. The vector-map contain information about the road network, including pedestrian crossings, turners, traffic lights and signs, among others. For example, using extrinsic sensor calibration, and vector map data, it will be easy to assign a perceived road user to a lane segment and predict its next move using the successor relationship in the lane segments. Figure 1-8: Vector and Point-cloud maps[6] shows examples of the two types of maps.

1.5.3 Localization

Autonomous vehicles need to know their position and orientation very accurately (within 0.10 m and 0.17 deg 95% of the time [2]) so that they can sense, predict, plan their movements, and run safely overall. One way to accomplish this is by online scanning and extracting features from the world, then comparing them to features from a stored map (high-definition map and vector map data). In the robotics literature [12], this is referred to as the observation model. Bayesian filtering techniques like the Extended or Unscented Kalman filter can be used to provide smooth pose estimations at high frequency. The vehicle motion model can be emulated using data from the inertial measurement unit (IMU) and wheel speed sensors.

1.5.4 Perception

The perception system is responsible for processing camera, radar, LiDAR, and microphone data in order to detect, classify, and track all the elements in the environment that are moving or have the ability to move and alter their state.

The world is extremely complicated, and no perception system will ever be able to classify everything it senses. In such instances, the perception system still detects and tracks such objects by providing their position, direction, or motion, and speed without providing the class of the objects.

The perception system detects, classifies, and tracks many types of objects, such as construction objects, bicycles, pedestrians, motorcycles, regular vehicles, large

vehicles, school buses, and emergency vehicles, among others. The class of the object is important so the prediction system can assign a specific motion model for each object. For example, a prediction model for a motorcycle would assume a maximum speed that is much higher than what it would assign to a bicycle. In addition to objects and road objects, the perception system is in charge of finding traffic light states, unmapped traffic signs, and the left, right, and hazard light states of the leading object.

The perception system systematically fuses data from different sensor modalities using independent sensing-algorithm pipelines; depending on the driving conditions, it might favor one pipeline over another. For example, in low lighting conditions, the LiDAR pipeline is more effective than the camera pipeline.

1.5.5 Prediction

The prediction system's role is to anticipate what other road users may do and how they will interact with the AV in the future. For example, at a protected (controlled by a traffic or stop sign) left turn, based on the incoming object's speed, the prediction system would forecast that the object would not stop even if the traffic light were in a solid yellow or red state. To achieve autonomy Levels 4 and 5, the prediction system should be very responsive to aggressive non-compliant objects, including speeding, running red lights, and making illegal U-turns. This is usually achieved by training the prediction subsystem on a wide range of scenarios captured from collecting large datasets from the ODD as well as manually creating edge test cases and rarely encountered situations through simulation scenarios.

1.6 Summary

In this chapter, we provided a holistic overview of the automated driving system by introducing the SEA autonomy levels for road vehicles and the importance of the operation design domain (ODD). We also provided a general overview of the hardware components of the self-driving system, including a comparison between sensor types that are used to enable the perception systems. Finally, we presented a simplified functional architecture that is used to achieve level-4 autonomy.

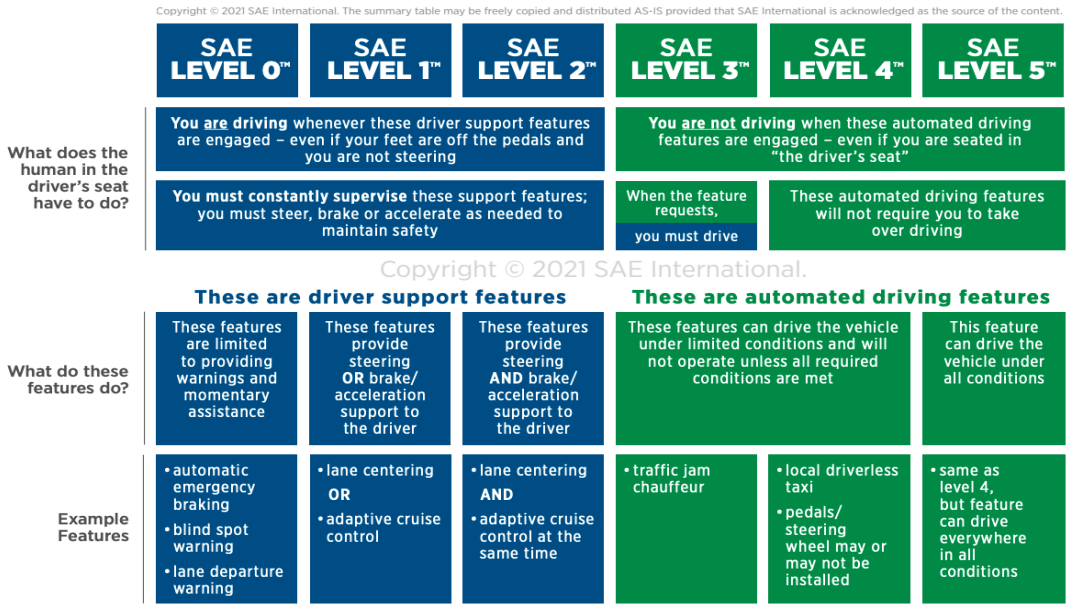


Figure 1-1: SAE levels of road vehicle autonomy [5]

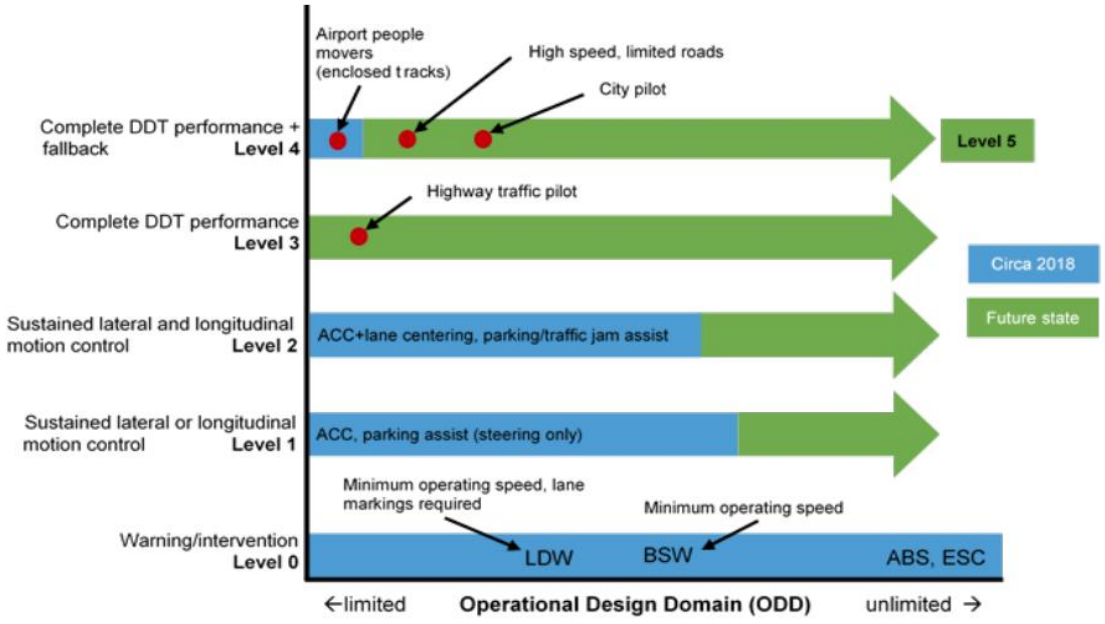


Figure 1-2: ODD relative to driving automation levels.

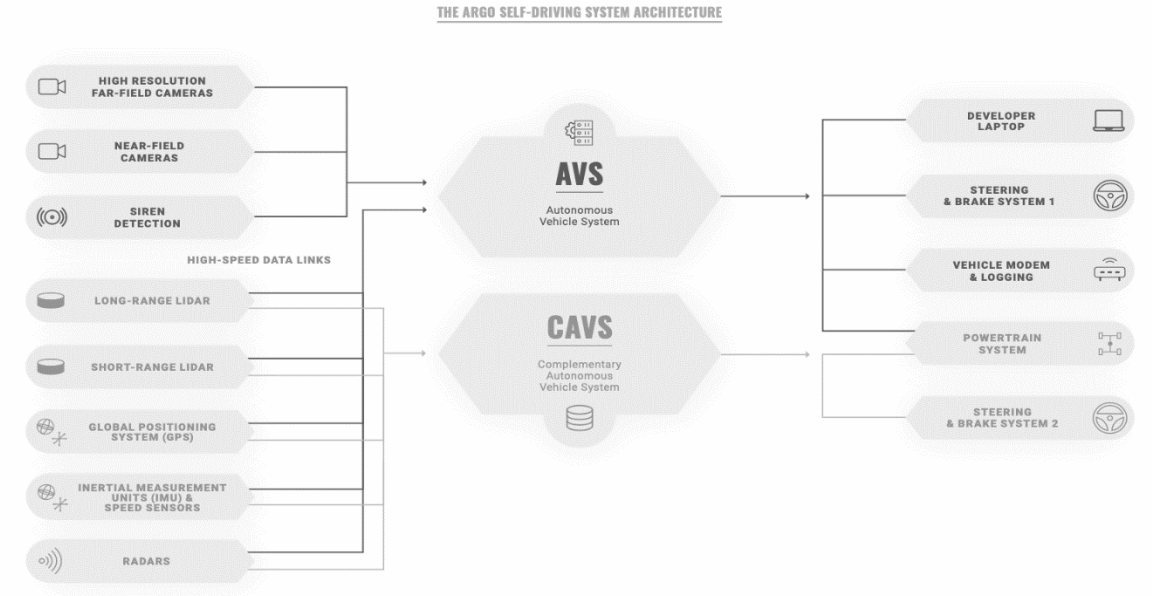


Figure 1-3: Argo Self-Driving System Architecture

Figure 1-4: Integration of the Argo AI SDS and Ford test vehicles

	Ultrasonic	RADAR	3D LiDAR		Cameras		
			Rotating	Solid State	VIS	IR	ToF
FOV	1	2	3	2	3	3	2
Range	1	3	3	3	2	3	2
Accuracy	1	2	3	3	3	2	2
Frame rate	2	2	2	2	2	3	3
Resolution	1	1	2	2	3	1	1
Colour perception	0	0	1	2	3	1	1
Size	1	1	2	1	1	1	1
Weather affections	1	1	2	2	3	1	3
Maintenance	2	1	2	1	2	2	2
Visibility	2	1	3	2	2	2	2
Price	1	2	3	1	1	3	2

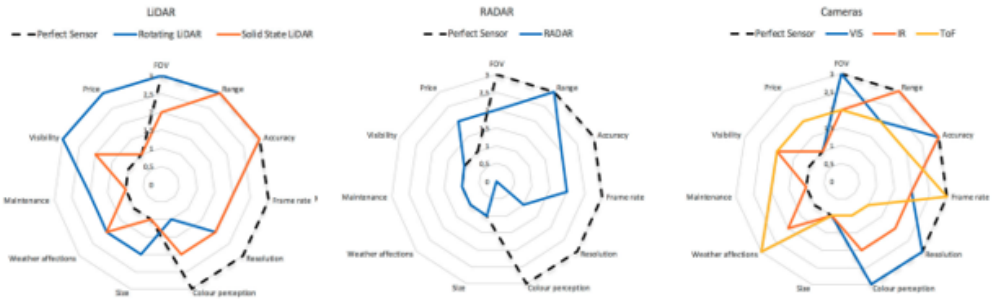


Figure 1-5: Comparison of the features of perception systems sensors [8]

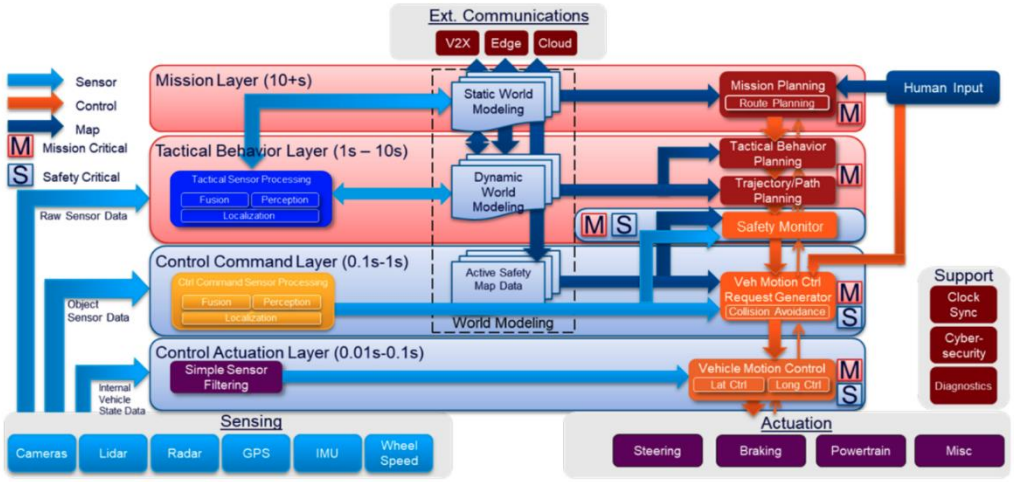


Figure 1-6: Overview of the reference ADS Functional Architecture [10]

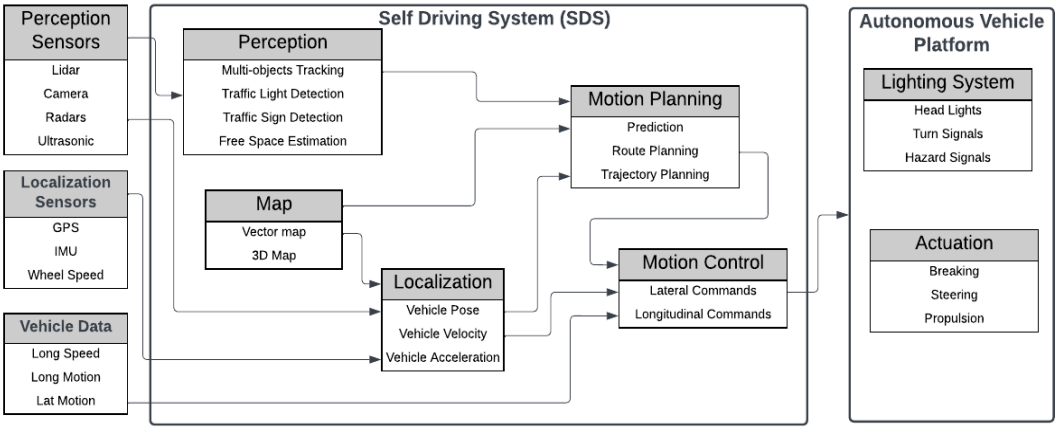


Figure 1-7: Simplified functional architecture.

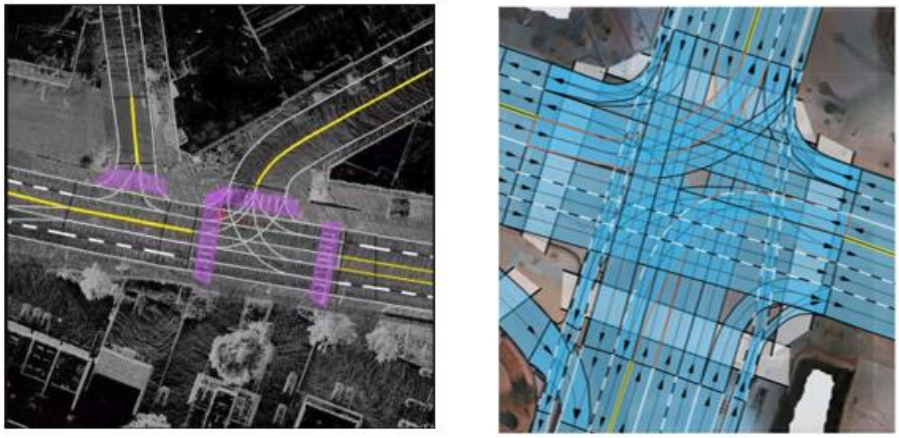


Figure 1-8: Vector and Point-cloud maps

CHAPTER 2: RELATED WORK

2.1 Introduction

The primary focus of this work is the verification of the perception system's (multi-object detection and tracking) performance. Specifically, out of all the objects and road objects in the scene, extract those from which the perception system for an AV must execute a near-perfect perception performance to avoid damaging and/or potentially life-threatening collisions with the AV. Although this dissertation deals with the perception system as a black box, i.e., it only looks at its input and output, it's essential to understand how the perception system works. Therefore, it's appropriate to divide this chapter into the following sections: In the first section, we review the latest advancements in perception systems; in the second section, we review the tracking system metrics, in the third section we introduce the methods used for relevant objects identification. In the fourth section, we look at how autonomous driving systems are validated in general and then take a closer look at the perception systems' verification. In the fourth section, we review some of the challenges related to autonomous system validation.

2.2 Multi-object Tracking

As one of the fundamental tasks of autonomous driving, Multi-Object Tracking (MOT) aims to sequentially process noisy sensor measurements to identify the number of dynamic objects and their state in the 3D world [16]. Besides the 3D location of the object in the world, the state of the object includes the width, length, and height, the class of the object (vehicle, pedestrian, construction cone, etc.), the speed and direction of

motion, among others.

As discussed in the previous chapter, sensors are subject to environmental noise, and no one sensor modality is uniquely capable of extracting the state information mentioned above, which is required for the AV to drive safely. For example, LiDAR is usually very good at estimating depth information, but class information is usually estimated from camera images. Besides this sensor's uncertainty and error (usually referred to as input error), multi-object tracking is subject to multiple challenges [17]: the number of objects is unknown, objects outside the sensors' field of view (FOV) can't be tracked, objects disappear because of occlusion or they exit the FOV, and objects can also enter the FOV (disappearance and appearance are usually referred to as object death and birth). As we will see in the next section, MOT relies on detectors that are not perfect and are susceptible to different kinds of errors, like missed detection, missed classification, and false negatives.

To address the above challenges, different approaches have been proposed, through this work, we examined two designs: one that divides the problem into two stages, namely detection and then tracking, by performing data fusion using Bayesian filtering techniques, and the second approach that provides an end-to-end solution (through a deep learning network) using a detector head that will be retrained as part of the end-to-end training stage [13]. In the sections that follow, we'll take a high-level look at both of these methods and the detectors that are used in both of them.

2.2.1 3D Detectors

In the past few years, there has been major advancement in 3D object detection for autonomous driving, thanks to many open-source AD datasets. For example, KITTI [19], NuSenses [20], and Waymo Open [21] are the most commonly used datasets in the development of 3D detectors for autonomous driving. The goal of 3D object detection is to classify and localize the objects of interest. The output of this process is the type of each object (e.g., regular vehicle, large vehicle, pedestrian, etc.), a 3D bounding box parametrized in its centroid (x, y, and z) and orientation yaw relative to the AV that captured the data, and the object's height, width, and length (h, l, and w). Figure 2-1 from [18] shows an example of the results of image-based 3D detection in both bird's-eye view and a two-dimensional image plane.

3D object detection approaches can be categorized into two groups based on the input data: image based and LiDAR-based [18].

Image-based 3D detection has made significant progress, enabled by the fast advancement of deep learning hardware and software. The first method is based on extracting the 2D location and other features, such as depth (that can be generated from a stereo pair or off-the-shelf depth or disparity estimator), of the object in the image coordinate system, then lifting the 2D detection to the 3D spaces using camera intrinsic and extrinsic parameters:

$$\begin{cases} z = d, \\ x = (u - C_x) \times z/f, \\ y = (v - C_y) \times z/f, \end{cases} \quad [1]$$

(Cx, Cy) is the principal point, f is the focal length and (u,v) is the object location in the image coordinates system.

One example framework of image-based 3D detector is proposed in [23], it is an end-to-end architecture of multiple deep learning networks (2D proposal, 3D regression and classification, Disparity prediction) that performs multi-level fusion on monocular images Figure 2-2 and Figure *2-3*.

LiDAR-based 3D object detection methods for autonomous driving have also made a lot of advances in the past decade due to the appealing features that LiDAR offers, like insensitivity to lighting conditions and the ability to accurately capture the 3D spatial information of objects and road users. Also, similar to image-based 3D object detection, the emergence of deep learning has boosted the performance of 3D point cloud analysis and made it possible to use it in real-time applications [22]. The LiDAR-based 3D detection pipeline (Figure *2-4*) is usually composed of three distinct stages: Sensor Data Representation (SDR) (Figure 2-5), Feature Extractions, and Core Object detection. In the SDR stage, the point cloud is converted into a structured representation (e.g., point-based, voxel-based, pillar-based, graph-based, projection-based, etc.). In the feature extraction stage, one or more deep neural networks (CNN, PointNet++, etc.) are used to retrieve rich and high-dimensional features. Then, these features are sent to detector networks, which output the class, location, orientation, and 3D bounding box of the objects in the FOV after a refinement stage to make the output more accurate. Object Tracking

While the 3-D detection subsystem's objective is to localize and classify objects on a frame-to-frame basis, the tracking subsystem makes the connection between these frames by assigning a track-ID to each object from its birth (the moment that it enters the field of view) until its death (the moment that it exits the field of view).

Detectors are not perfect; they are subject to noise and could output false positive detections or misclassify an object. Therefore, filtering detection outputs over time and making sure to provide accurate information to consumer layers (prediction and motion planning and control) or otherwise indicating that perception output for one or more objects is degraded by, for example, outputting "unknown" for the type or providing high covariance for the state vector of the object, so the upper layer can take proper actions like slowing down, Failing to do so could be catastrophic, as we saw on March 18, 2018, with Uber's incident that fatally injured a 49-year-old pedestrian crossing N. Mill Avenue outside the crosswalk. Based on the accident report concluded by the National Transportation Safety Board (NTSB) [24], 5.6 seconds before the collision, the ADS detected the pedestrian. Although the virtual driver continued to track the pedestrian until the impact, it never accurately provided the correct classification. The software labeled the pedestrian as "unknown," then as a "vehicle," and then as a "bicycle," making different guesses about where they would go next.

Bayesian Object tracking provides a probabilistic framework to solve the multi-object tracking problem. The goal of this framework is to perform multi-data fusion on the 3D detections (regardless of the sensor modality that they are generated from) in order to compute the state vector (class, location, orientation, size, speed, acceleration,

…etc.) and a track label for each object and road user in the field of view. It does that by recursively performing two steps (state prediction and state update) every time a measurement or detection is received. Figure 2-6 provides a high-level overview of this system:

- The prediction step uses the class information provided by the detector, the posterior from the latest state update, and an assumed motion model per object type to compute a prior probability distribution. For example, for an object of type vehicle, a bicycle model [25] is used to compute this distribution.

- The update step is a bit more involved; internally, it tries to probabilistically model the dynamic environment as well as measurement uncertainty. For example, when a measurement is received, a data association step is performed to assign a detection to an existing object based on some metric (holonomic distance); this is called gating. If a measurement could not be assigned to an object, this step also manages to completely ignore (clutter model) or start a new track (birth model). For those existing tracks that did not get any measurement assigned to them in the current step, the probability assigned to them will be updated using miss-detection in case of occlusion or the death model in case they are completely out of the FOV. Finally, in the data association step, multiple associations are possible; in some methods, only the association with the highest probability is considered (JPDA)[33]; in other methods, multi-hypothesis is maintained and updated in both the prediction and update steps, but to keep this computationally attractive, a pruning (only keep the hypothesis with probability

higher than a threshold) and capping (only keep the N hypothesis with the highest probability) are used.

2.2.2 Deep Learning Based Sensor Fusion for Multi-Object Tracking

Deep learning-based tracking by detection has achieved competitive results on large-scale datasets like nuScenes [20] and Waymo [21]. For example, the TransFusion architecture presented in [13] is adaptively learned to fuse camera and lidar data, that is, it's able to determine when and what information should be taken from the camera images verses LiDAR point cloud, for example in the case of image clutter, as well as when the lidar is not able to detect objects because of sparsity (far away objects) or low reflectivity (dark objects). This architecture is illustrated in Figure 2-7.

To link objects through time, this architecture used CenterTrack [26]. Where the center of each object is tracked through time, an offset vector from the current object center to its center in the previous frame is learned through training and output at inference, which is used for object association through greedy matching. Figure 2-8 provides illustration of centerTrack where object tracked by tracking their center.

2.2.3 Evaluation Metrics of Multi-Object Tracking (MOT)

An evaluation metric for multi-object tracking is an indicator of how well a tracker performs. It is computed by evaluating the similarities between the tracker prediction and the ground truth. The most common similarity metric is the intersection over union (IoU), which is the spatial intersection between two regions (2D rectangles or 3D boxes) over the union of the regions. Both the ground-truth tracks and the tracker

predictions have the same format. Tracks are represented as a set of detections in a well-defined coordinate system. Each detection is given a unique identifier, and it's expected that an object's trajectory has the same identifier over time. The class ID is also assigned to each object, and evaluation metrics are usually evaluated per class, then averaged over the class. There are mainly three categories of possible errors between a set of ground-truth and predicted tracks: detection errors that happen when the detector predicts detections that do not exist in the ground-truth or when it fails to detect objects that exist in the ground-truth, Localization errors occur when the tracker predictions are not spatially aligned with the ground truth, and association errors occur when the tracker assigns different track-IDs to the same objects or the same track-ID to two distinct objects. When computing the MOT evaluation metric, it’s a common procedure to perform one-to-one or bijective matching, that is, each ground truth label is matched to at most one prediction and vice versa. When doing so, any miss-detection or false-detection can be penalized. The Hungarian algorithm [30] can be used to find the best match (the match that optimizes the sum of similarity scores). The matching pairs are called True Positives (TP), which are considered correct tracker outputs. Any tracker predictions that are not matched are considered False Positives (FP). Any ground-truth labels that are missed are considered False Negatives (FN).

The MOT evaluation metrics are a great indicator of the performance of the perception system. They can be used to track the progress of the perception system and flag any regression in the development phase. Also, some of the evaluation metrics are able to indicate if the poor performance is due to a detection or association problem.

Finally, we must mention that benchmark datasets use these metrics to evaluate and compare perception algorithms and push the research topic forward.

Despite the benefits mentioned above, these evaluation metrics do not provide a reliable indication of whether the perception system meets the safety requirements. Our approach does not replace these metrics but complements them by focusing the evaluation and verification of the perception system on the objects that matter in the scene.

In the following sections, we provide an overview of the most used evaluation metrics for multi-object tracking in the context of autonomous driving. We will adopt the terminology in [15]. That is, we will refer to a ground-truth detection by gtDet, to its id by gtID, and to gtDets that represent the same object (which has the same gtDet) by gtTraj. Similarly, we will refer to the multi-object tracker's predicted detection by prDet, to its ID by prID, and finally to prDets that have consistent prIDs over time for detection of the same object by prTraj.

2.2.3.1 CLEARMOT

CLEARMOT [27] consists of two sub metrics: MOTA (Multi-Object Tracking Accuracy), which measures the detection and association errors, and MOTP (Multi-Object Tracking Precision), which measures the localization error.

In MOTA, a one-to-one (bijective) mapping is constructed between the gtDet and the prDets in every frame. Matched ground-truth and prediction becomes part of the true positive set (TP). Any remaining prDet that is not matched becomes part of the false

positive (FP) set. Any gtDets that are not matched become part of the false negative (FN) set. For example, IOU is used as a similarity measure, assuming some threshold. To measure association, Identity Switch (IDSW) is used. An IDSW is a TP that switches its ID relative to the previous frame, although it corresponds to the same object. The following equation is used to compute MOTA.

$$MOTA = 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDet|} \quad [2]$$

MOTP is used to measure localization accuracy, which is the average similarity score over the true positive set as per the following equation.

$$MOTP = \frac{1}{|TP|} \sum_{TP} S \quad [3]$$

2.2.3.2 IDF1

The Identification Metric IDF1 [28] creates a bijective mapping between a set of gtTraj and prTraj. Unlike MOTA, which matches at the frame level, a new type of match set needs to be defined. An identity true positive set (IDTP) is a set of matches on the overlapping part (where similarity is larger than a predefined threshold) of trajectories that are matched together. Identity false negative set (IDFN) and Identity false positive set (IDFP) are the remaining ground gtDets and the prDets that don't have matches. The IDF1 score is calculated as per the following formula:

$$IDF1 = \frac{|IDTP|}{|IDTP| + 0.5|IDFN| + 0.5|IDFP|} \quad [4]$$

2.2.3.3 HOTA

The Higher Order Tracking Accuracy (HOTA) Metric [15] combines the effects of detection, association, and localization errors into a single metric, similar to MOTP. It computes the localization accuracy LocA using the following formula:

$$LocA = \frac{1}{|TP|}\sum_{TP} IoU \quad [5]$$

Also, similar to MOTA, it measures the overall detection accuracy (DetA) by using the Jaccard Index using a predefined IOU threshold.

$$DETA = \frac{|TP|}{|TP| + |FN| + |FP|} \quad [6]$$

For association, HOTA proposed the concepts of TPA (True Positive Association), FPA (False Positive Association), and FNA (False Negative Association). Then for a specific localization threshold α , $HOTA_\alpha$ is computed which weight each true positive by association score A when computing the Jaccard metric:

$$HOTA_a = \sqrt{\frac{\sum_{c\in\{TP\}} A_{(c)}}{|TP| + |FN| + |FP|}} \quad [7]$$

$$A_c = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \quad [8]$$

This equation account for detection and association accuracy but does not account for localization accuracy, for HOTA to measure localization $HOTA_\alpha$ in integrated over a valid range of α as per the following

$$HOTA = \frac{1}{19} \sum_{\alpha \in 0.05 \ to \ 0.95 \ step=0.05} HOTA_{\alpha} \quad [9]$$

2.3 Relevant Objects Identification

Existing research on identifying important objects in a scene can be classified into three categories. The first category focuses on predicting the driver's gaze by imitating human drivers [40]. However, these methods typically only provide attention at the pixel or region level and do not indicate the significance of each object or instance. Additionally, human drivers may not always pay attention to the important objects and may instead focus on objects that are not related to the driving task, which reduces the accuracy of the information. The second category involves training attention-based models for specific tasks, such as trajectory forecasting [37]-[38] and end-to-end driving [39], with no explicit attention supervision. The third category involves identifying important objects with explicit supervision through human annotations, allowing for the incorporation of human knowledge. These models are trained through standard supervised learning [34], [35]. For example, Gao et al. formulate the problem of identifying important objects as Object Importance Estimation (OIE) in on-road driving videos. The important objects are defined as road users, such as vehicles and pedestrians, that are relevant for the driver of the ego vehicle to make a driving decision. To extract the important objects, they used the dynamics of the objects and the driving goal to train a model based on LSTM [36] on top of an object detection network.

Similar to [35], our approach uses the goal of the ego vehicle motion (through the route planner message) and the road object dynamics, but it identifies objects by modeling their kinematics considering non-compliant behavior, which is typically rare to see while testing and before production. In other words, in approach which is mainly used in the verification process of perception systems maximizes the use of data extracted from a scenario to predict the actual performance at production when the number of produced autonomous vehicles is very big compared to the fleet size. For example, in the Figure 2-9 scenario, the approach in [35] would only mark the pedestrian crossing the street as important (marked with a red dashed circle), while our approach would also mark the two vehicles (identified with blue and purple rectangles spatially relevant (assuming they are currently at near zero speed), since in a different scenario (assuming nobody is crossing the street), any of these vehicles could ignore traffic rules and etiquette and make a left turn in front of the AV.

With respect to the verification and evaluation of perception systems in the application of safe automated driving, understanding when perception errors are safety-critical is a major challenge. This area is just now beginning to be explored, and several methods have been created to solve this issue, mainly by creating task-aware evaluation measures, such as safety-aware metrics, or defining what is meant by safety-critical objects and computing the existing metrics only on this subset. Although task-independent metrics offer high comparability across a range of benchmarks, such task-agnostic measures fall short of capturing the true performance of a perception system when it is deployed in the real world and incorporated into an autonomous stack. This is

due to the possibility of highly varied behavior in subsequent tasks depending on the type of failure [41]. It is difficult to rely on existing metrics (CLEARMOT, HOTA, etc.) to check and verify perception algorithms implemented in the actual world due to the misalignment between these metrics and the application In fact, there has recently been a trend for employing task-aware measures as the gold standard for evaluating perception algorithms [41], In particular, the nuScenes benchmark [43] has adopted the planning-aware metric proposed in [44]. The planned planning-aware metric compares the ego vehicle's plans with and without perfect detection using KL-divergence. Unsurprisingly, [44] demonstrated empirically that the significance of correctly detecting an object depended on the distance from the ego vehicle and the future trajectory of the ego vehicle, and they demonstrated that their findings were consistent with the perception of humans regarding which perceptual uncertainties are dangerous. In addition to these initiatives, various safety-aware metrics have been proposed, such as [45], which suggests (linearly) integrating scores indicating detection quality, collision potential, and detection time. Another example is [46], which classifies objects according to their likelihood of colliding (imminent, potentially, none) as determined by a streamlined computation of the forward reachable set under the assumption of isotropic force.

Another approach to solving this challenge is to identify the environment's safety-critical objects and road users and ensure that perception performance is high for such a subset. In this vein, other techniques have been proposed, many of which can be seen as particular versions of the reachability problem [47]. For example, [45] uses the responsibility-sensitive safety (RSS) framework [48], which classifies another vehicle as

safety-critical if a collision with it would happen if both used their brakes and stopped. Similarly, [46] deems a collision between an ego vehicle and another vehicle to be safety-critical if both vehicles continue to move at a steady speed (i.e., zero control input). While considering the dynamics of the agents, these safety assessments nevertheless make a number of strong behavioral assumptions about the agents, including assumptions about the actions of other vehicles (such as performing a hard brake), over which the ego-vehicle regrettably has no control. Our approach builds on top of these methods by modeling the AV trajectory along the planned route, the objects' longitudinal behavior (acceleration, deceleration, and constant speed motion), as well as their 2-D possible paths (U-Turn, lane-change, left and right turn) constrained by their current state, class type, and the drivable area boundaries, in our approach although we have some assumption about the object's acceleration and deceleration parameters we relax these by testing reachability at one second interval instead of instant reachability.

2.4 Perception System Verification Process

Before being able to put a level-4 autonomous driving system on the road, all the safety claims and goals should be proven and achieved. This requires many years of continuous development and testing that takes the self-driving system (SDS) and the autonomous vehicle platform (AVP) through multiple generations of hardware. In each generation, tens to hundreds of vehicles are built and sent to test tracks and public roads (within the ODD) to drive autonomously, but under the monitoring and supervision of one or more test specialists that are prepared to intervene by taking control of the vehicle (Autonomy Disengagement) in the event that they determine that the AV is about to

encounter a situation beyond its current development capabilities [6]. After the fact, the logs of that takeover are analyzed, and an assessment is made of what the AV would do without intervention.

Besides the obvious reason that public roads are where ADS ultimately will be deployed, testing on public roads has many benefits. The most relevant reason for the perception system is collecting sensor data from real-world scenarios to create an incremental dataset (logs and ground truth labels) that will be used to validate subsequent software releases in a virtual environment using a tool (called Resimulation in [6]) that takes a recorded log and runs the new perception stack over the sensor data (camera, lidar, and radar). The new output (tracks) of the perception system is then compared with the ground truth, and failures are prioritized to be fixed. Figure *2-10* illustrates a high-level development process of the perception system for autonomous vehicles. It's worth mentioning that the labeling process it's costly.

In such a highly dynamic environment and complex system, reaching zero failure rate is near impossible, but the failure rate of the perception system must be below a certain probability threshold, which is usually derived from a system level failure rate (collision per mile unit). For example, [2] derives this number by targeting the system-level failure rate of commercial aviation, then allocating a failure rate for each subsystem in such a way that the overall failure rate is still achieved Figure 2-11. In the architecture presented in this reference, the acceptable perception system failure rate was anything less than one failure per 10^{-9} mile.

2.5 Summary

In this chapter, we examined the perception system used for autonomous driving, with a focus on the problem of multi-object tracking (MOT) and its associated challenges. We provided an overview of the state-of-the-art in MOT and the main components involved, as well as introduced the algorithms used to measure performance. We also discussed the verification process for the perception system.

The following chapter discusses the motivation for this work, presents the problem addressed, and explains how our approach fits into the perception system and verification process to achieve desired safety goals.

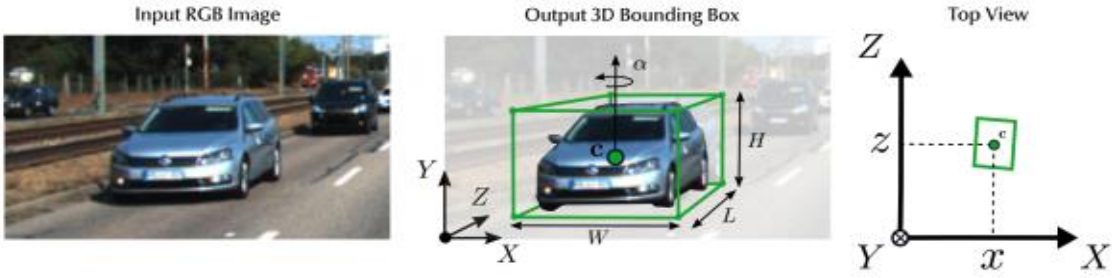


Figure 2-1:Example result of image-based 3D detection from mono camera.

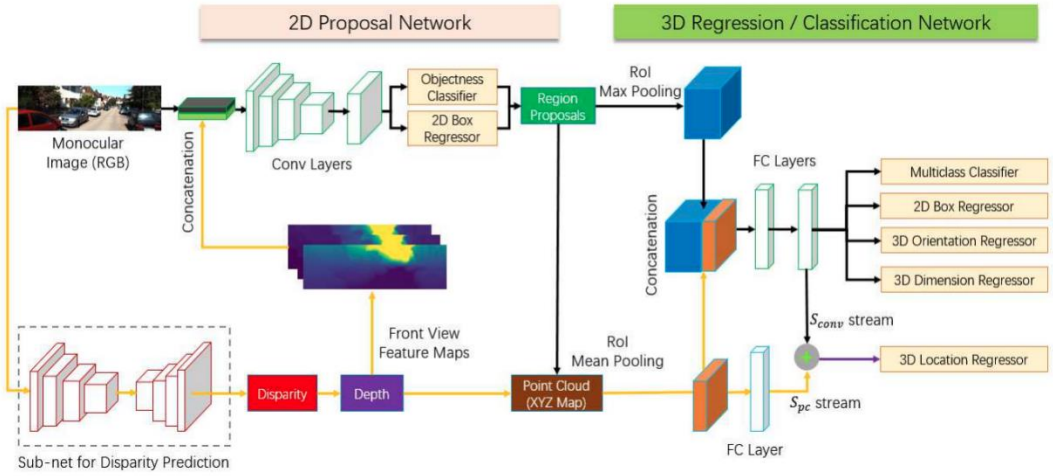


Figure 2-2: Multi-Level Fusion based 3D Object Detection [23]

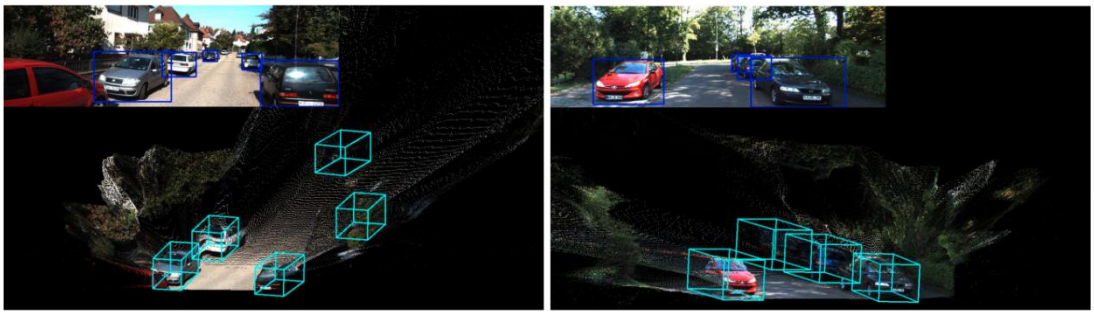


Figure 2-3: Visualization for 2D and projected 3D detection boxes from [23]

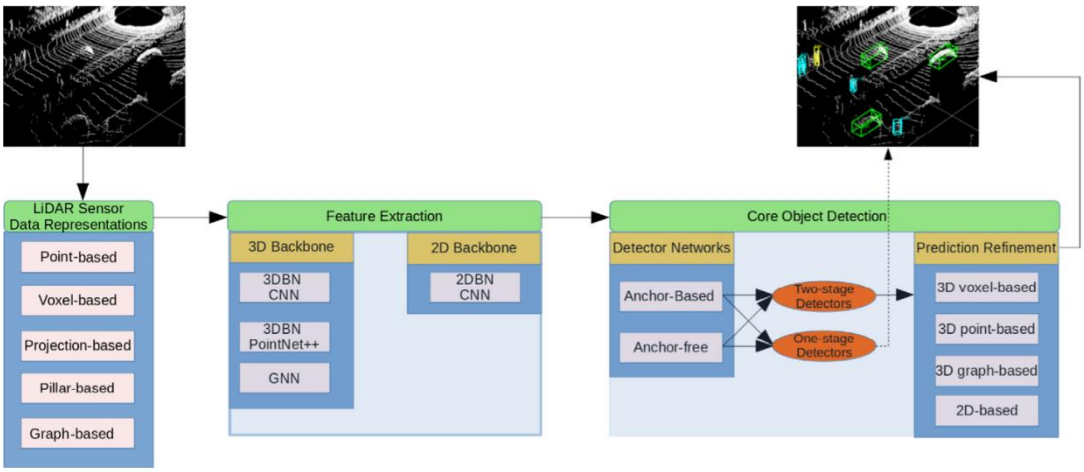


Figure 2-4: Pipeline of LiDAR-based 3D object detectors with DL

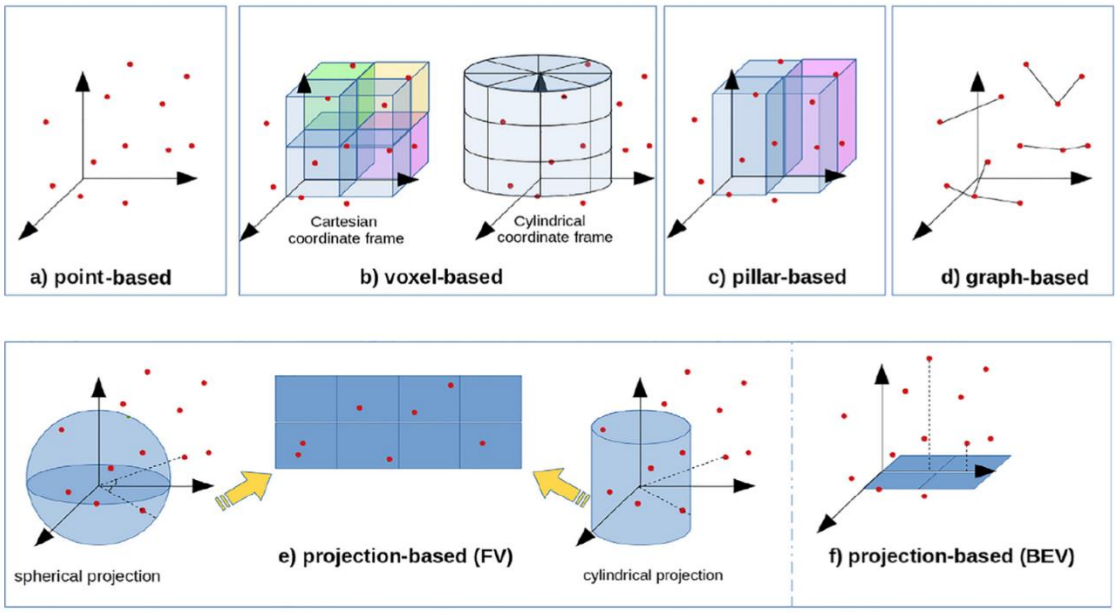


Figure 2-5: Point Cloud representation [22]

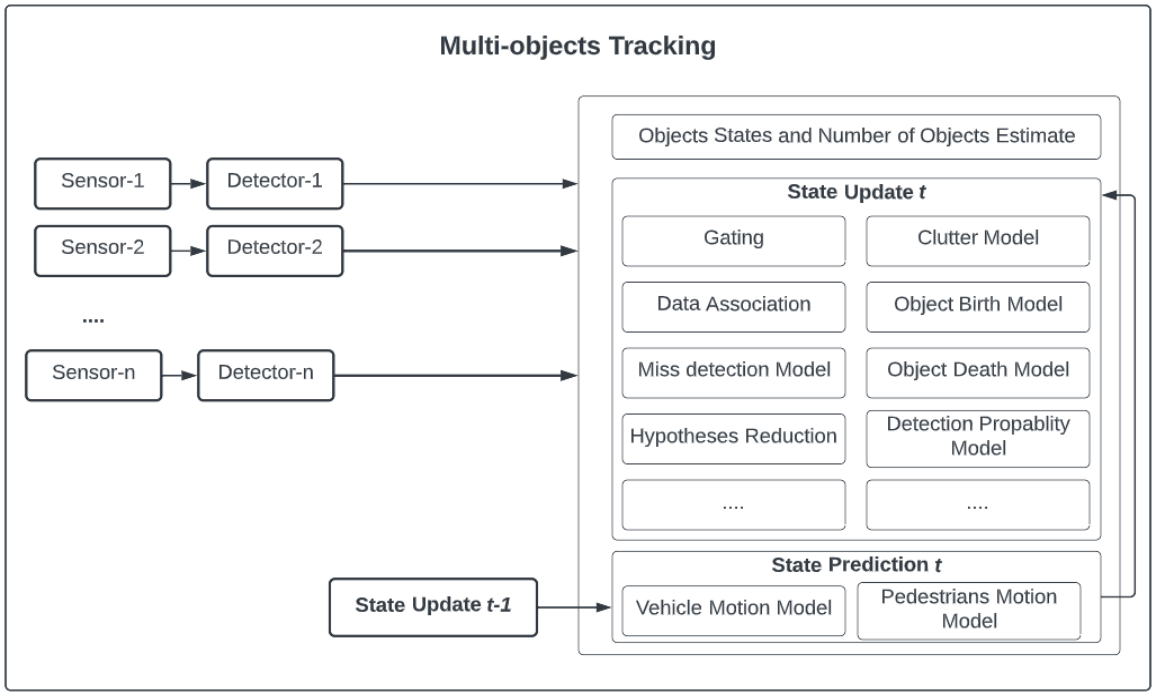


Figure 2-6: Multi-Object tracking system.

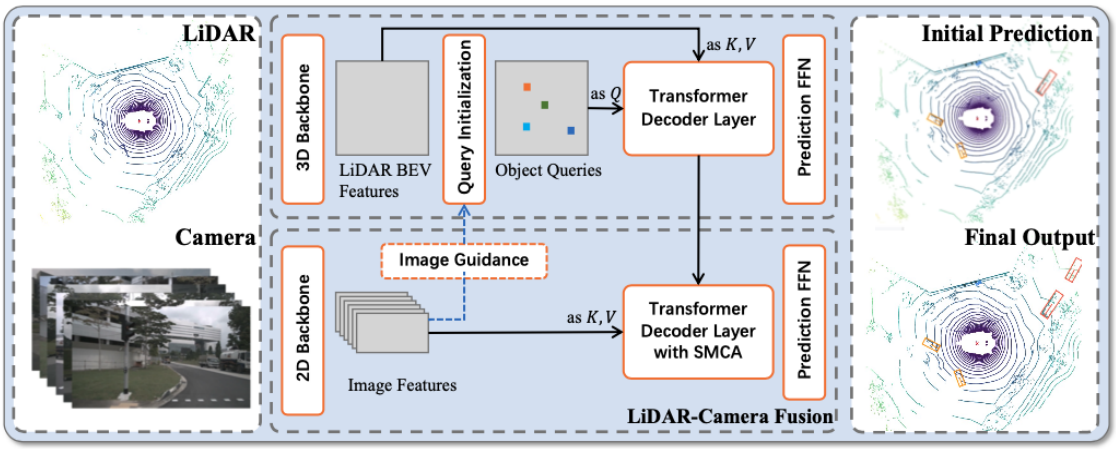


Figure 2-7: Transfusion Architecture [13]

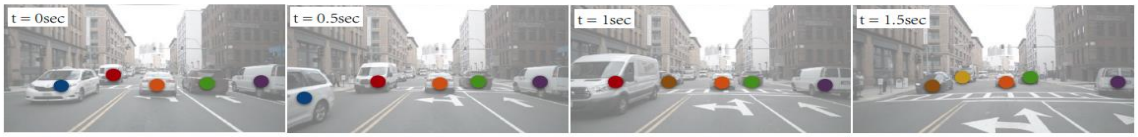


Figure 2-8: illustration of centerTrack [26]

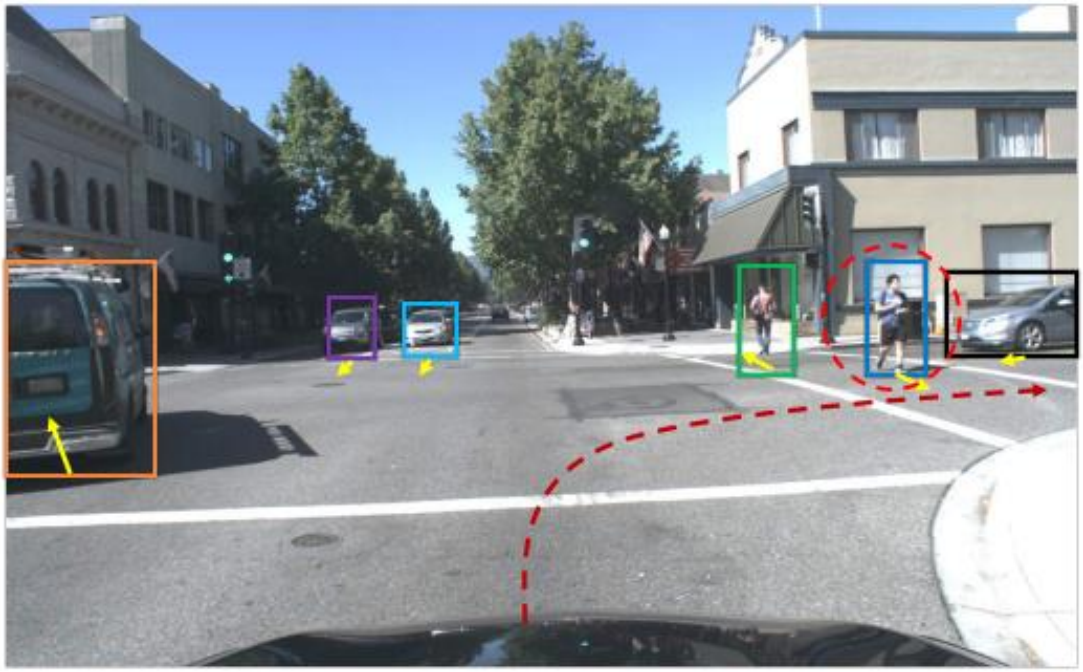


Figure 2-9: Goal-oriented Object Importance Estimation [35]

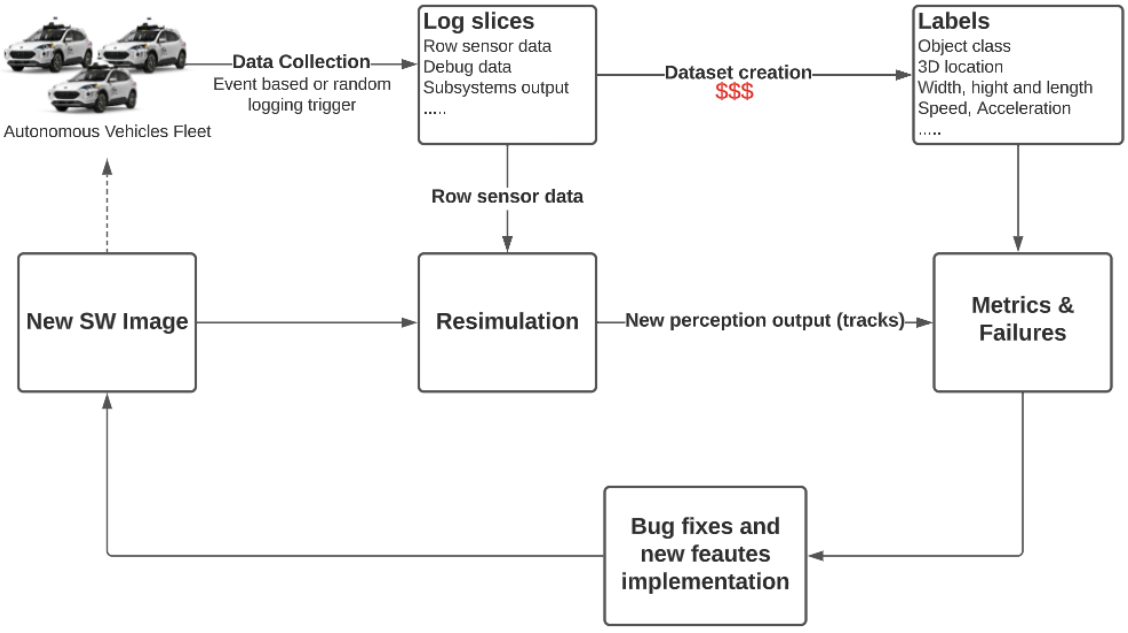


Figure 2-10: Perception system verification process

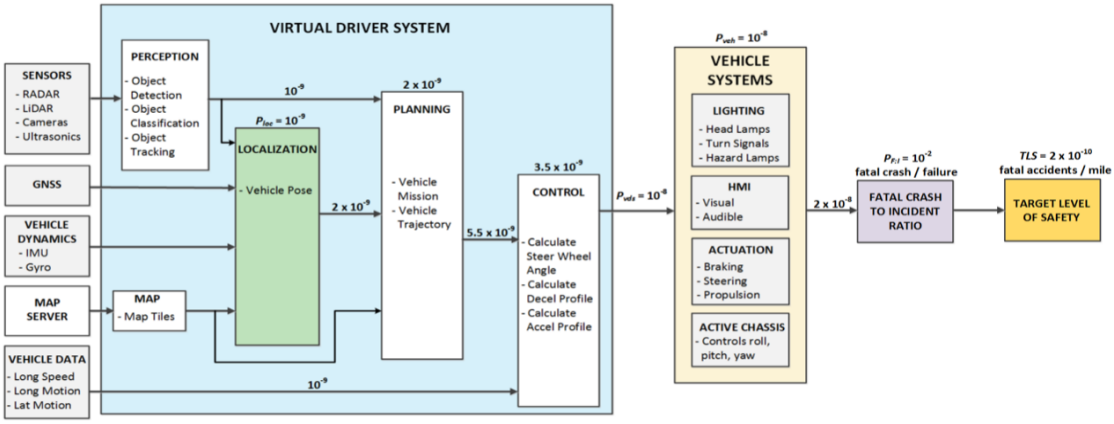


Figure 2-11: Virtual Driver System integrity risk allocation [2]

CHAPTER 3: PROBLEM STATEMENT

3.1 Motivation

In the last decade, automated driving systems in general and perception systems for automated driving in particular have made significant progress. However, there are many challenges related to both the development of the runtime system as well as the verification of the system. One challenge related to this work is assessing if the autonomous driving system is ready to operate on its intended ODD. Karla et al. [14] used statistics to figure out how many miles need to be driven before a self-driving system can be said to be safe for public use. They found that hundreds of millions of miles would need to be driven to prove its reliability and concluded that new methods are needed to show that the system is safe and reliable.

As explained in the previous chapter, one of the new methods to validate the system is to set a system-level failure rate and then allocate to each layer (perception, localization, prediction, motion planning, etc.) a target failure rate, in such a way that the system-level failure rate is still achieved when applying joint and conditional probability laws. This allocation of failure rate makes it possible to evaluate the performance of the perception system (sense, detect, and track) independently of the rest of the system. This is usually done by creating ground truth labels from the sensor data and then using these labels to calculate metrics based on the output of the tracking system. Looking at the leaderboard of one perception benchmark dataset [21], it is hard to draw a conclusion about the readiness of the perception system for the application of autonomous drivers, i.e., there is no connection between the failure rate of the perception system when

calculated using these metrics and the overall safety of the autonomous driving system. On the other hand, taking all the objects and road objects in the scene into consideration when evaluating the performance of the perception system would penalize the system more than it should.

In this dissertation, we automatically filter in a set of objects and road objects that could interact with the AV; we refer to these objects as "spatially relevant objects," and we expect the perception system to have near-perfect performance when computing the evaluation metrics. This is important because it brings everything into focus and pushes the development of the perception system in the right direction. Furthermore, for each of the spatially relevant objects, we automatically get at least one motion trajectory (regardless of compliance with road rules) that could cause a hypothetical collision with the AV, as well as the severity of the collision, which is used to rank the importance of these road objects. In the simulation environment, the spatially relevant object can be forced to follow these motion trajectories (individually or in combination) and check if the AV would behave correctly and be able to successfully avoid a collision with the simulated object. This is a novel approach to automatically augmenting the scenarios dataset, creating meaningful scenarios that the AV could encounter on public roads, and consequently bringing the autonomous system failure rate down.

Figure *3-1* illustrates how the system can be used offline for the purposes of both perception system verification and sim generation. Figure 3-2, illustrates how spatial relevancy modifies the evaluation process of Multi-Object tracking system.

It is important to note that the reachability principle is the core factor used in determining the spatial relevance of an object. As a result, mapping the system-level probability of failure to the perception-level probability of failure becomes more straightforward and accurate. In its simplest form, if a multi-object tracker fails to track a spatially relevant object, this can be automatically considered as a system-level failure.

3.2 Problem Formulation

This dissertation proposes a system to be used in the development and verification of the perception subsystem of autonomous vehicle. This system can be used off-line as part of the verification process of the perception system, and a way to automatically generate edge case simulation scenarios that are taken from real-world situations with the intent of improving the performance of the ADS. The main goal of this system is to extract spatially relevant objects from ground truth. A spatially relevant object (SRO) is defined as a road object (vehicle, motorcycle, bicycle, etc.) that could reach the AV's intended trajectory using a kinematically feasible trajectory. This trajectory does not have to be compliant with the rules of the road, and one or more road object trajectories must intersect, in time and space, with one or more AV trajectories. Within a given time interval, if a road object is intermittently spatially relevant, the road object is considered relevant throughout the entire duration between the first time-step and the last time-step that it is spatially relevant.

In addition to classifying objects as either spatially relevant or not, the proposed method gives a way to rank the spatially relevant objects based on how bad the hypothetical collision with the autonomous vehicle would be.

In order to achieve this, at each time step, the system uses the AV information (planned route, vehicle pose, motion capabilities, and constraints) to simulate motion trajectories along the planned route. Also, it uses the road objects' attributes, vector map data, and predefined assumptions and constraints to simulate kinematically reachable motion trajectories for each road object, then tests for intersection in time and space between the AV-generated motion trajectories and each of the road objects' generated motion trajectories, after discretizing at a one-second interval. If an intersection occurred, then the object is classified as spatially relevant and ranked based on the simulated collision severity that is computed using relative speed at the intersection point and the object class.

The environment that this system would operate in (real-time or from recorded logs) is very dynamic. For example, the AV and road objects can execute different random maneuvers at different speeds, and they can be anywhere within the ODD. This significantly complicates the problem:

- The first complication is that the search space of kinematically possible trajectories that the AV and the road objects can execute is infinite, how do we choose the subset of these trajectories that make the implementation of this

system feasible, and what safety rules to consider when choosing these trajectories.

- The second complication is once the subset of trajectories is identified, how to model these trajectories in a way that would work anywhere in the ODD regardless of the geometry of the road.
- The third complication is how to rank the spatially relevant objects to prioritize failure fixes.
- And finally, the fourth complication is how to handle interaction between the AV and the road objects as well as the road objects themselves.

3.3 Approach

- To overcome the complications in the above section, we first use the following intuitive yet powerful rules [31] that the autonomous driving system must adhere to while performing a trip:
    - **Rule-1:** Do not collide with any object.
    - **Rule-2**: Do not cut-in recklessly.
    - **Rule-3:** If you can avoid an accident without causing another one, you must do it.
    - **Rule-4:** Right-of-way is given, not taken.
- To find spatially relevant road object in the scene, we ask: Could the AV break any of these rules if the perception system failed to detect, classify, or localize this object?

- For Rule-1 the AV should have enough time to stop and avoid a collision with a front road object that is already stopped or stopping in the middle of the lane. Also, if the AV is stationary, it should not start accelerating before the front object starts moving. Evaluating if any of the AV's deceleration or acceleration trajectories intersect in time and space with the object's deceleration or stop in place trajectory (assuming some constraints about the AV and each of the road objects) would indicate that this object is important to track and thereby the rest of the system can predict what it will do next and evaluate if it needs to modify the AV's current planned trajectory.
- For Rule-2, the AV should not change its current steady state, causing other objects to perform aggressive maneuvers to avoid collision. This includes cases where the AV is performing right or left turns, merging into traffic, or lane changes. For example, when the AV is performing a right turn or merging in traffic, it should not get rear ended by a road object that is driving in the target lane but has the right of way. It should also not cause that object to make an aggressive deceleration to avoid the collision. Evaluating the intersection in time and space of the AV's accelerating or lane change trajectories with each object's steady speed or comfort deceleration trajectories would indicate if the object is spatially relevant or not.

- Rule-3: can be achieved when the perception system is able to identify objects that might collide with the AV if it happens to undergo a sudden change in its velocity or direction. These are the "behind" or "left or right adjacent objects" relative to the AV. Similar to Rule-1 and Rule-2, evaluating the time and space intersection of the object comfort deceleration and the AV's safety evasive deceleration would indicate if that object is spatially relevant or not.
- Rule-4: can be achieved if the AV takes into consideration that a road object might not comply with the road rules or driving etiquette (for example, if a driver is distracted and swerves into the oncoming lane, they might fail to stop for a stop sign or red light or jaywalk in the case of pedestrians). Therefore, when computing the acceleration or deceleration trajectories of a vehicle type object, for example, only the drivable area boundaries, the object's inferred properties (speed, direction of motion, dimension, etc.), and some assumptions about its capabilities will be considered, but nothing about the direction of the lane it's located on or if there is a traffic sign or light that would normally control its motion.

- The route planner subsystem provides the AV's planned route, represented as a set of adjacent lane segments we call a "map corridor." We use that information to compute the centerline of this route, then compute the AV's acceleration and deceleration trajectories along that centerline from the current AV location. The

AV’s trajectories shall be clipped by a static object on its route if it cannot pass this object without going out of the route corridor.

- AV’s trajectories along the map corridor are discretized at a one-second interval. The result of this process is multiple polygons, which are called iso-temporal regions. To find out these isotemporal regions, we use the map corridor centerline heading to draw a perpendicular line at a one-second interval. Intersecting points between these lines and the left and right boundary lines of the map corridor will create the iso-temporal region for the acceleration or deceleration trajectories.
- For vehicle type objects (passenger cars, large vehicles, motorcycles, and cyclists), it’s hard to know what they will be doing next. Therefore, we compute many trajectories for both acceleration and deceleration behaviors to cover the whole kinematically reachable space based on the object speed, minimum turning radius, lateral acceleration limit, and bicycle model. The trajectories are then cut off at the point where they meet the edges of the drivable area or fixed objects.
- For stationary vehicles, a reverse trajectory is also added to capture the case where an object is pulling out of a parking space.
- For pedestrians and animals, a random walk trajectory at constant speed is evaluated without considering the boundary of the drivable area.
- At each time frame, for each object that can be seen by the AV, the intersection in space and time between the AV's isotemporal regions and the computed trajectory for each object is checked to see if it is spatially relevant or not.

- When evaluating the relevancy of an object, visibility and occlusion are considered, and objects with low visibility are considered not relevant.
- Based on the relative speed at the intersection point and the type of the object, the severity classification of a spatially relevant object is computed, this classification can be used for ranking the importance of the spatially relevant objects.
- We apply a sliding window of a certain width within a time interval if the road object is intermittently spatially relevant, the object is considered relevant throughout the entire duration between the first-time step and the last time step that it was spatially relevant.

3.4 Spatially Relevant Objects Example

To illustrate the concept of spatially relevant objects, we use the following hypothetical scenario, illustrated in Figure *3-3*, in this scenario, the AV is stopped at an intersection and trying to do right turn, as illustrated by the green arrow. At the current timestep, cars (c1, c2..., c7), pedestrians (p1, p2), motorcycles (m1, m2), and large vehicles (lv1) are in the field of view of the AV or have corresponding ground truth. The black arrows attached to some of the objects indicate that this object is in motion and pointing in the direction of motion; also, the number near to each moving object represents its speed in miles per hour. The speed of the lane where the vehicle wants to go is 45 mph, we will assume that it would take the AV 5 seconds to reach the road speed limit and claim the lane.

The following is the expected output of the spatial relevancy system with the corresponding reasoning (Figure 3-4):

- Spatially Relevant Objects:
    - **Bicyclist b1**: at speed of 5 mph, it would take this cyclist about 2 to 3 seconds to cross to the opposite side of the road, it could also speed up or slow down, the AV acceleration trajectory will collide with the acceleration or deceleration trajectory of the bicyclist.
    - **Pedestrian p1**: although this object is currently not moving, it could start jaywalking to cross the street, so it's better to keep track of it.
    - **Vehicle c6:** This object is either parked in a driveway or getting out of a parking lot. It is feasible for this object to make a right turn while the AV is accelerating, if not outputted, there is a chance of collision.
    - **Motorcyclist m1:** is stationary at the intersection. Most probably this object will proceed forward or do a left turn, but it’s feasible for this object to do a non-compliant U turn and collide with the AV.
    - **Large Vehicle lv1**: This object is also stationary; its trajectory could intersect with the AV trajectory by acceleration and changing lanes, regardless of the right of way or traffic light status.
    - **Vehicle c7:** this is a high-speed vehicle driving on AV target lane, it’s about 60 meters away from the AV, and assuming the object will not slow down, and the AV decides to execute the right turn trajectory, the AV will be rear-ended by this object between seconds 4 and 6.
- Not Spatially Relevant Objects:

- **Vehicle C3, C5**: Moving in the opposite direction of where the AV is going in a different lane, no trajectories intersection would happen in the next 5 seconds.
- **Vehicle C4:** Parked vehicle out of the drivable area, no visible entry, it would take this vehicle more than 5 seconds to reach the AV.
- **Pedestrian P2:** Even if the intersection is clear of vehicles, it would take this pedestrian longer than 5 seconds to reach the other side.
- **Motorcycle m2:** driving away from where the AV currently located or where it is intending to go. To interact with the AV, it will first have to slow down, do a U-turn, then accelerate, which will take more than 5 seconds.

3.5 Summary

The problem statement of this dissertation, which is finding spatially relevant road objects, is introduced in this chapter, along with the motivations and application use cases. Also, a high-level approach is presented, as well as an example for illustrating spatial relevancy concept. The next chapter will go into details about the proposed methods for modeling the motion and computing the trajectories for both the autonomous vehicle and the road objects, explain the spatial relevancy algorithm to filter in must-track objects, and finally provide a method to order these objects based on the severity of simulated interaction with the autonomous vehicle.

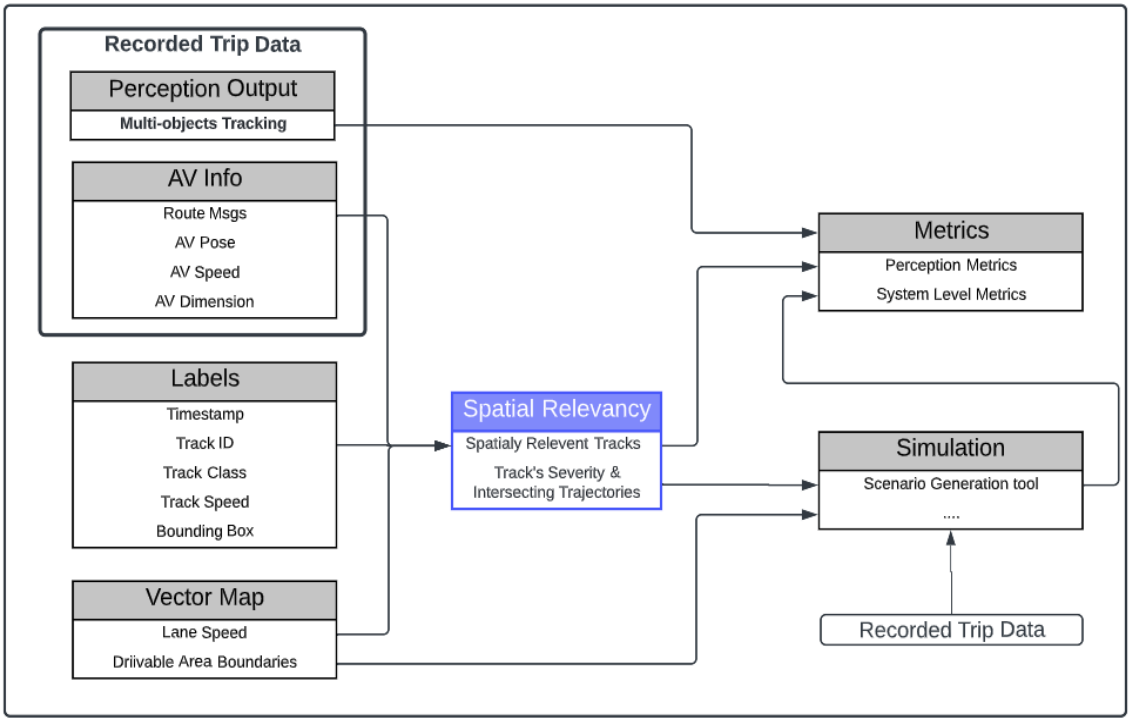


Figure 3-1: Offline applications of spatial relevancy

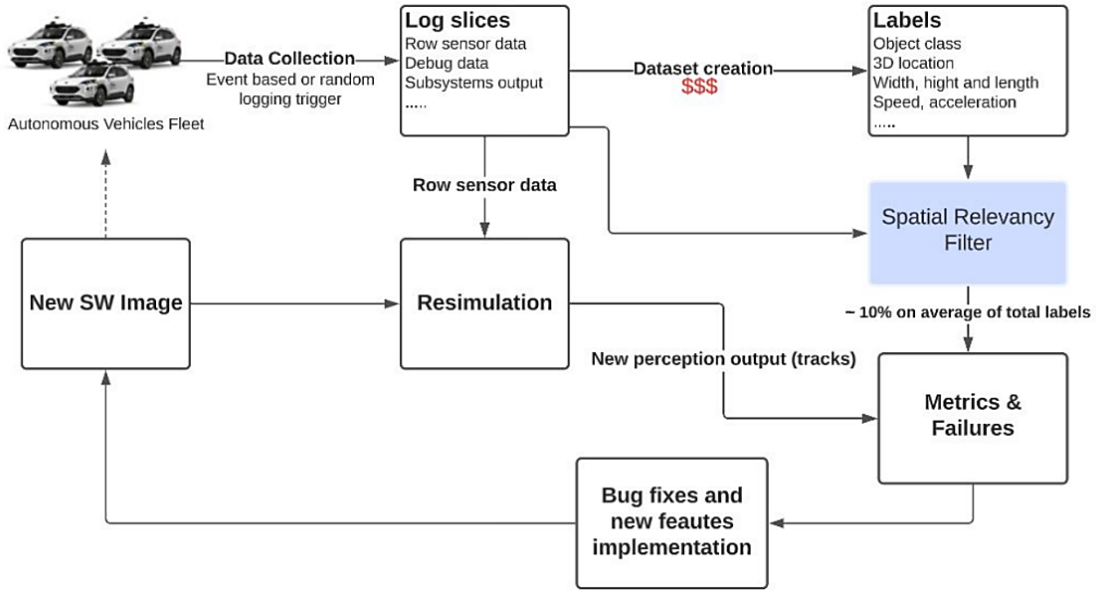


Figure 3-2: Improved evaluation of MOT.

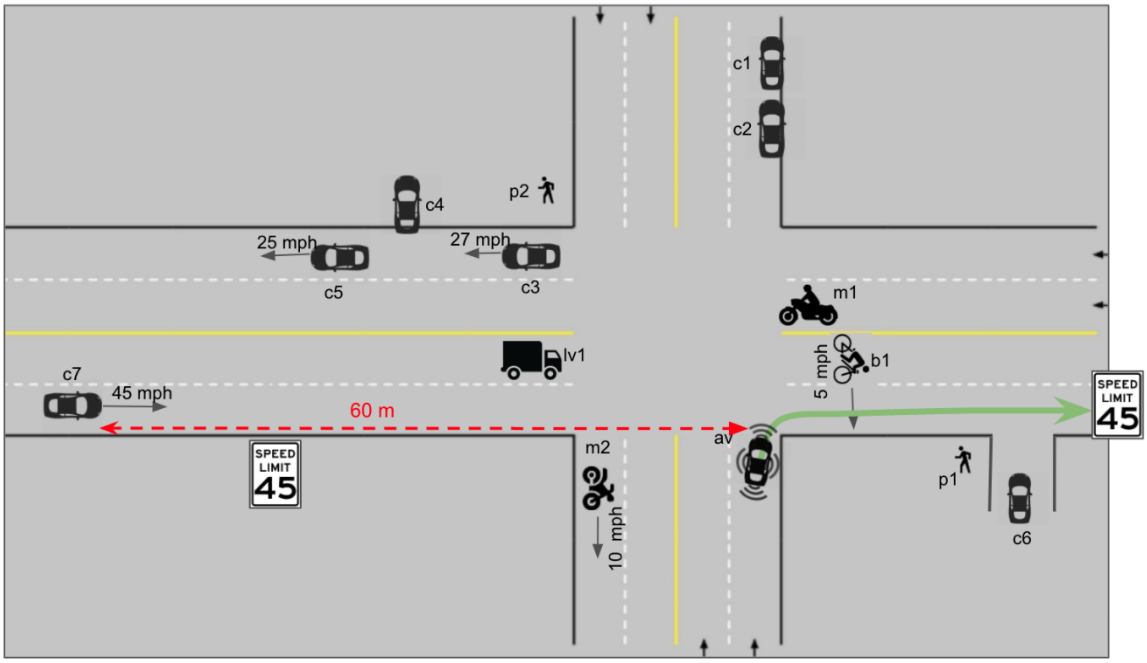


Figure 3-3: Example scene to illustrate spatial relevancy.

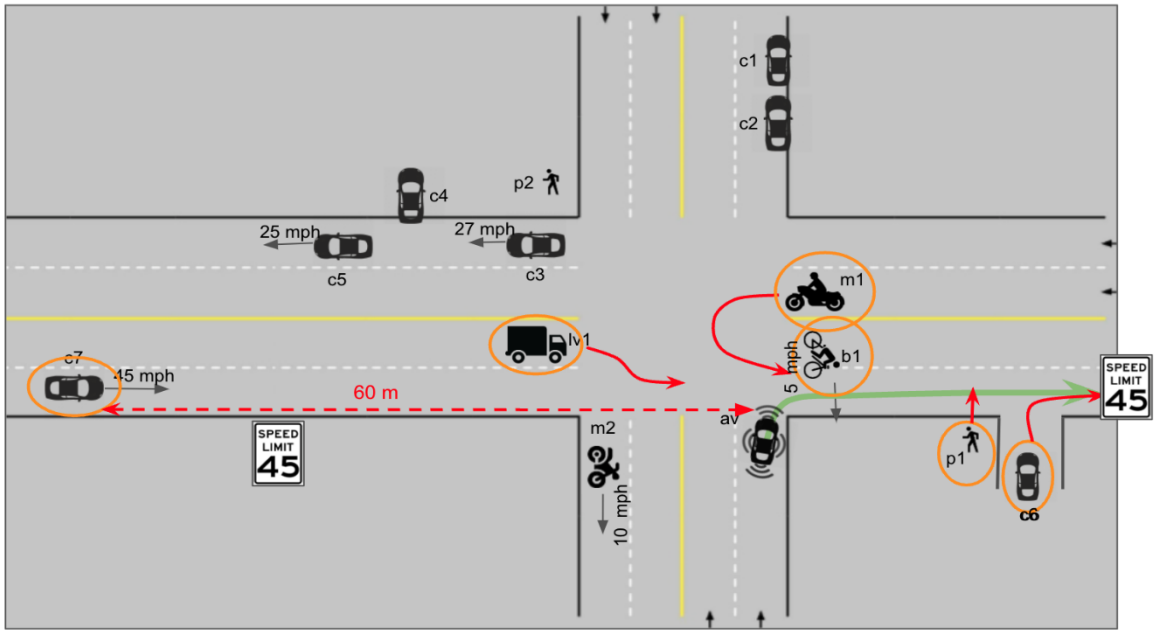


Figure 3-4: Spatially relevant objects and their trajectories

CHAPTER 4: FRAMEWORK AND IMPLEMENTATION DETAILS

4.1 Overview

In the previous chapter, we concluded that to compute spatially relevant objects, we needed to test for time and space intersection between the AV future trajectory and list of reachable trajectories for each object in the scene. To perform this intersection test, we need to model the motion of the AV as well as the motion and behaviors of the objects in a computationally feasible manner. Also, the time intersection test is discretized at a one-second interval, that is, if the autonomous vehicle and the road object could occupy the same area within one second of each other, then the road object is marked as relevant. This chapter demonstrates how this is done by first explaining the motion model of the autonomous vehicle, then the motion model and behavior of the objects in the scene, and finally the intersection algorithm. Also, in this chapter the architecture and the development process of the spatial relevancy system are explained.

4.2 Modeling the Autonomous Vehicle Motion

The AV motion is more predictable compared to the objects in the scene; that is, the spatial relevancy system has access to the planned route, which consists of the lane segments that the AV intends to traverse in order to reach its destination; once lane segments are extracted, the centerline is computed for the whole planned route as illustrated in Figure 4-1 and Figure 4-2. The following algorithm is used to extract the centerline of the planned route.

Algorithm: create_corridor_centerline

Input:

- lane segments that belong to the AV planned route.

Output: Centerline of the AV route (x, y, heading)

Step 1: Create the AV route polygon using the geometric union operation on all input lane segments polygons.
Step 2: Extract the exterior points of the AV route polygon.
Step 3: Classify each point in the list as either a left or right point.
Step 4: Create the centerline by triangulating between the left and right points, then adding the midpoint to the map_corridor_cl_list to be returned.
Step 5: Interpolate the centerline at a fixed distance (0.1 m)
Step 6: Compute the heading for each centerline point.

Algorithm 1: create corridor centerline

Then, at each time step (10 HZ), based on the current AV speed and constraints about the maximum acceleration, maximum deceleration, and maximum jerk of the AV, the AV's longitudinal deceleration to zero speed and longitudinal acceleration to lane speed for a predefined time horizon (7 seconds) are calculated. If the AV reaches the lane speed before the time horizon, it is assumed to follow the lane speed until the time horizon.

4.3 Compute Autonomous Vehicle’s Motion Profiles.

4.3.1 AV Deceleration Profile

A longitudinal trajectory optimizer that closely approximates the motion planner output used in the AV is used to calculate a longitudinal trajectory between the current AV speed (extracted from the information provided by the localization system) and a target speed of lane segment speed (extracted from the map). If target speed is reached before a configurable time horizon (7 seconds), the longitudinal trajectory is extended at

0 mph (stop in place); if zero speed is reached after the time horizon period, the trajectory is cropped at the time horizon limit.

4.3.2 AV Acceleration Profile

A longitudinal trajectory optimizer (that closely approximates the motion planner output) used in the AV is used to calculate a longitudinal trajectory between the current AV speed (extracted from the information provided by the localization system) and a target lane speed. If the target speed is reached before the configurable time horizon (7 seconds), the longitudinal trajectory is extended at a constant speed. If the target speed is reached after the time horizon, the trajectory is cut off at the time horizon limit.

For each of the motion profiles above, the trajectory is projected or interpolated on the map corridor centerline, starting from the closest point of the AV pose at the current frame. This trajectory is then sampled at 10 Hz. Information from these two motion trajectories is used to calculate the iso-temporal regions (next section) and collision severity when calculating the severity of a road object.

For example, the longitudinal acceleration trajectory for the scenario in Figure 4-2 looks like Figure 4-3, Figure 4-4 illustates Projecting this longitudinal trajectory to the computed lane centerline of the planned route lanes at the preconfigured sampling rate.

4.3.3 Compute av's Isotemporal Regions

Although the AV route is at the lane segment level, it could be very wide, and the AV might occupy any space of that route (note in Figure 4-4 how the AV's center is not aligned on the centerline). When testing for relevancy, we take this into consideration

simply by considering the whole map corridor. Also, to make sure the relevancy test is computationally cheap, we discretize the AV motion profile along the map corridor at a one-second interval. The resulting polygons are called iso-temporal regions. To find out these isotemporal regions, we use the map corridor centerline heading to draw a perpendicular line at a one-second interval. Intersecting points between these lines and the left and right boundary lines of the map corridor create the isotemporal region for the acceleration or deceleration profiles. Please note that these regions overlap with each other as the AV has a non-zero length. Figure 4-5 is an example of the isotemporal regions for the above acceleration trajectory.

4.3.4 AV's Motion Profile Clipping

When a stationary road user is on the AV's route and intersects with the AV's trajectory (acceleration or deceleration), the trajectory is clipped at the point of intersection, and consequently, the iso-temporal region is only computed until the point of intersection. The rationale behind this is that the AV can't go through this stationary object. Figure 4-6 illustrates the clipping of the AV profile.

4.4 Modeling Objects Motions

The AV encounters a wide range of objects within the ODD. Pedestrians, regular vehicles, motorcycles, and stationary objects each have a unique set of behaviors. For example, a lane change maneuver might be applicable to vehicles (regular vehicle, large vehicle, motorcycle) and bicycles but not to pedestrians. Also, entering and exiting the drivable area might be constrained for vehicles and only available through specific

places, whereas for cyclists and pedestrians, they can enter the drivable area from almost anywhere, unless there is a fence that separates the drivable area from the non-drivable area. Therefore, the SRO algorithm must consider multiple criteria ("filters") when evaluating the relevance of an object. The following table summarizes a list of behaviors and filters considered in this study:

Table 1: Objects Behaviors and Filters

	Vehicle (Large, regular)	**Motorcycle**	**Bicyclist**	**Pedestrian & Animals**	**Stationary**
Kinematic Model	Bicycle	Bicycle	Bicycle	Random Walk	Const. zero speed
Can Accelerate	Yes	Yes	Yes	No	No
Can Decelerate	Yes	Yes	Yes	No	No
Can Const. Speed	Yes	Yes	Yes	Yes	No
Can Reverse	Yes	No	No	No	No
Is Drivable Area Constrained	Yes	Yes	No	No	N/A
Use all Prior Timestamps for isotemporal intersection.	No	No	No	Yes	No

4.4.1 Acceleration Model

For any object that "Can Accelerate" a profile that simulates its acceleration from its current speed to the lane speed or maintaining its current speed if it is at lane speed is created. As we are required to take non-compliant objects into consideration, the acceleration path is modeled as driving on circles of different radii for distance d Then continue on a straight line from the last heading at the end of the circular movement. The distance d is computed as per the following:

$$d = \pi * minimum\_turning\_radius \quad [10]$$

And the *minimum_turning_radius* is computed conservatively as a linear function of length of the vehicle. Figure *4-7* illustrates this idea. In Figure *4-7*(a) the green lines all have the same length as per the above equation (circular motion), and the blue lines are the linear motion from the final heading of the circular motion. Please notice on Figure *4-7* (a) how this motion model captured an object doing a non-compliant turn in place in front of the AV, something we see frequently when doing road testing. To constrain the lateral motion, a maximum lateral acceleration parameter is introduced so that the starting radius of motion is:

$$r_0 = Max(\frac{{v_0}^2}{lat\_accel_{max}}, minimum\_turning\_radius) \quad [11]$$

The effect of this constraint is explained by comparing the acceleration profile in Figure *4-7*(a) for a stationary object to the acceleration profile of an object traveling at 30 mph (Figure *4-7*(b)). Notice how U-turns, left turns, and right turns are eliminated at this speed.

4.4.2 Deceleration Model

For any object that can decelerate, a maximum comfort deceleration and jerk are assumed, in combination with a lateral acceleration limit. Multiple trajectories are computed assuming circular motion; similar to the acceleration profiles, the starting radius of motion is computed using the following equation:

$$r_0 = Max(\frac{{v_0}^2}{lat\_accel_{max}}, minimum\_turning\_radius) \quad [12]$$

Figure 4-8 shows examples of the deceleration trajectories for objects at different speeds (a) of 11 mph and (b) of 41 mph.

Algorithm: Compute Acceleration Trajectories

Inputs:
- Lateral acceleration limit
- Initial velocity
- Deceleration limits
- Jerk limits
- Map_SE3_object
- Minimum turning radius

Output: Acceleration trajectories

Step 1: Compute the starting turning radius using the following equation:

$$r_0 = Max(\frac{{v_0}^2}{Lat\_accel_{max}}, minimum\_turning\_radius)$$

Step 2: For r = r0 to 10,000 m (nearly straight line)

Step 2.1: Compute the longitudinal trajectory that takes the road object from v_start = object's current speed to the final speed (max lane segment speed within a 20 m radius centered around the road object)

Step 2.2: If the duration of the trajectory is less than 4 seconds, extend it using constant speed until 7 seconds. If the trajectory length is more than 7 seconds, trim the trajectory at 7 seconds.

Step 2.3: Compute the (x_i, y_i) locations along the trajectory.

$$\theta_i = \frac{s_i}{r}$$
$$(x_i, y_i) = (r * sin(\theta_i), r * cos(\theta_i) - 1) \ for \ si < d$$
$$(x_i, y_i) = (y_i - s_i * sin(\theta_i), y_i + s_i * cos(\theta_i)) for \ s_i >= d$$

Step 2.4: Transform (x_i, y_i) to map coordinates using the map_SE3_object.

Step 2.5: Create a trajectory from the position and velocity information computed above.

Step 2.6: Append this trajectory to the output list.

Note: Above (x_i, y_i) corresponds to the center of the rear axle of the object.

Algorithm 2: Compute Acceleration Trajectories

Algorithm: Compute deceleration trajectories:

Inputs:

- Lateral acceleration limit
- Initial velocity
- Deceleration limits
- Jerk limits
- Map_SE3_object
- Minimum turning radius

Output: deceleration trajectories

Step 1: compute start turning radius using the following equation.

$$r0 = max(\frac{v_0^2}{lat_{acc\_limit}}, min\_turning\_radius)$$

Step 2: For r =r0 to 10,000m (nearly straight line):

Step 2.1: Compute longitudinal trajectory using bang-off-bang Acceleration, that takes the vehicle from $v_{start} = v_0 \ mph$ to $v_{end} = 0 \ mph$, using the deceleration and jerk limits input parameters.

Step 2.2: Extend the beginning of the longitudinal trajectory by $t_{latency}$ at constant speed of v_0 to simulate vehicle reaction time. The output of this step will be a longitudinal trajectory.

Step 2.3: If the duration of the trajectory is less than 7 seconds, extend it in place until 7 seconds, if the trajectory length is more than 7 seconds, trim the trajectory at 7 seconds.

Step 2.4: Compute the (x_i, y_i) locations along the trajectory:

$$\theta_i = s_i/r$$
$$(x_i, y_i) = (r * sin(\theta_i), r * (cos(\theta_i) - 1)$$

Step 2.5: Transform (x_i, y_i) to map coordinates using map_SE3_object.

Step 2.6: Create a 2D trajectory from the longitudinal trajectory projected on the (x_i, y_i) created in the previous step.

Step 2.7: append this trajectory to the output list.

Note: Above (x_i, y_i) corresponds to the center of the rear axle of the object

Algorithm 3: Compute deceleration trajectories

4.4.3 Constant Speed Profile

Many objects within the scene are not well described by accelerating and decelerating vehicle kinematic models. Pedestrians, animals, and other vulnerable road users can turn in place, moving in any direction, and moving outside of the drivable area. A constant velocity model with candidate trajectories in all directions is used for these objects. This profile allows the object to move in any direction at any speed at or below the maximum velocity defined for that object class. Any possible path that an object could take that interacts with the AV in time and space is considered relevant and assessed for collision severity.

Algorithm: Compute constant speed trajectories:

Inputs:

- object current velocity
- object velocity lower limit
- object velocity upper limit
- map_SE3_object

Output: constant velocity trajectories

Step 1: generate list of heading angles in 360 degrees around the object.

Step 2: For θ = 0 to 2π in steps of 2n/72 (5 degrees):

Step 2.1: Compute longitudinal trajectory using constant velocity.

$$s = \max(v_{min}, \min(\mathrm{v_{current}}, v_{max})) * t$$

Step 2.2: Compute the (x_i, y_i) locations along the trajectory:

$$(x_i, y_i) = (s * sin(\theta_i), s * (cos(\theta_i) - 1))$$

Step 2.3: Transform (x_i, y_i) to map coordinates using map_SE3_object.

Step 2.4: Create a trajectory from position, velocity information computed above.

Step 2.5: append this trajectory to the output list.

[1] Above (x_i, y_i) corresponds to the center of the object polygon (different from rear axle definition used in vehicle profiles)

[2] object velocity is not considered for severity calculation w/ pedestrians as the AV velocity is the driving factor in collision severity.

Algorithm 4: Compute constant speed trajectories.

The constant velocity profile determines the reachable area of the object based on the current velocity, constrained within specified upper and lower bounds. Figure *4-9* (a) shows the reachable area for an object capable moving at speed of 8.3 m/s. Any location contained within these one-second regions (inclusive of smaller regions) is considered reachable by the object. Figure *4-9*(b) shows how the reachable areas are broken up into trajectories with a small angle difference.

The computed trajectories are translated to the map frame using the object to map frame transform (Figure 4-10).

4.4.4 Stationary Objects Handling

Stationary objects like construction barrels and cons have zero velocity motion profile for the length of the horizon that is if their bounding polygon intersects with any of the AV's acceleration or deceleration iso-temporal regions, they will be considered spatially relevant.

4.4.5 Reverse Profiles

For objects that can reverse, an additional longitudinal reverse trajectory is computed for each stationary object (Figure *4-11*). The purpose of this trajectory is to emulate a vehicle reversing from a parking spot to perform a K turn. The object is assumed to accelerate to a speed of 5 mph (configurable) and then decelerate to 0.0 mph with a maximum acceleration of 0.2 g and a maximum deceleration of -0.2 g.

Similar to the deceleration trajectory, this longitudinal trajectory is interpolated into many circular paths, starting from the minimum turning radius to a straight line (radius of 10,000 m) from the left and right sides.

4.5 Spatial Relevancy Algorithm

If a road object is found to be detectable and in the drivable area, the spatial relevancy algorithm performs the following steps to determine if the object is relevant or not:

```
Algorithm: Check Relevancy
    Inputs:
        - Evaluation_list: [(ov_accel_trajectories, av_isotemporal_accel_polygon),
                            (ov_accel_trajectories, av_isotemporal_decel_polygon),
                            (ov_decel_trajectories, av_isotemporal_accel_polygon),
                            (ov_decel_trajectories, av_isotemporal_decel_polygon)]
        - Drivable area
        - Stationary road objects
        - Is drivable area constrained
        - Use all prior times for isotemporal check

    Output: List of trajectories that make the road object intersect with the AV

    Step 1: Down select the trajectories that intersect with the union of av_isotemporal_polygon.
    Step 2: Trim any trajectories that intersect with static road objects. If the object has the attribute
    "Is drivable area constrained", trim trajectories that leave the drivable area.
    Step 3: Down-select trajectories that intersect with the isotemporal polygons within their
    respective time windows. If the object has the attribute "Use all prior times for isotemporal
    check", include all intersections from the initial position of the object to the last final position
    for an isotemporal window.

    Return the output of step 3.
```

Algorithm 5: Check Relevancy

Figure *4-12* and Figure 4-13 demonstrate the spatial relevancy algorithm for both regular vehicle and pedestrian.

.

4.6 Object's Ranking Using Relative Speed.

To determine the ranking of objects based on their spatial relevance, we assign a severity rating to each object. This rating is calculated by simulating a collision between the autonomous vehicle (AV) and the object and determining the worst-case severity of the collision. To do this, we intersect the object's trajectory, as determined by the spatial relevancy algorithm, with the AV's trajectory. The severity is then calculated based on the relative velocity at the point of collision. If an object has the potential to collide with the AV at multiple locations, we take the maximum severity at all these collision points to determine the overall severity of the object. (Figure 4-14).

4.7 Compute Objects Detectability

The ground truth file includes labels for objects that may be partially occluded. However, we don't want to penalize the multi-object tracker for not tracking occluded objects below certain thresholds. To do this, we calculate a detectability score (occlusion percentage) and then use a thresholding operation to determine if a road object or actor is detectable or not. This detectability score is obtained by comparing the full extent of the object (which is usually determined by the labeler by considering multiple frames) to the portion of the object that the AV can see at the current frame. The detectability check is performed using the algorithm shown in Figure *4-15*

Algorithm check_detectability

Input:

- Road Object' visible cuboid
- full extent cuboid
- lidar_SE3_vehicle

Output: Boolean value indicates if an object/actor is detectable or not

Step-1: Transform both input cuboids from vehicle frame to Lidar frame.
Step-2: Transform each point of the cuboids (lidar frame) to azimuth-elevation frame (project to a plane perpendicular to the line of sight between the lidar and the centroid of the object), after this step all the faces of each cuboid will be transformed to polygons in the same plane.
Step-3: Merge the result polygons from each cuboid through union operation, this will result in 2 polygons: one for the visible cuboid and another one for the full extent cuboid.
Step-4: Compute intersection over union (IOU) between the two polygons
Step-5: Compare the IOU with IOU_AZI_ELE_THRESHOLD to determine detectability

Algorithm 6: Check Detectability

While the detectability score helps us avoid penalizing the perception system for occluded objects, each object will have a different distribution of IOU scores. For example, large cuboid objects (such as large vehicles or school buses) generally have a high unoccluded IOU regardless of the AV's viewing angle, while smaller irregularly shaped objects (such as construction objects or vulnerable road object) can have low IOU scores with small occlusion. For example, if there are no LiDAR returns from a pedestrian's legs due to point sparsity, the detectability score could be as low as 0.4. In this case, the object should be treated as relevant to the multi-object tracker and should be evaluated. This difference in distributions (shown in Figure 4-16) justifies the use of a class-specific threshold for detectability.

4.8 System Architecture

4.8.1 Spatial Relevancy System Inputs and Outputs

As mentioned before, the spatial relevancy system is an offline system that is used as part of the verification process, and therefore it has access to all the required inputs for

the presented algorithms to work. At a high level, the system has the following inputs and outputs.

4.8.1.1 System's Inputs

- Log slice: a snapshot of data (16–20 seconds) recorded while the AV is driving in autonomous mode. Beside the row sensors' output, the log slice contains the subsystem's outputs (e.g., AV pose, planned route, sensor's health...etc.). The system of spatial relevancy requires access only to the following information:
    - Route Planner information: that indicates the lane segments that the AV is required to follow to reach its destination.
    - Lidar Msg: This is required to make the connection between the AV pose at each time step and the corresponding ground truth labels.
    - AV Pose and Speed: This is required for creating the AV motion profiles and transforming the objects from AV local coordinates to map coordinates.
- Ground Truth Labels: For each log slice that is part of the verification dataset, a json file that contains the labels for the road actors (e.g., Road actor class, SE3 Transformation to AV Frame, speed, bounding cuboids... etc.) is created through a semi-automated process under human supervision. The labels are time-stamped and synced with the Lidar message (10 HZ), therefore a ground truth label file for a log slice of 16 seconds in length will have 160 frames.

- Vector map: sqlite file that corresponds to a log-slice used to extract geometry information needed (e.g., drivable area, lane segments that belong to the AV route, lane segment speed... etc.)
- Configuration parameters: the spatial relevancy system assumes certain motion models of the object on the road, the parameters of these models are stored in a YAML configuration file. Max long deceleration/acceleration, max centripetal acceleration, and minimum turning radius are a few examples of these parameters.

4.8.1.2 System's Outputs

- Visualization widget: mainly used for debugging the SRO system, the widgets can be loaded into Jupyter notebook through a simple interface.
    - Visualize all or part of the logslice, where it shows the drivable area, AV's route, and all the objects (color coded depending on their state; see states below).
    - Spatial Relevancy algorithm steps as shown in Figure *4-12* and Figure 4-13 Severity interaction as presented in Figure 4-14
    - Detectability Algorithm as presented in Figure *4-15*

Note: actor state could take one of the following states:

- RELEVANT_SEVERITY_LEVEL_0 #Spatially relevant
- RELEVANT_SEVERITY_LEVEL_1 #Spatially relevant
- RELEVANT_SEVERITY_LEVEL_2 #Spatially relevant
- RELEVANT_SEVERITY_LEVEL_3 #Spatially relevant
- DETECTABLE_NOT_RELEVANT #Not spatially relevant
- NOT_DETECTABLE #Not spatially relevant
- OFF_DRIVABLE_AREA #Not spatially relevant

- SRO for all objects (across all frames): to run verification on scale in an automated manner, it's important to provide the results in a textual format. This output is represented by a dictionary to be serialized, then ingested into a database to be queried for analysis (please see the next chapter) within the verification process. Figure 4-17 summarize the systems input and output.

4.8.2 Spatial Relevancy Software Block Diagram

The spatial relevancy system is designed to be a lightweight library with a straightforward interface that enables parallel processing to accelerate execution. As shown in Figure 4-18, the library is composed of various blocks and layers that work together to provide the desired functionality.

4.8.3 Spatial Relevancy High Level Flowchart:

Figure *4-19* depicts a flowchart of the system, which consists of three main steps: data preprocessing, the spatial relevancy algorithm, and outputting results. The first step involves loading the inputs and setting up the required data structures for the SRO algorithm. The second step involves running the SRO algorithm in a loop for each frame and object, with the loop being distributed across multiple processes for faster execution. Finally, the results of the algorithm are presented to the user through the visualization subsystem or in a data structure for further verification.

4.8.4 Development Process

The development process for this project followed the V-Model software development process (Figure *4-20*), which involves a series of steps from gathering requirements to testing and deployment. At the beginning of the process, a list of requirements that would achieve the project's intended goal was compiled. The design phase included the exploration of various design options, the creation of quick prototypes, and the selection of the best design that met the requirements. The chosen design was then documented through data flow architecture and interface diagrams.

The spatial relevancy system and visualizer were implemented using Python programming language and jupyter Notebook technology. To ensure the accuracy and reliability of the system, manual testing was conducted by comparing the system's output to what humans would pay attention to while driving. During this phase, numerous edge cases were identified and addressed by adding new features to the system.

In the integration step, the spatial relevancy system was combined with our metrics, and performance metrics were only calculated for spatially relevant objects. This stage included running the system on thousands of fleet logs. A design of experiment was also conducted to determine the impact of the spatial relevancy system on the Multi-Object Tracking Metrics. During the operation phase, the system will be used for verification purposes, using new data from fleet trips.

4.9 Manual and Automated Verification of the System

The manual verification of the system is conducted by reviewing the output of the algorithm and comparing it frame by frame to the images from surrounding cameras, in

addition to the camera images, information about the AV speed and the planned route, as well as each object's type, speed, acceleration, and direction of motion, is provided. Figure *4-21* illustrates this concept: on the right are the images from the cameras; on the left is the visualizer output; green objects are the spatially relevant objects; red objects are not spatially relevant objects. The red arrows connect the objects between the camera image and the visualizer output.

The automated verification of the system is performed by selecting road objects (during a trip) that were tagged by the safety driver as being important for the AV to recognize, for example because they behaved in a noncompliant way, or the AV did not react correctly to them, also by generating simulation scenario that targets corner cases for certain objects behavior that rarely occur on the road. The spatial relevancy algorithm was run on these logs and simulation scenarios, and we made sure that these objects were selected as spatially relevant in the corresponding frames. Additionally, the spatial relevancy algorithm framework is designed to efficiently run on logs produced by simulation scenarios, including those that may be challenging. We ran the spatial relevancy algorithms on these scenarios and ensured that objects within them were accurately flagged as spatially relevant.

4.10 Summary

We presented the algorithms that will enable us to evaluate the proposed approach. We started by modeling the AV motion profiles as well as the road objects' motion profiles, taking into consideration non-compliant scenarios. Then we provided an example of the algorithm that determines if an object is spatially relevant or not. We also

presented a method to classify and rank objects by the severity of simulated interactions. To prevent penalizing the multi-object tracking system for objects that are occluded, we introduced the concept of detectability and presented an algorithm to compute a detectability score as well as a data driven approach to determining a detectability threshold based on object type. Finally, the architecture and the development process of the spatial relevancy system were explained.

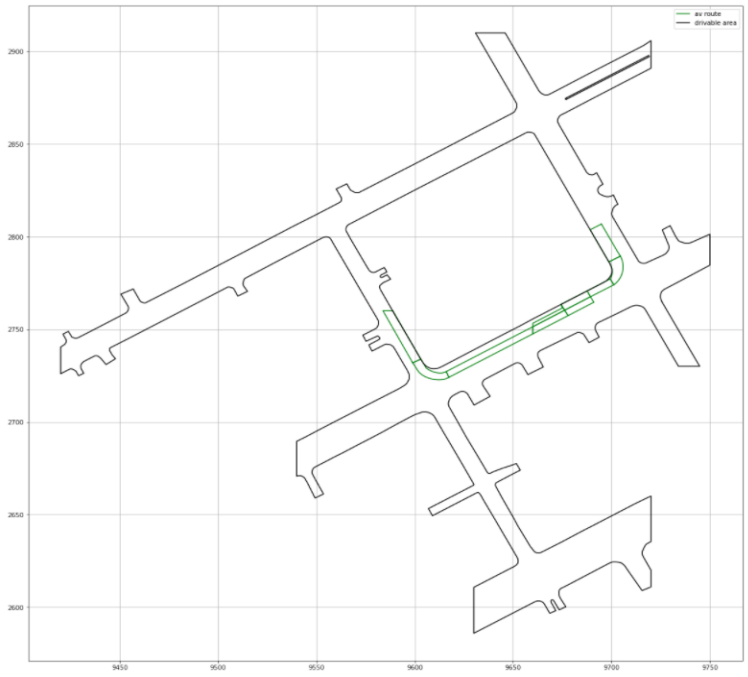


Figure 4-1: e.g of planned route for an AV (green) inside drivable area (black)

Figure 4-2: Approx. of AV path by computing the centerline of the AV route.

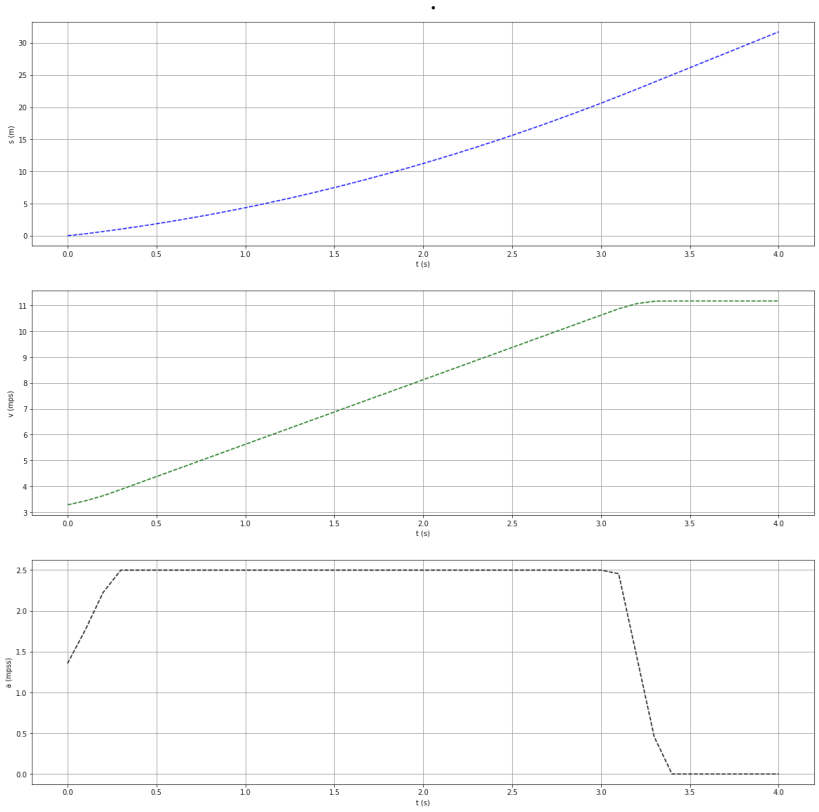


Figure 4-3: Long trajectory for right Turn

Figure 4-4: Projecting the long trajectory on centerline of the trajectory.

Figure 4-5: Isotemporal regions

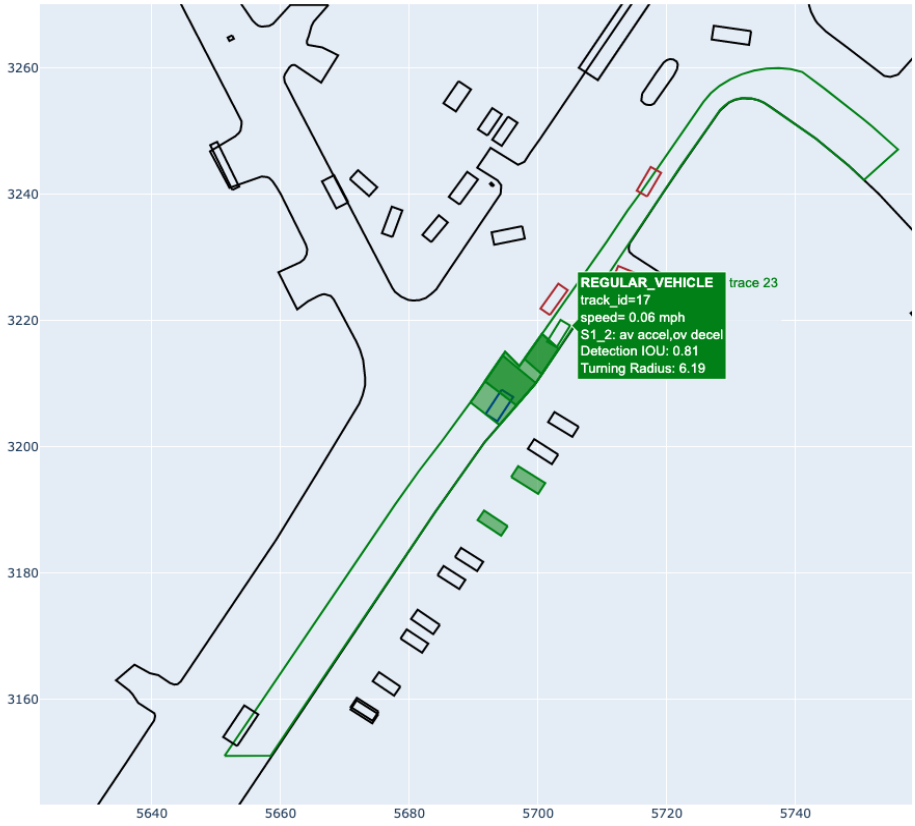
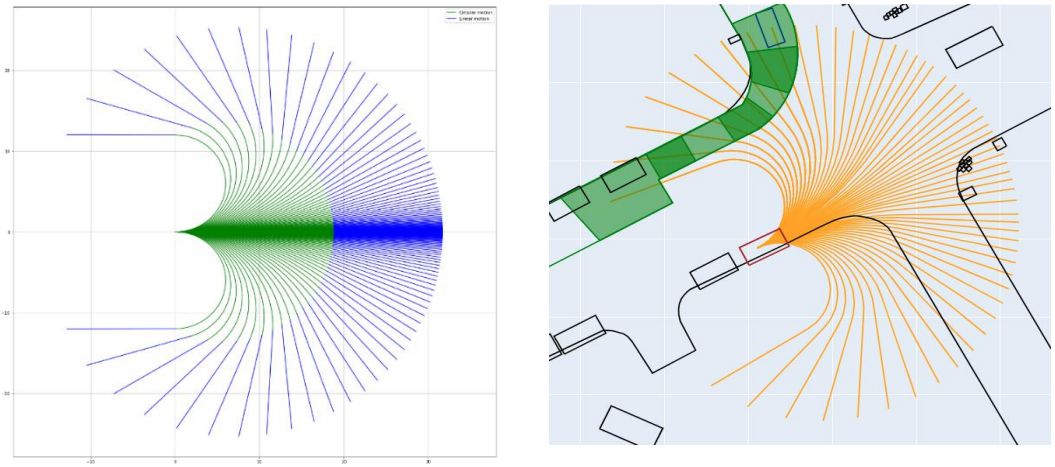
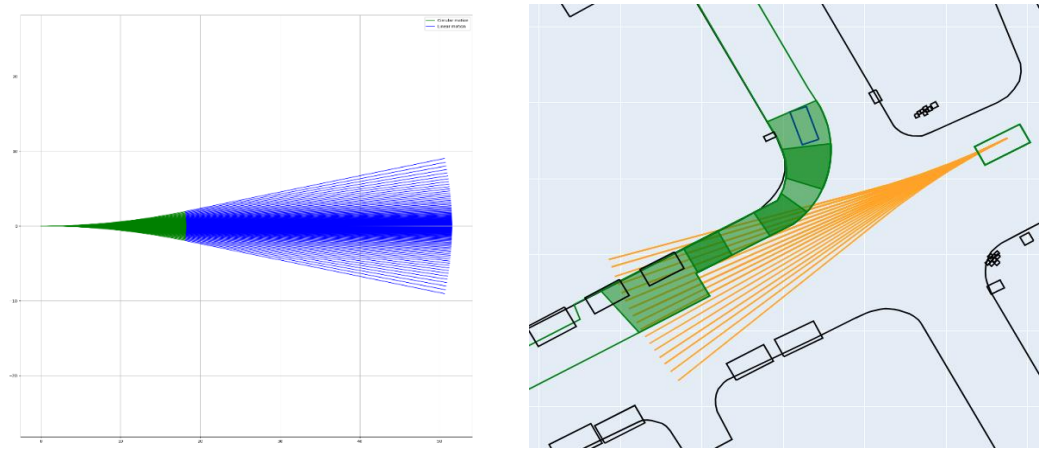


Figure 4-6: Illustration of AV motion profile clipping

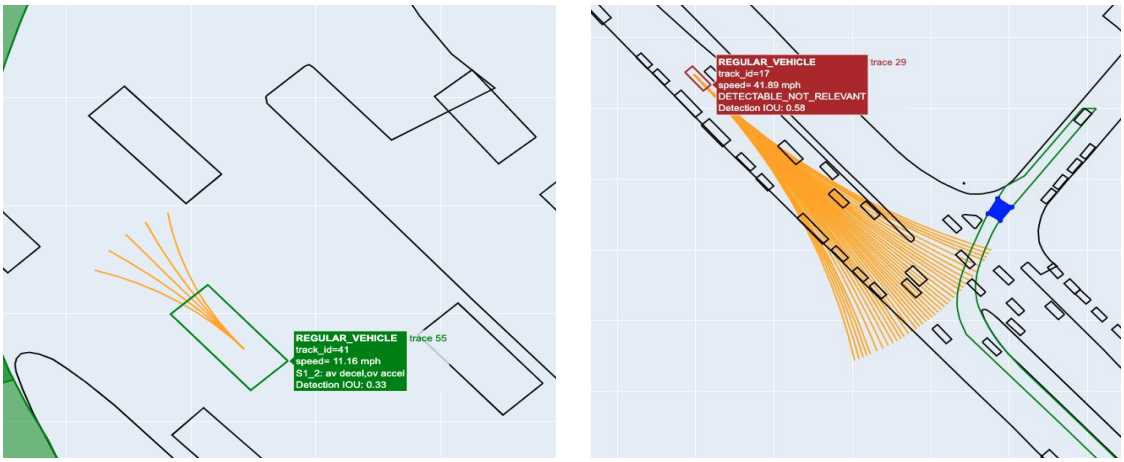


(a)



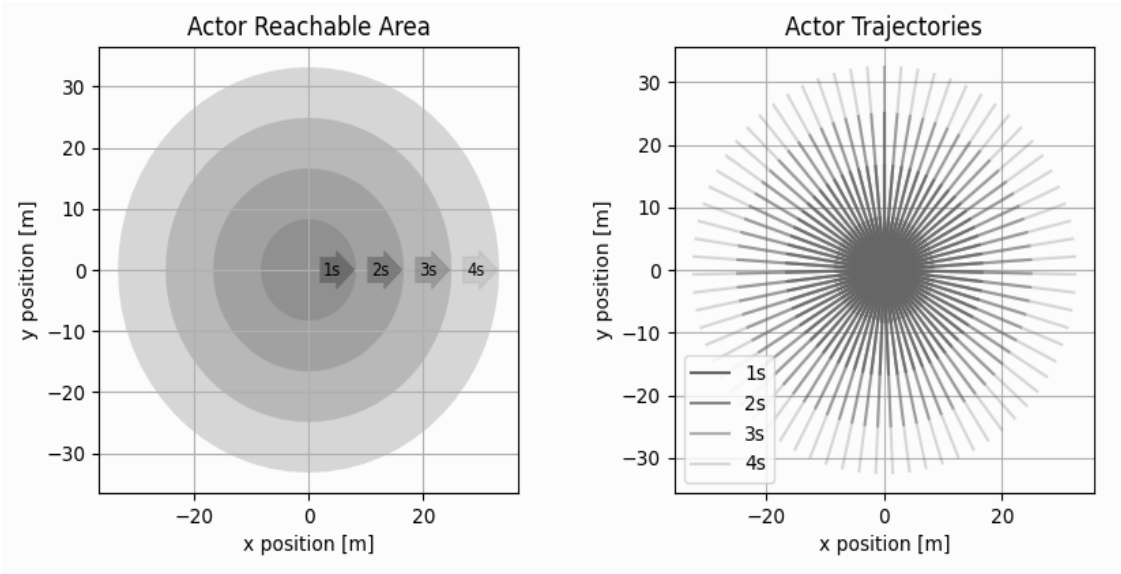
(b)

Figure 4-7: object's acceleration trajectory



(a) (b)

Figure 4-8: Deceleration profile for objects at different speeds.



(a) (b)

Figure 4-9: Constant velocity trajectory generation for 15 mph object

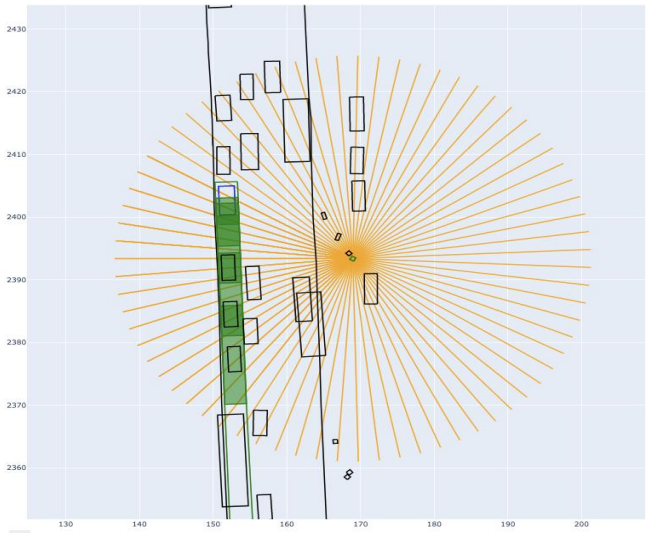


Figure 4-10: pedestrian object trajectories in the map frame

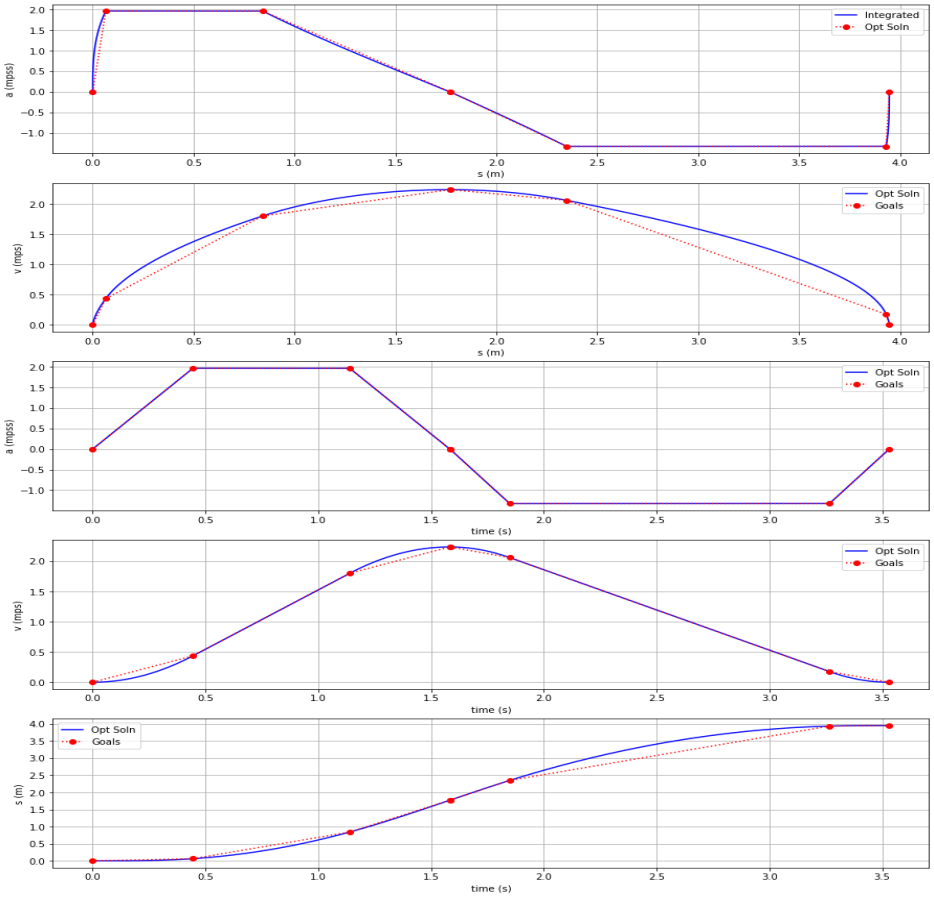
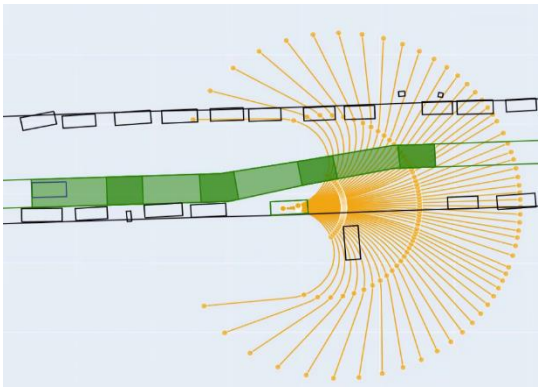
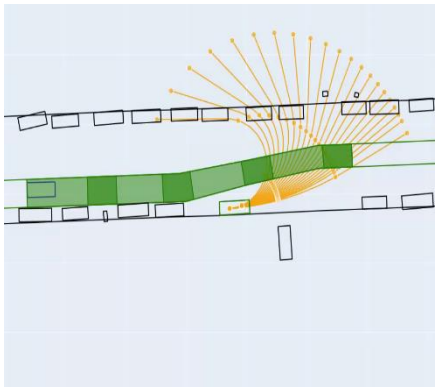


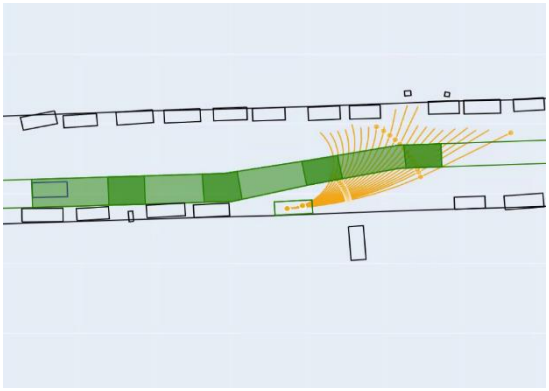
Figure 4-11: Longitudinal reverse trajectory



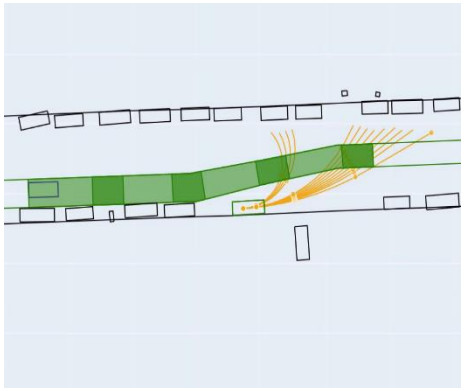
input



step-1



Step-2



Step-3

Figure 4-12:Spatial Relevancy Algorithm

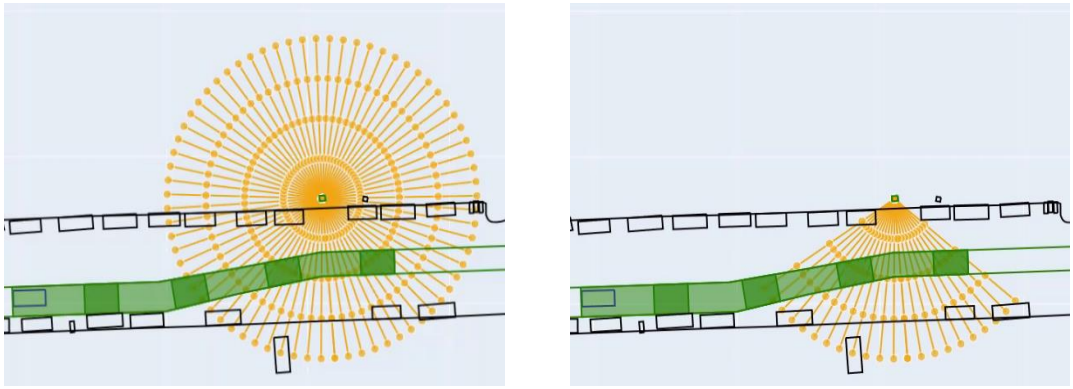


Figure 4-13: Spatial Relevancy Algorithm for Pedestrian type

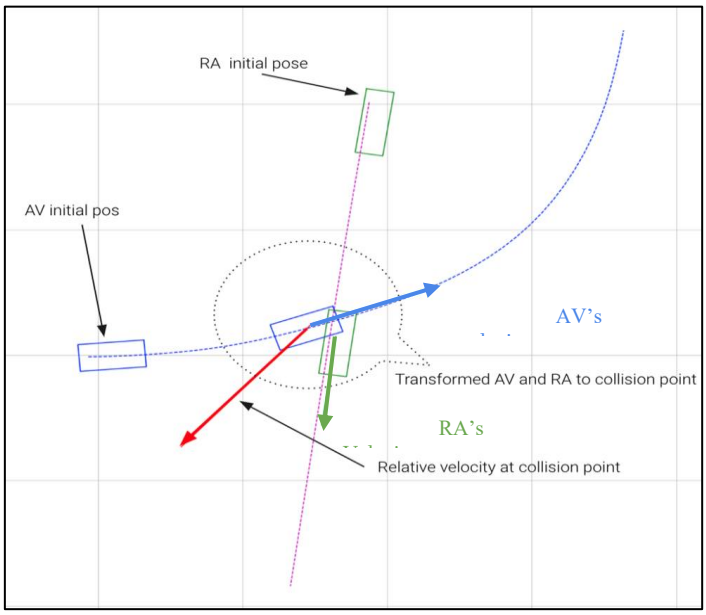
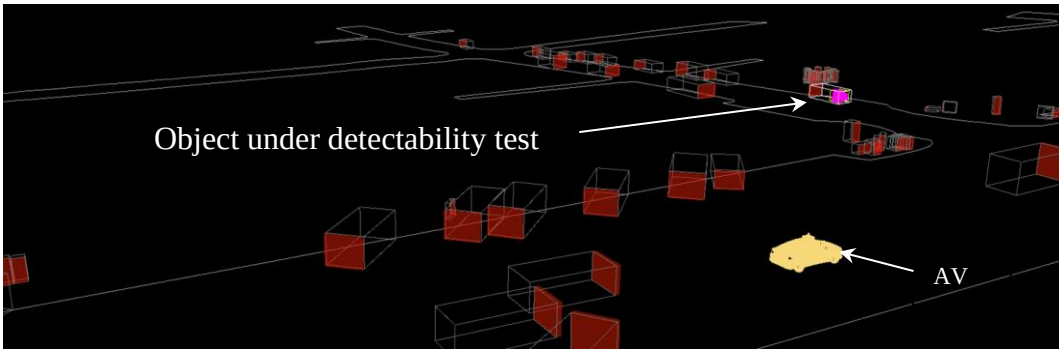
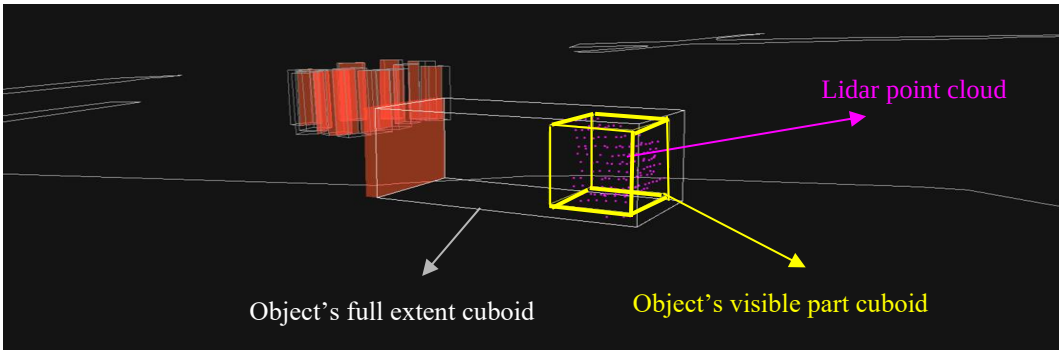


Figure 4-14: Severity Calculation

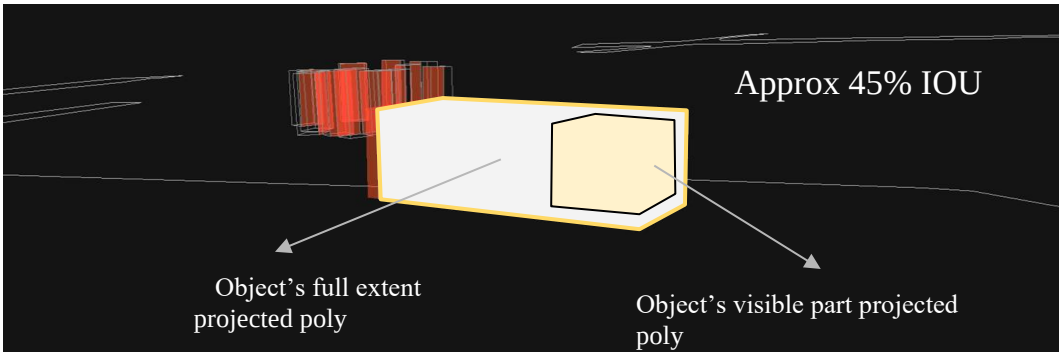
(a) The object under detectability test in camera view



(b) The object under detectability test relative to the AV



(c) Full extent cuboid vs visible part cuboid



(c) Projection to the Azimuth-Elevation frame

Figure 4-15: Illustration of Detectability Algorithm

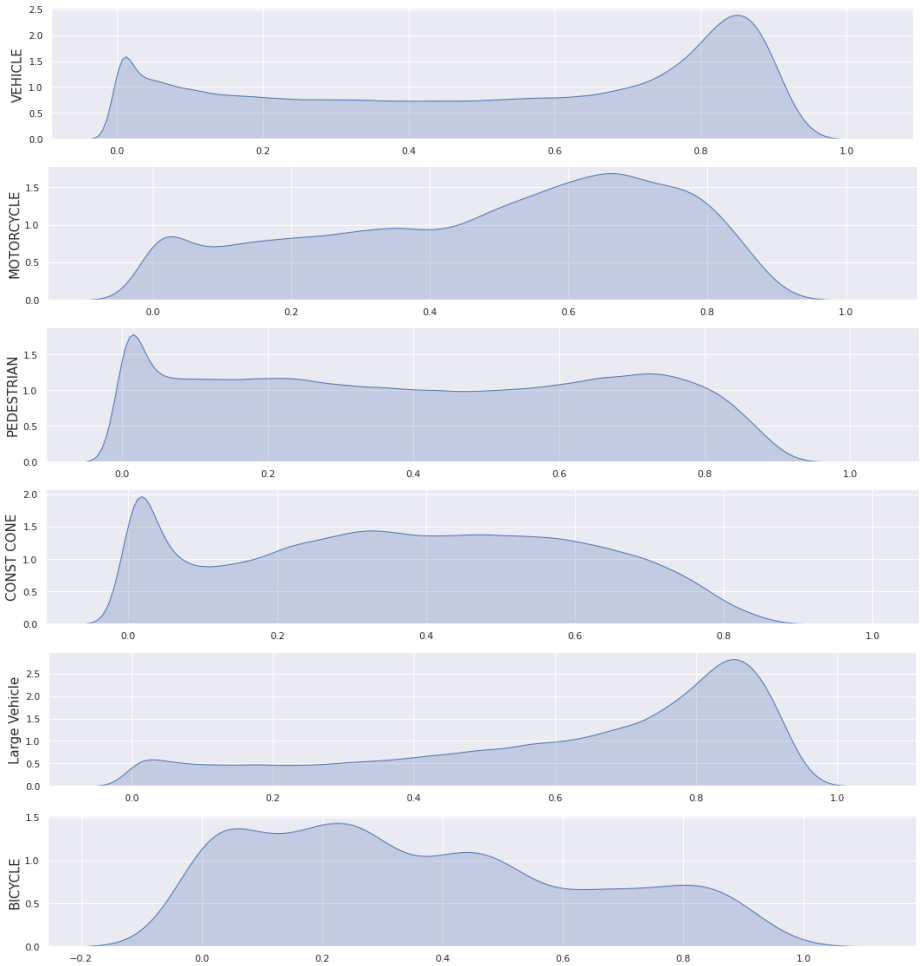


Figure 4-16: Detectability KDE by Category

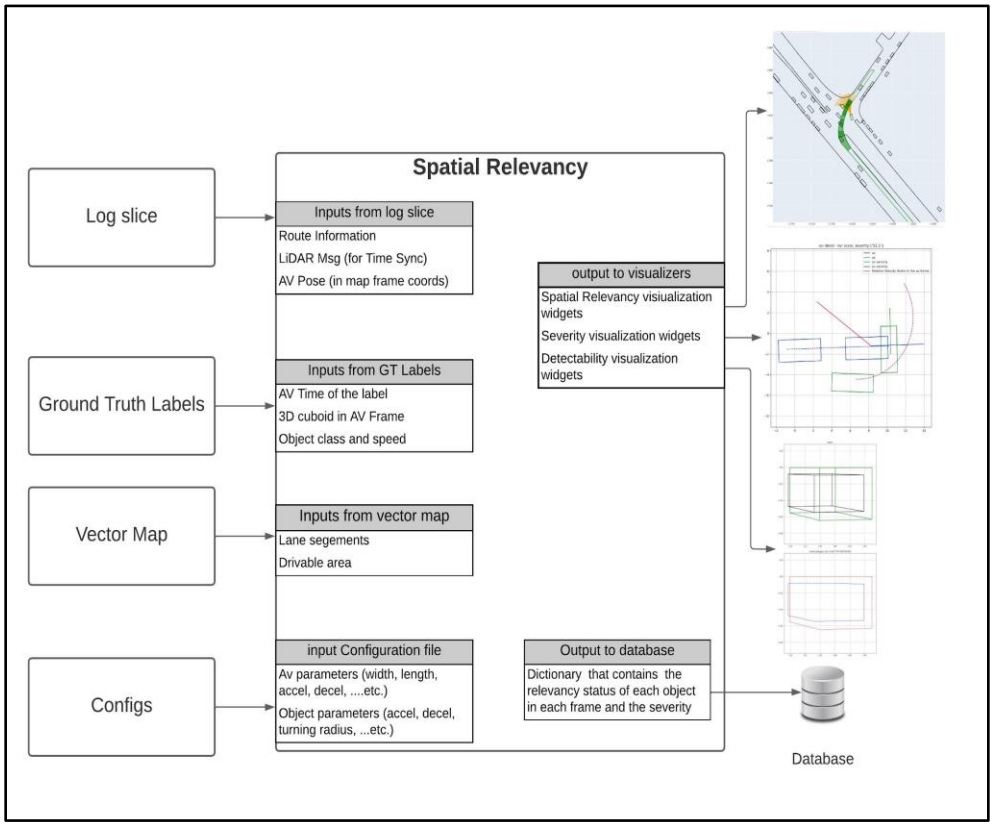


Figure 4-17: Spatial Relevancy system inputs and output

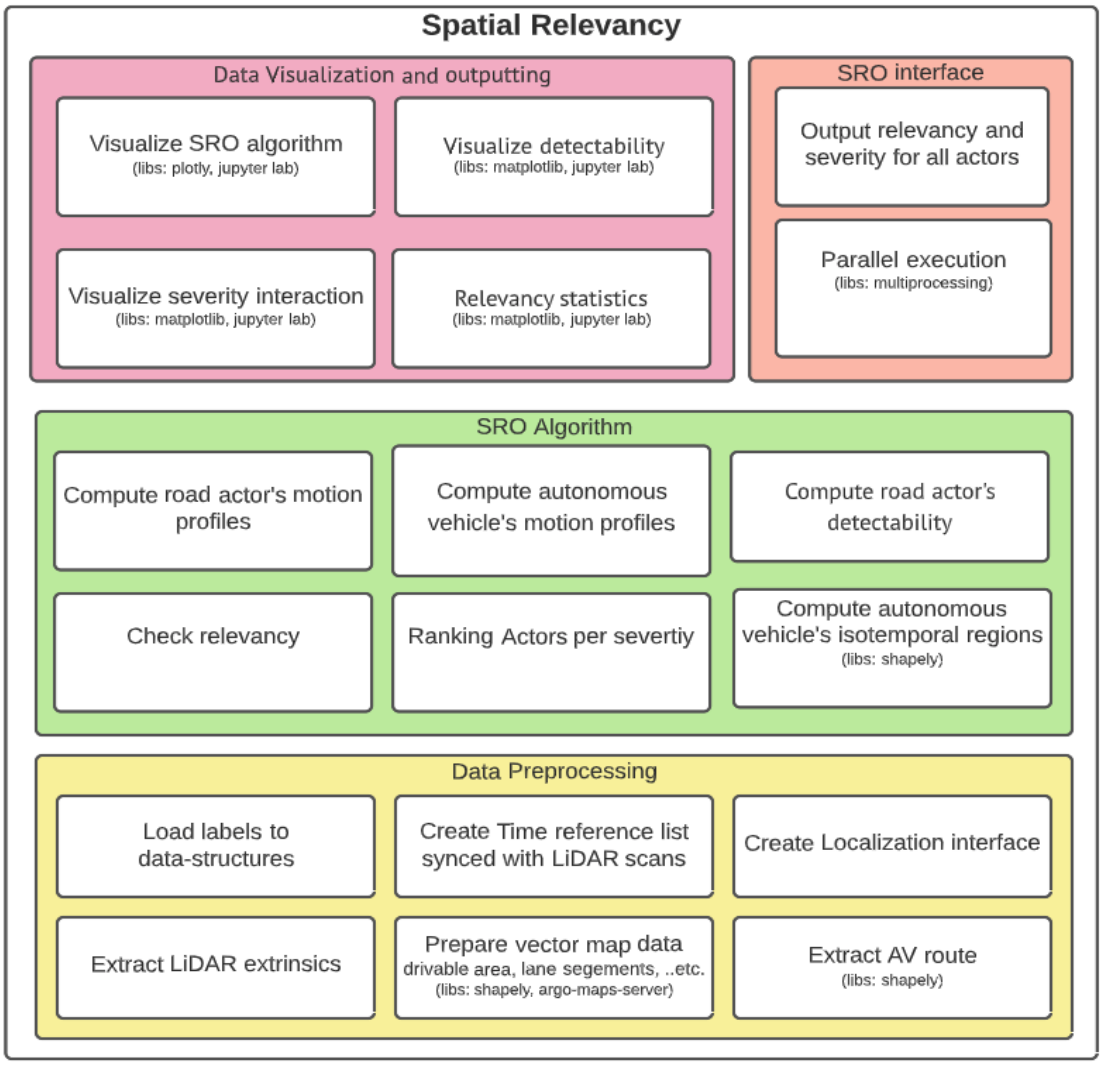


Figure 4-18: Spatial Relevancy System Architecture Diagram

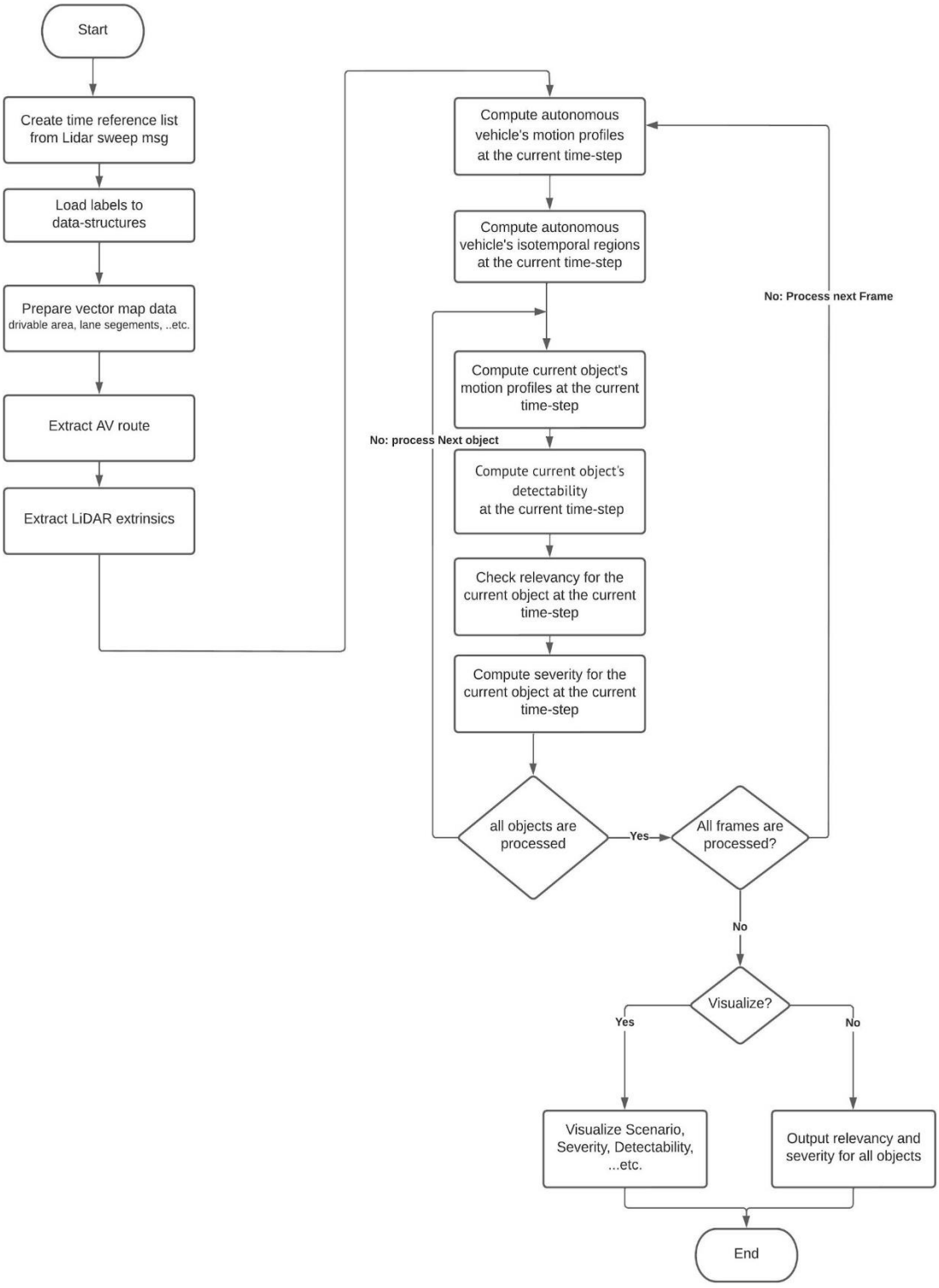


Figure 4-19: Spatial Relevancy High Level flowchart

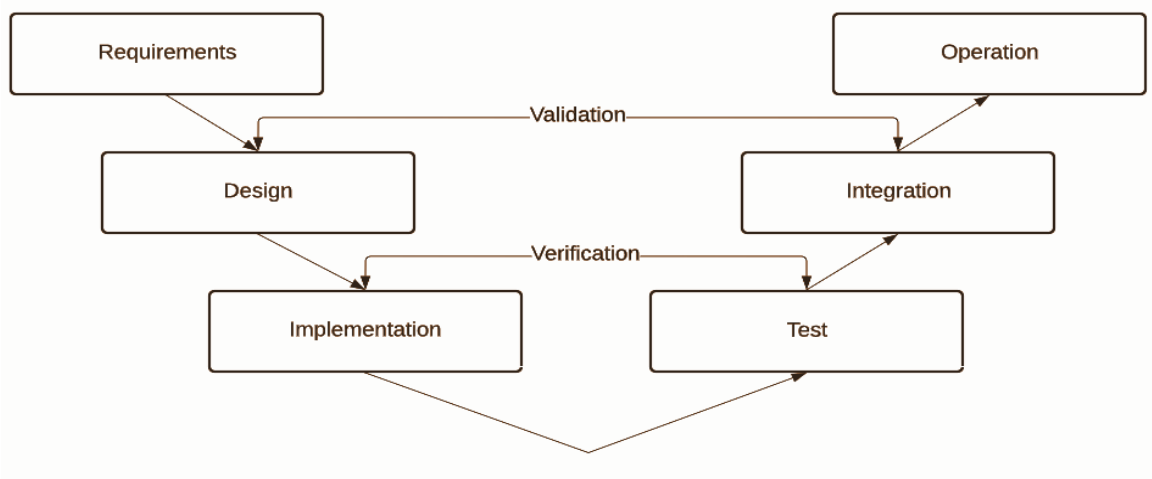


Figure 4-20: Simplified V Software Development Process

Figure 4-21: Manual verification using camera images.

CHAPTER 5: SYSTEM EVALUATION

5.1 Overview

In this chapter, we will be evaluating the performance of our spatial relevancy filter on a dataset collected from an urban environment in two cities: Miami and Austin. The data was collected using our autonomous vehicle fleet, which allows us to assess the effectiveness of our framework in a realistic setting. To evaluate the effect of our spatial relevancy filter, we will study how it impacts sensing requirements and compare detection metrics such as accuracy, precision, recall, and F1 score before and after applying the filter. We will also compare multi-object tracking metrics, such as CLEAR-MOT and IDF1, with and without applying the spatial relevancy filter to determine its impact on tracking performance. Through this evaluation, we aim to provide compelling evidence of the effectiveness and usefulness of our spatial relevancy filter in improving the development of the perception system for autonomous vehicles.

5.2 Dataset Description

The dataset used in this evaluation was collected using the Ford Escape (Figure 1-4) Autonomous Vehicle fleet in both Austin and Miami. It contains a total of 1567 log-slices, each approximately 18 seconds in length and labeled at a rate of 10 Hz. These log-slices were randomly recorded on urban roads with high traffic density and intersections, as shown in Figure 5-1. This type of environment provides a challenging and realistic testbed for evaluating the performance of the autonomous vehicle system. However, it's worth noting that the results presented in this chapter would likely be different if the dataset were selected from a highway environment with high-speed traffic. This is

because the characteristics of the environment, such as the speed, density of traffic, and the type of actors can significantly impact the results of the spatial relevancy filter. As such, it's important to carefully consider the characteristics of the environment when applying spatial relevancy for evaluating the performance of the perception system of an autonomous vehicle.

The dataset is a total of 8 hours of sensors data, Figure 5-2 shows the distribution of sequences across geographical areas in Miami and Austin, statistics about the distribution of the time of day for the log-slices as well as the distribution of log-slices duration in seconds. It also shows the distribution of object classes, with the ‘vehicle’ class dominating the dataset.

Figure 5-3 illustrates the distribution of the number of unique object classes, including vehicles, motorcycles, bicycles, large vehicles, pedestrians, and construction objects, across the log slices in the dataset.

Finally, Figure 5-4 shows the distribution of speed for the moving objects in the following classes: vehicles, motorcycles, large vehicles, and bicycles. These figures provide valuable insights into the characteristics and patterns of the dataset and will be useful in evaluating the performance of the framework.

5.3 Evaluation Results

In this section, we investigate the impact of applying a spatial relevancy filter at the sensing stage of a perception system on the dataset described previously. Specifically, we examine how the filter affects the total number and spatial distribution of objects that are included in the verification process and how these changes impact the field of view

requirements of the autonomous driving system. We also compare the results of classification and detection tasks with and without the use of the spatial relevancy filter by computing a confusion matrix and various metrics. Finally, we evaluate the effect of the filter on multi-object tracking performance by comparing the corresponding metrics. For this evaluation we utilized an experimental perception stack that was integrated into our autonomous driving system. We conducted extensive road testing within our ODD using one of our fleet vehicles to verify the performance of this perception stack which performed exceptionally well without the need to perform any takeover by the safety driver.

5.3.1 Number of Objects and Sensing Requirements:

The spatial relevancy filter was applied to a dataset consisting of log slices that each lasted 18 seconds. On average, the filter retained only about 11% of the objects within these log slices. However, in a small number of instances, the filter retained a higher percentage of objects, ranging from 30-50% of the total number of objects. These results suggest that the spatial relevancy filter is very effective at reducing the number of objects in a log slice, though there are some exceptions where a larger percentage of objects are retained (*Figure 5-6*).

Figure 5-5 is a polar plot that compares the detections in a dataset with and without the application of a spatial relevancy filter. The plot shows all detections relative to an autonomous driving vehicle (AV), which is located at the center of the plot and heading towards zero. One of the key benefits of applying the spatial relevancy filter is confirming the field of view required for the AV to operate in its target environment. For

instance, the sensors used to collect the dataset in Figure 5-5 are capable of detecting objects up to 300 meters away and covering a full 360 degrees. However, in most cases, the objects that could potentially interact with the AV are only within a range of 150 meters, and the necessary sensing coverage may only be required at certain angles, rather than a full 360 degrees. This means that fewer sensors are needed, and the sensors can be placed in a way that smoothly integrates and retrofits with the vehicle. These factors contribute to hardware cost savings and faster scaling to production. Figure 5-7 displays polar plots separated by actor type (in this case, regular vehicles, pedestrians, and motorcycles). It is worth noting that the data for the class of regular vehicles is saturated, but more data is needed for the other rarer objects in order to finalize the field of view or sensor placement.

5.3.2 Misdetection Error

We conducted an analysis of the detection error in our previously specified dataset by comparing the results with and without the application of a spatial relevancy filter. As depicted in Figure 5-8(a), there were two histograms: one for all the non-occluded objects (orange) and one for what our perception system was able to detect without applying the spatial relevancy filter. From this analysis, we found that approximately 30% of the objects were misdetected in a system that was otherwise perfect and had no failures caused by the perception system. However, after applying the spatial relevancy filter, as shown in Figure 5-8 (d), there was a significant reduction in the percentage of missed detections that the development team needs to address. This demonstrates that the use of the spatial relevancy filter helps to reduce the number of

issues that need to be fixed, leading to a faster development cycle. Figure 5-8 (b) and (d) show the locations of these misdetections without and with the application of the spatial relevancy filter, respectively, relative to the AV. It's important to note that these are frame-level misdetections, with no frame-to-frame association or smoothing.

5.3.3 Misclassification Error

An analysis of the classification error was conducted by comparing the results with and without the application of a spatial relevancy filter. As shown in Figure 5-9(a) and (c), the ratio of misclassified objects to all detected objects was relatively unchanged with the use of the filter. However, the number of misclassifications that required correction was significantly reduced, as shown in Figure 5-9(b) and (d).

The findings from the previous sections can help the development team prioritize issues in their backlog and enable a more efficient development process. When the development team starts looking at each individual failure after running a Resimulation of the original log (Figure 3-2) to address them, they can avoid spending time trying to fix problems related to objects that do not affect safety. Instead, they can focus on failures related to spatially relevant objects, which is a much smaller subset, as shown in Figure 5-8 and Figure 5.9 when comparing the polar plots (b) without spatial relevance and (d) with spatial relevance.

5.3.4 Classification metrics

Precision, recall, and F1 score are all evaluation metrics used to assess the performance of a classification model. They are commonly used to measure the

performance of a binary classifier but can also be extended to multi-class classification problems.

Precision is defined as the number of true positive predictions made by the model, divided by the total number of positive predictions made by the model:

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives} \quad [13]$$

Recall is defined as the number of true positive predictions made by the model, divided by the total number of actual positive cases in the dataset:

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives} \quad [14]$$

F1 score is the harmonic mean of precision and recall, and is calculated as follows:

$$F1 = 2 * \frac{Precision\ *\ Recall}{Precision\ +\ Recall} \quad [15]$$

The F1 score is a balanced measure of precision and recall and is often used as a single metric to compare the performance of different classification models. A model with a higher F1 score is generally considered to be better, as it is able to achieve a balance between precision and recall.

We noticed that when computing the confusion matrix and classification metrics with and without the application of the spatial relevancy filter, the filter did not improve the metrics for classes of objects with high occurrence in the environment, such as

regular vehicles and pedestrians. However, for classes of objects with low occurrence, such as large vehicles, motorcyclists, and bicyclists, the spatial relevancy filter did improve the classification metrics, as shown in the following table.

Table 2: Classification Metrics

	Recall		Precision		f1_score	
	Without SRO	With SRO	Without SRO	With SRO	Without SRO	With SRO
BICYCLIST	0.81	↑0.92	0.76	↑0.87	0.78	↑0.89
LARGE_VEHICLE	0.71	↑0.90	0.81	↓0.78	0.76	↑0.84
MOTORCYCLIST	0.65	↑0.82	0.67	↑0.79	0.66	↑0.81
PEDESTRIAN	0.94	↓0.92	0.96	↑0.98	0.95	0.95
VEHICLE	0.98	0.98	0.98	↑0.99	0.98	↑0.99

5.3.5 Tracking Metrics

In this section, we investigate the impact of applying a spatial relevancy filter on multi-object tracking metrics, specifically Multi-Object Tracking Accuracy (MOTA), Multi-Object Tracking Precision, and the Identification Metrics (IDF1). We compare these metrics with and without the spatial relevancy filter applied. For this experiment, we only computed the performance on objects of the "regular_vehicle" class, which is highly representative in our dataset. The metrics were calculated at the log-slice level, and a final overall metric was computed for the entire dataset. It is worth noting that we used the Euclidean norm squared between the center of the tracker output and the label, with a gating window of 5 meters, for matching ground truth labels to the tracker output. Figure 5-12 illustrates the distribution of MOTA in the dataset with and without the application of the spatial relevancy system. The figure shows that the overall score

improved by 19.44%, and the new distribution has a lower standard deviation It is important to note that MOTA measures three types of tracking errors: false positives (FP), false negatives (FN), and ID switches (IDSW). The MOTP overall score did not improve, but this is expected as this metric only measures the localization accuracy of the detector (as shown in Figure 5-13) and the localization error without and with the application spatial relevancy did not change illustrated in Figure *5-14*. Finally, IDF1, which emphasizes association accuracy over detection accuracy, showed an overall score improvement of 13.96% after the application of the spatial relevancy. The distribution of this metric also had a lower standard deviation, making the new score more representative (as shown in Figure *5-15*). The significant improvement in the MOT metrics when computed using only spatially relevant objects, as opposed to all the ground truth labels, confirms our hypothesis that excluding system tasks (such as automated driving in our case) can render these metrics non-representative of the system's actual performance (as mentioned earlier the experimental perception stack used for this evaluation did not cause any safety issue when it was deployed to one of our fleet vehicle for public road testing) . Another important conclusion from this result is that because none of the filtered-out objects (i.e., not spatially relevant objects) could collide with or reach the autonomous vehicle, even if the perception system failed to track it, the computed metrics now consider safety and could be used to support safety claims in the safety case Finally, similar to misdetection and misclassification, the fact that the blue histogram (with spatial relevance) has a higher mean and lower standard deviation compared to the orange histogram (without spatial relevancy) in Figure 5-12 and Figure

5-13 shows that less work is needed to achieve the performance goal where the MOT system is safe and ready for production.

5.3.6 Object Velocity and Heading Errors

In this section, we study the velocity and heading errors of associated objects when using a spatial relevancy filter, compared to not using the filter. While object velocity and speed are not typically considered in the evaluation of a perception system, they are important in the context of autonomous driving applications. In these systems, velocity and heading are part of the state vector, and are used by other subsystems such as prediction and motion planning. Errors in velocity and heading can have significant impacts on these subsystems and potentially cause issues such as causing other objects to aggressively brake to avoid a collision with the AV or predicting an incorrect lane change intention of other vehicles.

To study the impact of the spatial relevancy filter on velocity and heading errors, we used Root Mean Square Error (RMSE) with respect to range. We find that the error curve after applying the spatial relevancy filter closely follows the error curve without the filter, but with lower confidence due to the smaller number of samples retained after applying the filter. This result suggests that the spatial relevancy filter does not significantly impact the velocity and heading errors, but it also highlights the need to increase the accuracy of the system to accurately estimate these values. Overall, this analysis helps to improve the development of the perception system for autonomous vehicles. Figure *5-16* and Figure 5-17 illustrates the velocity and heading error curves with and without spatial relevancy.

5.4 Summary

In this study, the performance of a spatial relevancy filter was evaluated on a dataset collected from urban environments in Austin and Miami. The data was collected using autonomous vehicles, allowing for a realistic evaluation of the filter's effectiveness. The filter's impact on sensing requirements and detection metrics such as accuracy, precision, recall, and F1 score were compared before and after its application. The filter's effect on multi-object tracking metrics, such as CLEAR-MOT, and IDF1, was also studied to determine its impact on tracking performance. The results showed that the filter significantly reduced the number of objects that needed to be checked as part of the verification process, resulting in a reduction in sensing requirements and computational load. The filter also improved detection and tracking performance, as evidenced by improved classification and detection metrics and multi-object tracking metrics. Overall, the study demonstrated the effectiveness and usefulness of the spatial relevancy filter in improving the development of the perception system for autonomous vehicles.

(a) Miami (b) Austin

Figure 5-1: Geometry of the roads in the evaluation dataset

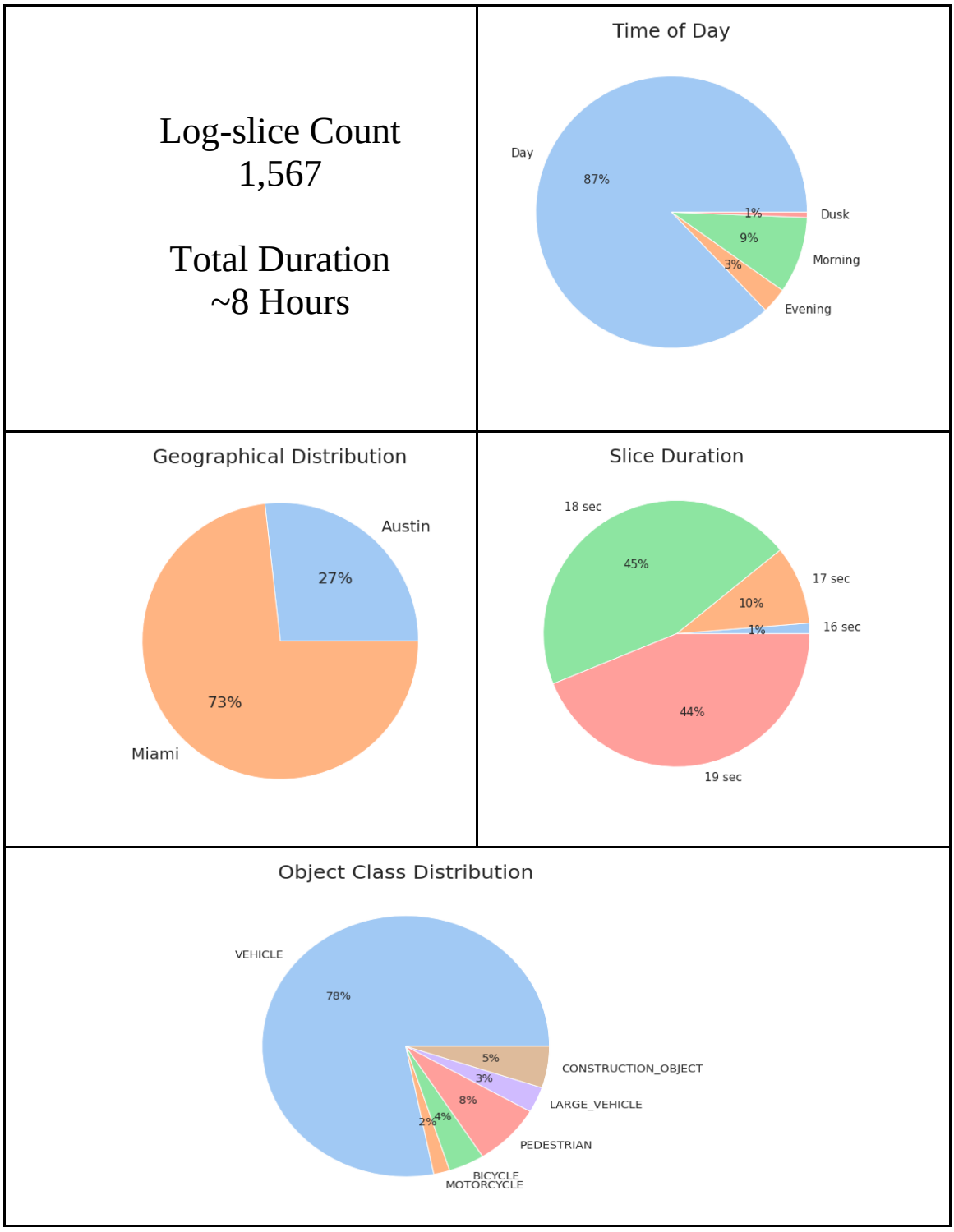


Figure 5-2: Evaluation Dataset Statistics

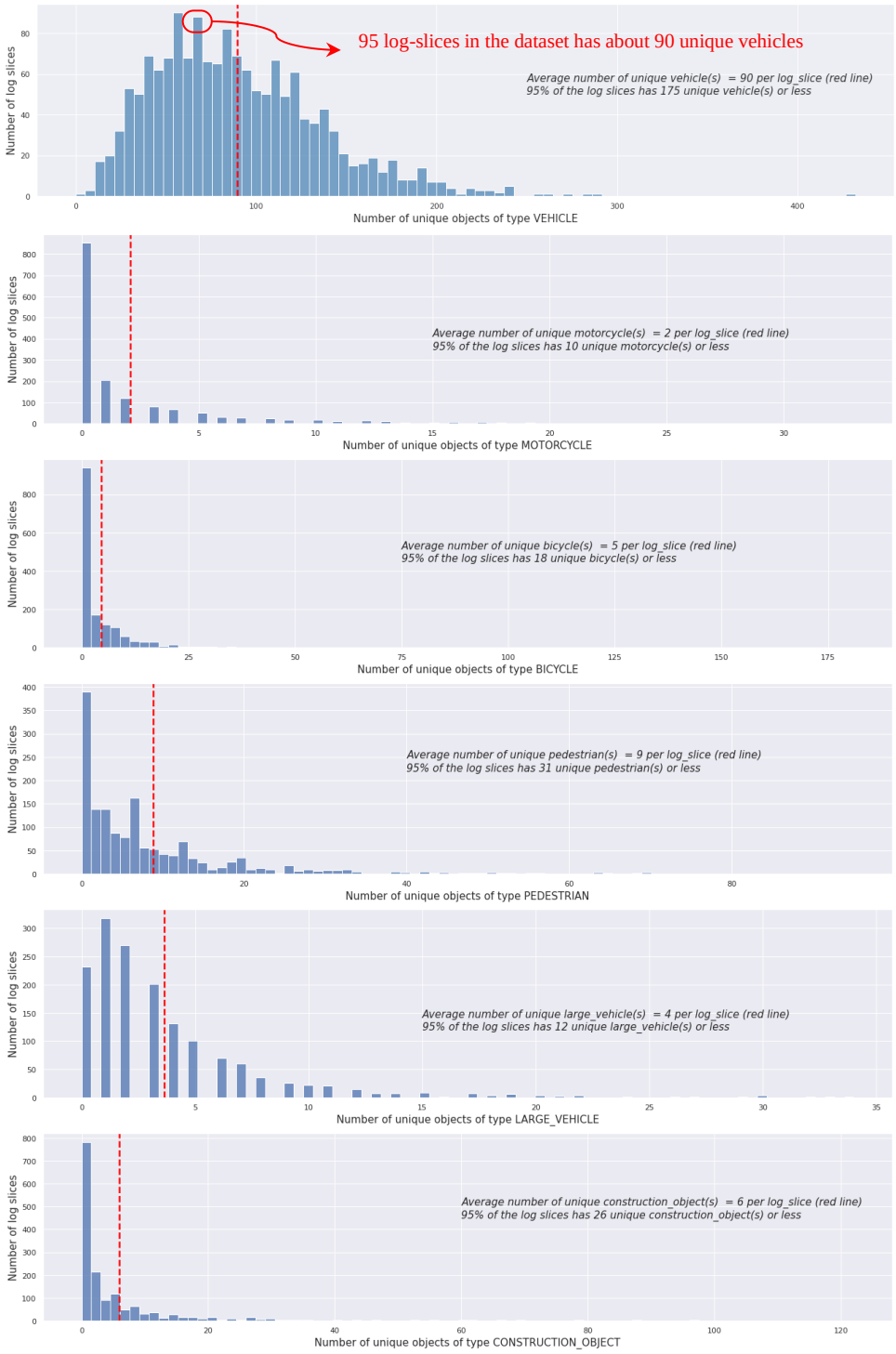


Figure 5-3: Histogram of the number of unique objects per slice

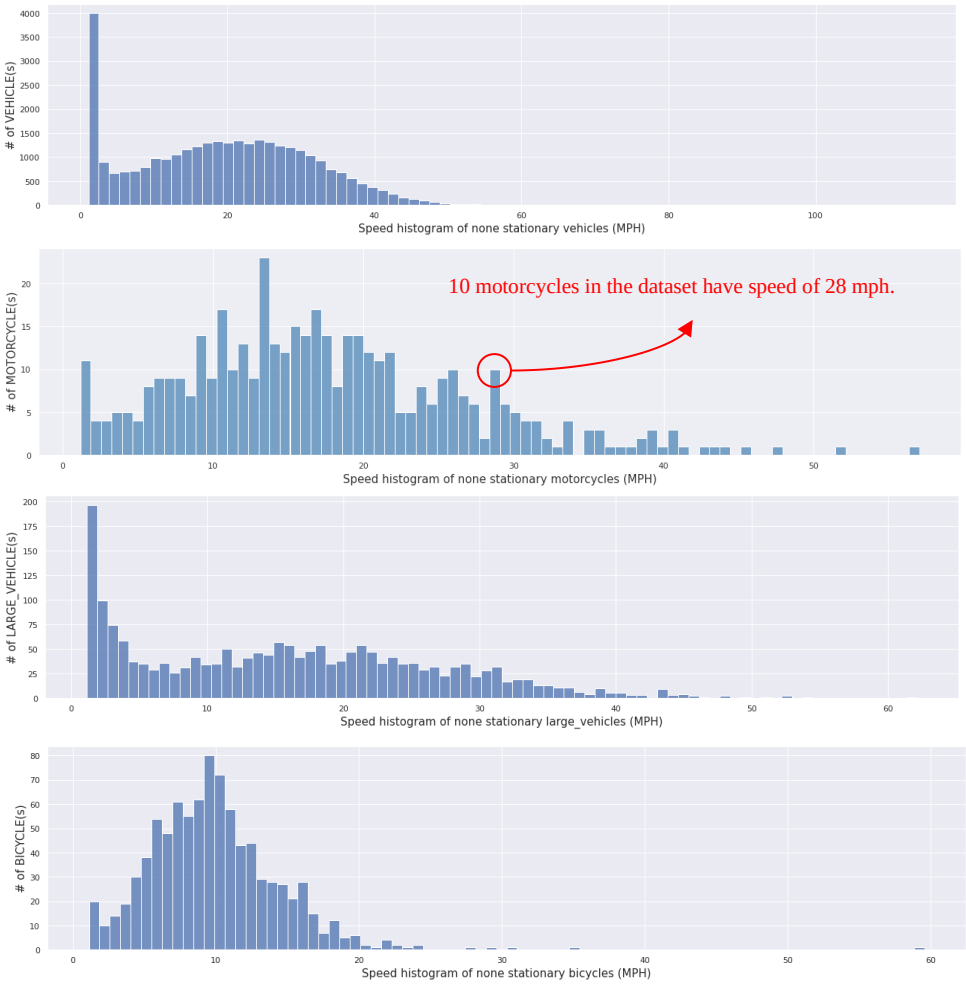


Figure 5-4: Histogram of the object's speed in the dataset

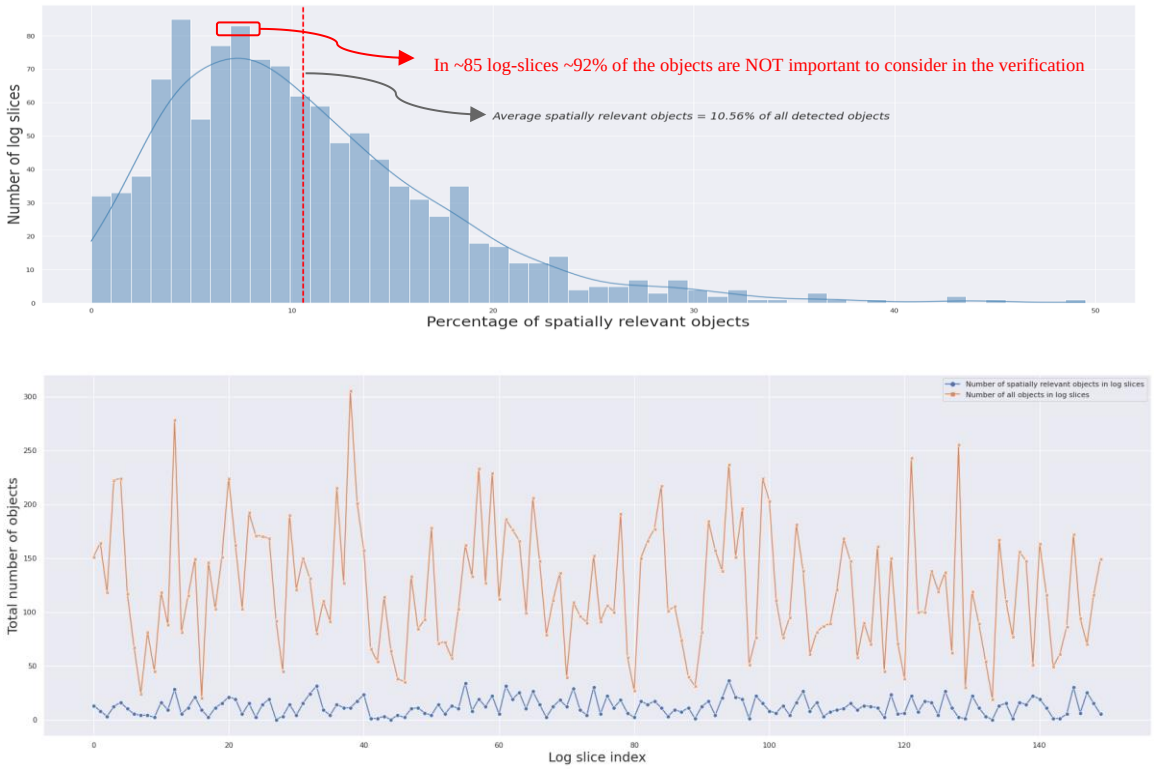
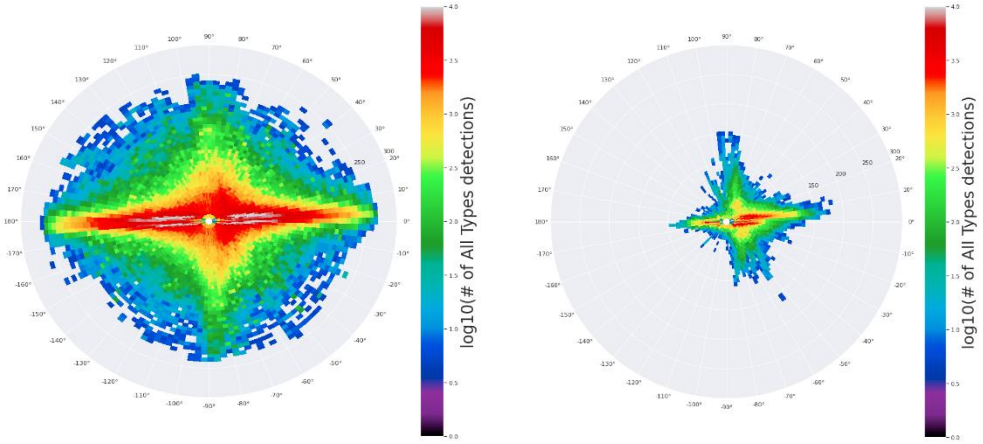
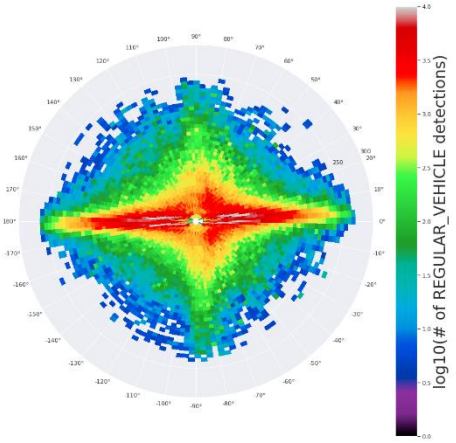


Figure 5-6: Reduction of number of objects when using spatial relevancy.

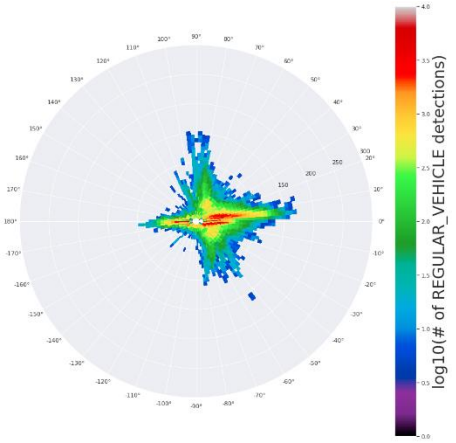


All objects before applying SRO All objects after applying SRO

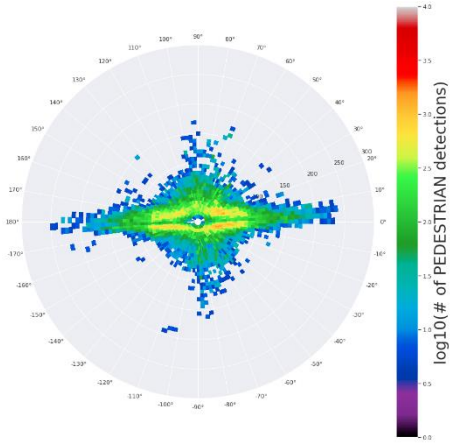
Figure 5-5: Reduction of object relative to the AV



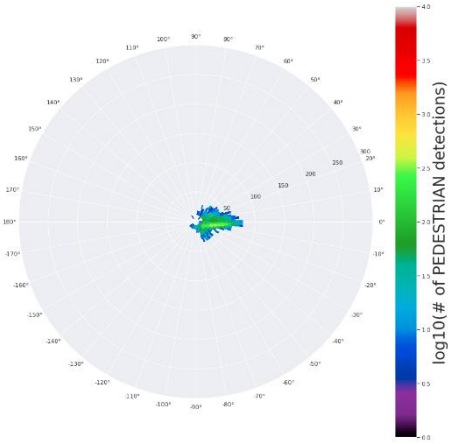
Regular Vehicles without SRO



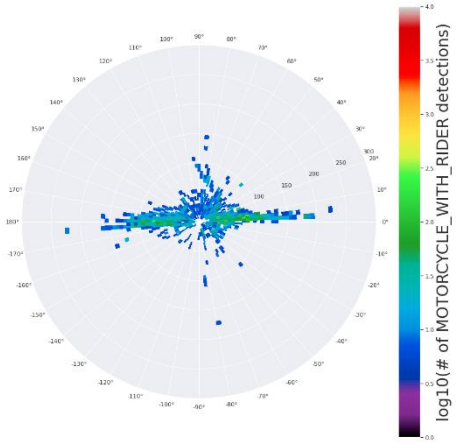
Regular Vehicles after SRO



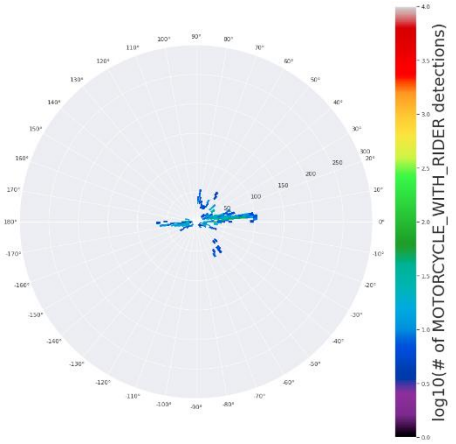
Pedestrians without SRO



Pedestrians after SRO

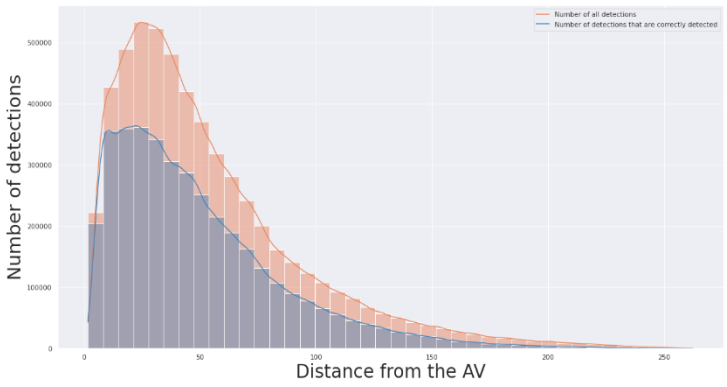


Motorcycles without SRO

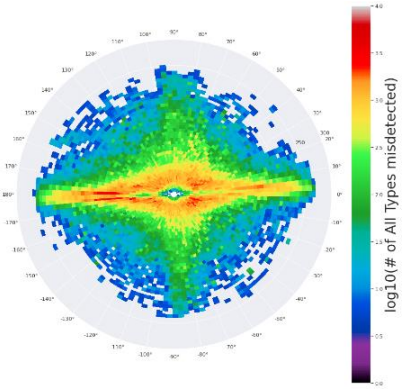


Motorcycles after SRO

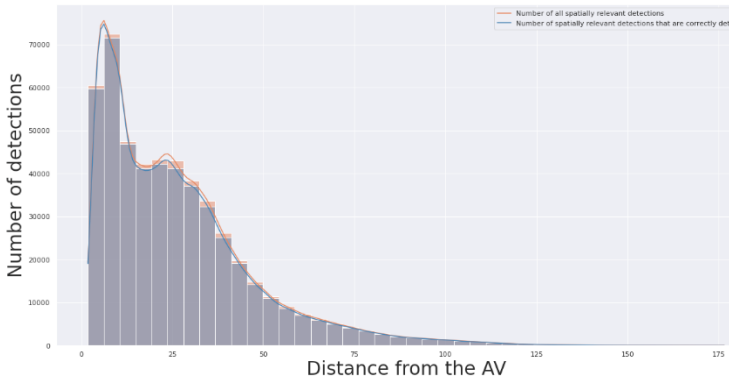
Figure 5-7: Reduction of number of objects relative to the AV (per object type)



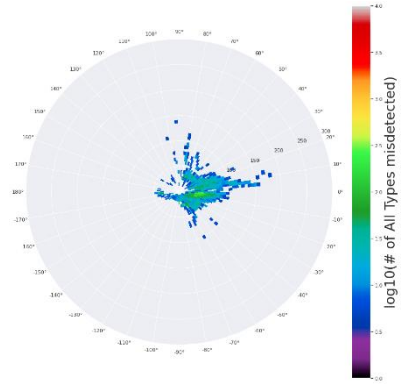
(a) Correctly detected and all objects distribution w/o SRO filter



(b) Misdetections relative to the AV w/o SRO filter

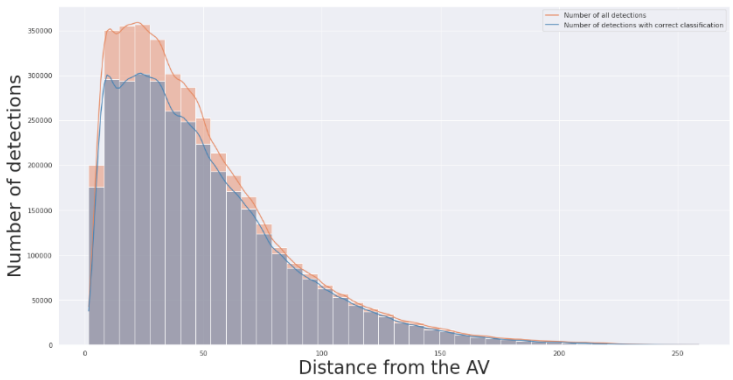


(c) Correctly detected objects and all objects distribution after applying SRO filter

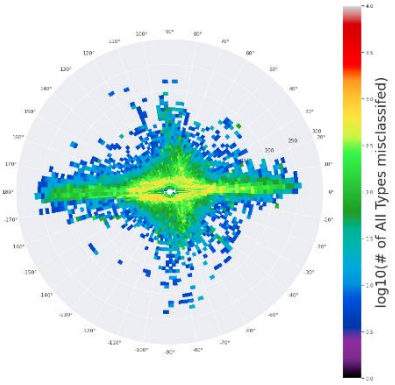


(d) Misdetections relative to the AV after applying SRO filter

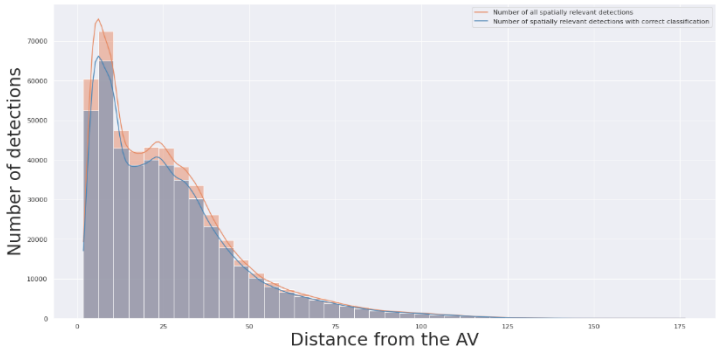
Figure 5-8: Misdetection error w. and w.o. applying SRO filter.



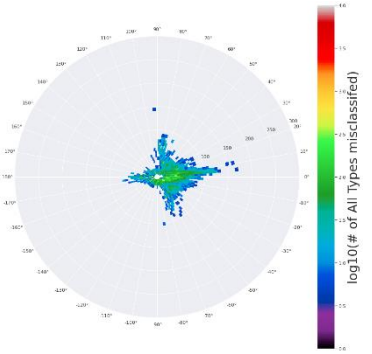
(a) Correctly classified objects distribution relative to all detections w.o. SRO filter



(b) Misclassification error relative to the AV w.o. SRO filter



(c) Correctly classified objects distribution relative to all detections after applying SRO filter



(d) Misclassification error relative to the AV after applying SRO filter

Figure 5-9: Misclassification error w. and w.o. applying SRO filter.

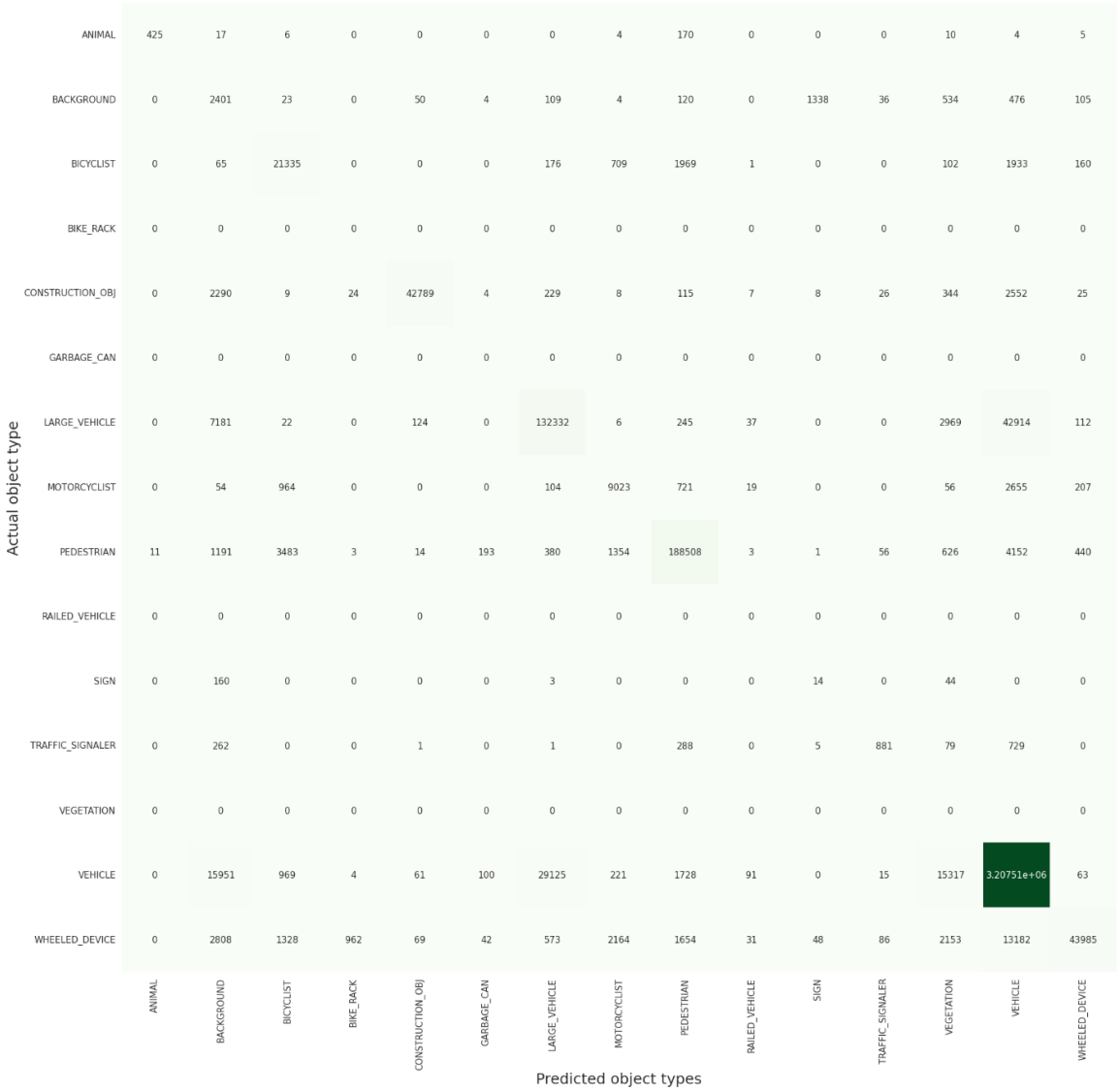


Figure 5-10: Confusion matrix without applying spatial relevancy

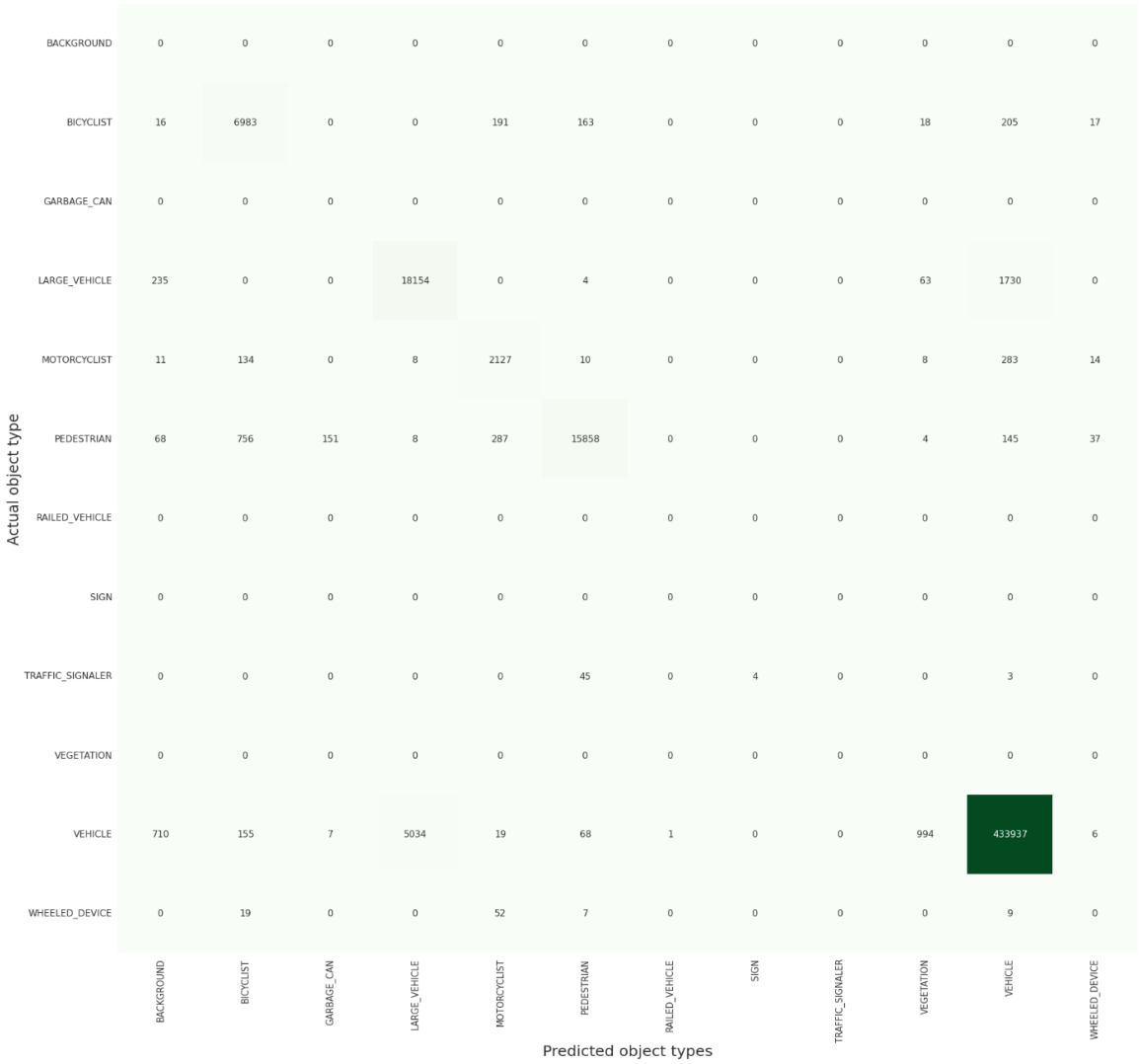


Figure 5-11: Confusion matrix After applying spatial relevancy

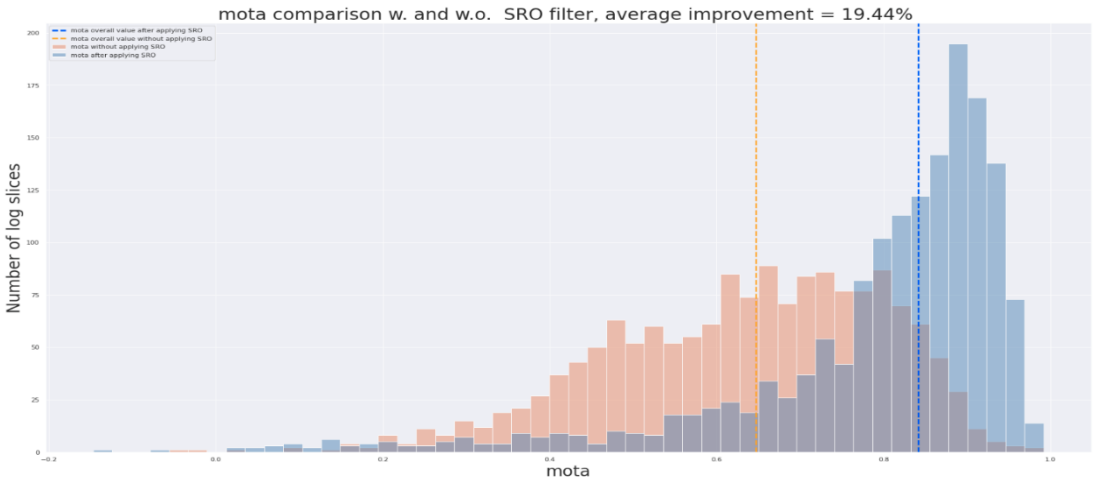


Figure 5-12: MOTA Distribution w. and w.o. applying SRO filter

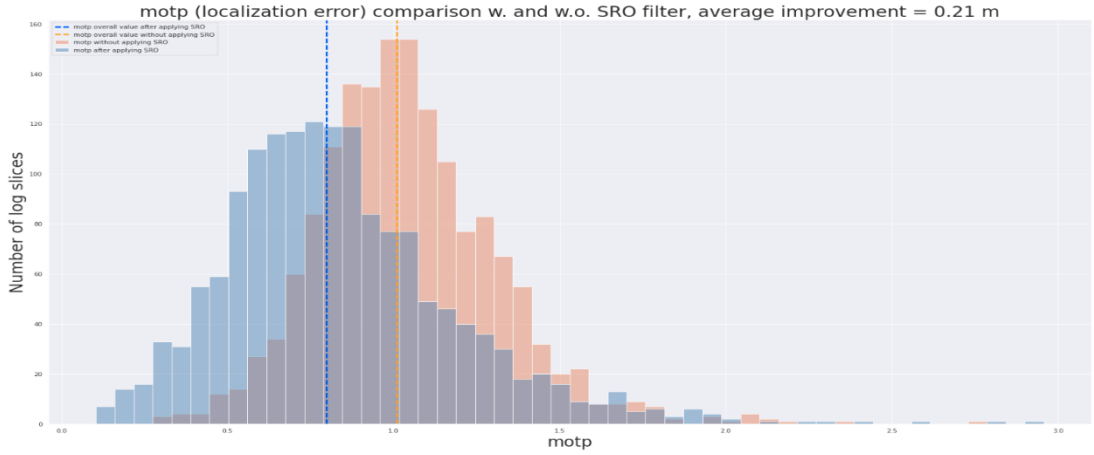


Figure 5-13: MOTP Distribution w. and w.o. applying SRO filter

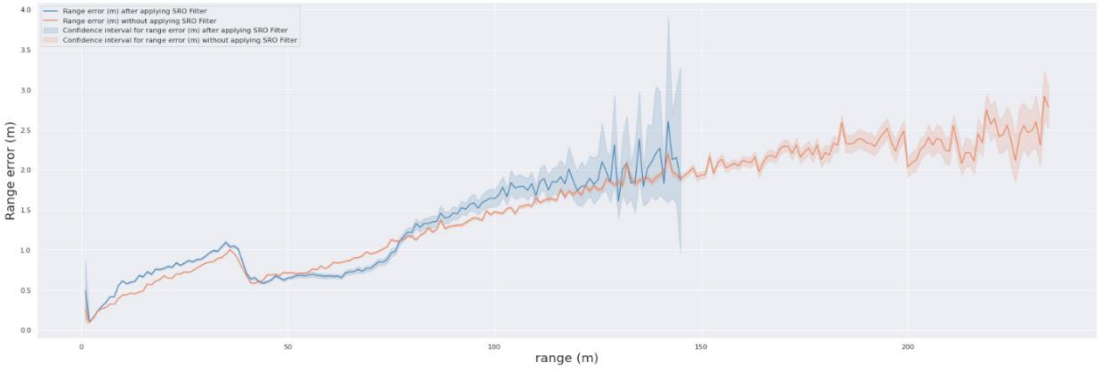


Figure 5-14: Range error relative to range w. and w.o. applying SRO filter

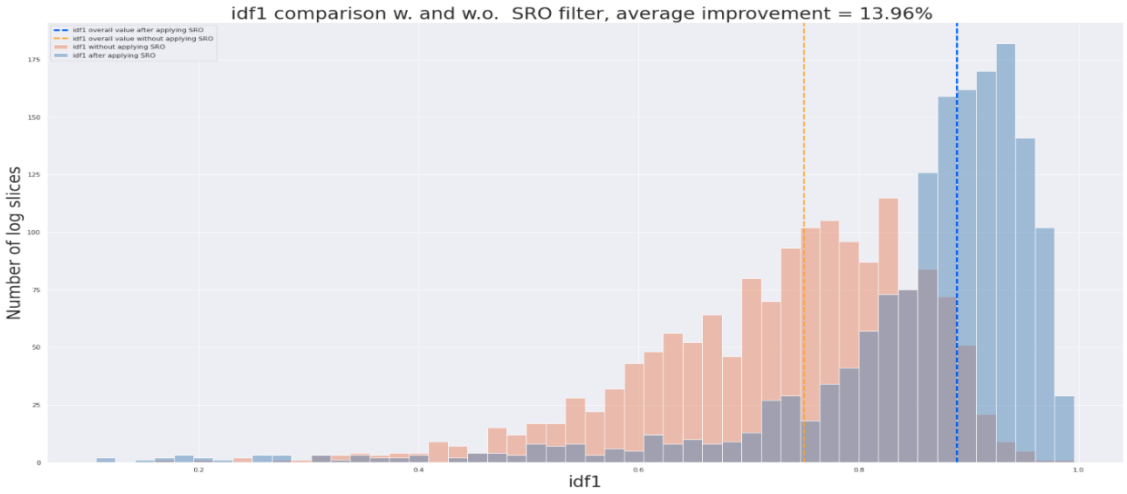


Figure 5-15: IDF1 Distribution w. and w.o. applying SRO filter

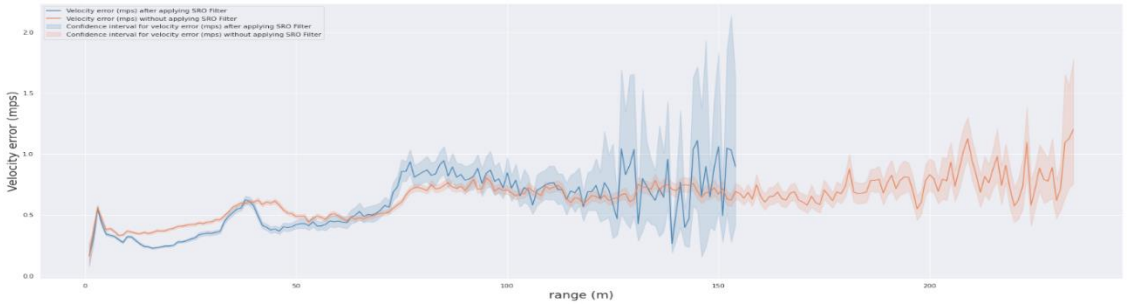


Figure 5-16: velocity error relative to range w. and w.o. applying SRO filter

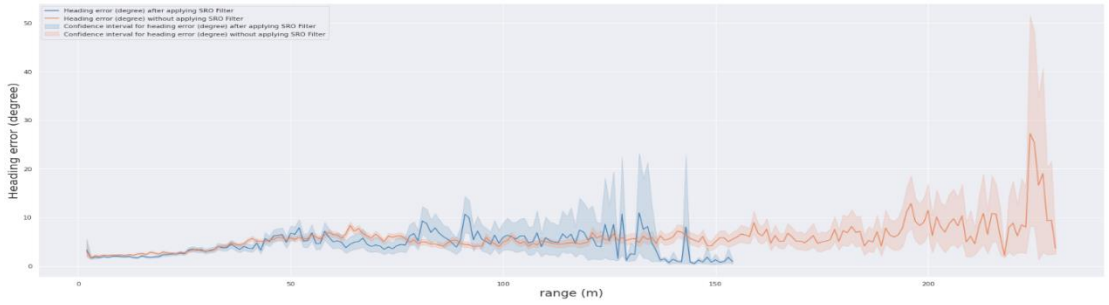


Figure 5-17: Heading error relative to range w. and w.o. applying SRO filter.

CHAPTER 6: CONTRIBUTION AND FUTURE WORK

6.1 Contribution

This dissertation presents a novel method for identifying key objects that are critical for the perception system of an autonomous vehicle to accurately detect. By focusing on these objects, our proposed system is able to speed up the verification process while still considering the potential non-compliant behavior of other objects. The main contribution of this work is the development and evaluation of an offline system that can prioritize important objects while still considering the potential risks posed by all objects in the environment, which can be used to determine a verification and training dataset for the perception system. Contributions of this work can be summarized in the following:

- Development and implementation of a framework that utilizes recorded data from actual rides of the autonomous vehicle and their corresponding labels to prioritize and rank the objects detected by the AV. This framework runs offline and enables us to efficiently identify the most important objects for the AV to accurately detect and track. Our frame includes several modules and algorithms, including:
    - Algorithms to model the autonomous vehicle's acceleration and deceleration trajectories and create isotemporal regions along the trajectory.
    - Algorithms to model the motion profile of environmental objects, taking into consideration the possibility that these objects may move in a non-compliant manner.

    - An algorithm for identifying occluded objects using a detectability score, which helps ensure that the computed metrics of the perception system are not negatively impacted.
    - A method for ranking spatially relevant objects based on relative velocity and object class, providing a more nuanced and accurate evaluation of their importance.
    - A visualization tool that can aid in debugging the algorithms and overall system performance.
    - A system architecture that includes detailed dataflows, input/output specifications, and block diagrams for a comprehensive understanding of the design and functionality.
- Evaluation of the performance of a spatial relevancy filter on a dataset collected in urban environments in Austin and Miami using autonomous vehicles, this evaluation included:
    - Evaluation method to study effect of the filter on sensing requirements and detection metrics (accuracy, precision, recall, F1 score) and on tracking metrics (CLEAR-MOT, IDF1)
    - Evidence demonstrating the effectiveness and usefulness of the spatial relevancy filter in streamlining the development process and improving the overall performance of the perception system for autonomous vehicles.
    - Evidence that our filter reduces the field of view requirements for the perception system and can be used to prioritize sensor placement when

resources are constrained, making it a valuable tool for optimizing system performance.

    - Evidence showing that classification, detection, and multi-object tracking metrics are negatively impacted when computed on the entire dataset without considering spatial relevancy, and do not accurately reflect the actual performance of the system.
    - When engineering resources are limited, our framework's severity-based ranking can be used to prioritize perception-related issues, ensuring that the most pressing concerns are addressed first.

- The following scholarly publications resulted from the research conducted for this dissertation:
    - [Patent No. 17/581,568 filed 2022-01-21] Method for Determination of Perceptual Spatial Relevancy of Objects and Road Actors for Automated Driving.
    - Maen Hammod et al., Enhancing Autonomous Vehicle Perception Validation through Relevant Object Identification [Draft Pending patent publication date ETA 2023-07-27]
    - Maen Hammod et al., Determining Spatial Relevancy of Objects For Improved Evaluation Of Multi-Object Tracking In Autonomous Driving Systems [Draft Pending patent publication date ETA 2023-07-27]
    - Maen Hammod, Guangzhi Qu, Osamah A. Rawashdeh, "Deploying and Scheduling Vision Based Advanced Driver Assistance Systems (ADAS)

on Heterogeneous Multicore Embedded Platform", The 9th International Conference on Frontier of Computer Science and Technology (FCST 2015), Dalian, China, August 26-28, 2015.

6.2 Future work

The research presented in this dissertation has established a foundation for future work in identifying key objects that the perception system of autonomous vehicles must accurately detect. The framework and data-driven evaluation methods we developed provide a solid starting point for future research and applications. Some potential areas of investigation include:

- Evaluating the performance of our system in different operating environments or objective design domains (ODDs), such as highway environments, to ensure the robustness of our approach.
- Testing the consistency of our results on different datasets from similar ODDs to confirm the validity of our findings.
- Using Monte Carlo simulation techniques to further characterize the performance of our filter by applying different levels of noise to the ground truth labels and evaluating the output of multiple multi-object trackers. This will help us understand the generalizability of our approach and its robustness to variations in tracking performance.
- Investigating why certain object classes, such as construction objects, were not retained after applying the spatial relevancy filter, even though they were present in the scene. This may be due to the fact that these

objects are static and do not intersect with the AV's planned trajectory. Further research is needed to address this issue and improve the performance of our system.

- Utilizing our framework to evaluate other frameworks that are designed to run onboard and prioritize objects in real-time.
- Adapting our framework to run online, allowing it to process and evaluate the objects detected by the perception system in real-time. Please note that our approach heavily rely on the class to model the kinematics of the object and therefore to be used onboard, the objects classifier need to have very high performance.
- Using the trajectories provided by our framework for spatially relevant objects that can move (e.g. vehicles, motorcycles, pedestrians) in simulation environments to create rare but realistic scenarios that may not be observed in the real world. Evaluating the performance of the AV in these scenarios can highlight any gaps and help to improve the overall performance of the autonomous driving system.

BIBLIOGRAPHY

[1] L. D. Burns and C. Shulgan, Autonomy: The Quest to Build the Driverless Car—And How It Will Reshape Our World.

[2] T. Reid et al., “Localization Requirements for Autonomous Vehicles | Semantic Scholar,” Localization Requirements for Autonomous Vehicles | Semantic Scholar, Jan. 01, 2020.

[3] “Vision Zero Network,” Vision Zero Network, Mar. 03, 2022.

[4] “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016B),” SAE International, Apr. 30, 2021.

[5] “SAE J3016 LEVELS OF DRIVING AUTOMATION,” LEVELS OF DRIVING AUTOMATION, Apr. 30, 2021.

[6] “Argo AI Safety Report,” Apr. 30, 2021.

[7] “Waymo Safety Report,” Feb. 01, 2021.

[8] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, “A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research | HTML,” MDPI, Feb. 05, 2019.

[9] “A Framework for Automated Driving System Testable Cases and Scenarios,” NHTSA, Sep. 01, 2018.

[10] “Definitions for Terms Related to Automated Driving Systems Reference Architecture (SEA J3131),” SEA International, Mar. 02, 2022.

[11] J. S. Albus et al., “4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems | NIST,” NIST, Aug. 22, 2002.

[12] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. 2005. doi: 10.1604/9780262201629.

[13] X. Bai et al., “TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers | Semantic Scholar,” [PDF]

[14] N. Karla and S. M. Paddock, “How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability? RAND Corporation, 2016,” JSTOR, Jan. 01, 2016.

[15] J. Luiten et al., “HOTA: A Higher Order Metric for Evaluating Multi-object Tracking | Semantic Scholar,” [PDF]

[16] Y. Xia, “Conjugate Priors for Bayesian Object Tracking,” CHALMERS RESEARCH, Jan. 01, 2020.

[17] “Multi-Object Tracking for Automotive Systems | edX,” edX. (Accessed May 02, 2022).

[18] X. Ma, W. Ouyang, A. Simonelli, and E. Ricci, “3D Object Detection from Images for Autonomous Driving: A Survey,” Semantic Scholar, Jan. 01, 2022.

[19] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” The KITTI Vision Benchmark Suite, 0 0, 2012.

[20] H. Caesar et al., “nuScenes: A Multimodal Dataset for Autonomous Driving ,” nuScenes: A Multimodal Dataset for Autonomous Driving , Jan. 01, 2020.

[21] P. Sun et al., “Scalability in Perception for Autonomous Driving: Waymo Open Dataset,” Scalability in Perception for Autonomous Driving: Waymo Open Dataset , Jan. 01, 2020

[22] G. Zamanakosa, L. Tsochatzidisa, A. Amanatiadisb, and I. Pratikakisa, “A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving,” A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving - ScienceDirect, Jul. 07, 2021.

[23] B. Xu and Z. Chen, “Multi-Level Fusion Based 3D Object Detection From Monocular Images,” CVPR 2018 Open Access Repository, Jun. 01, 2018.

[24] “Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian Tempe, Arizona March 18, 2018,” National Transportation Safety Board, Nov. 19, 2019.

[25] P. Polack, F. Altché, B. d’Andréa-Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for

autonomous vehicles?" The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?, Jan. 11, 2017.

[26] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking Objects as Points," Tracking Objects as Points, Jan. 01, 2021.

[27] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics - EURASIP Journal on Image and Video Processing," SpringerOpen, May 18, 2008

[28] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance Measures and a Data Set for Multi-target, Multi-camera Tracking ," Performance Measures and a Data Set for Multi-target, Multi-camera Tracking , Jan. 01, 2019.

[29] J. Luiten, "How to evaluate tracking with the HOTA metrics," Medium, Mar. 10, 2021.

[30] M. James, "Algorithms for the Assignment and Transportation Problems," Shibboleth Authentication Request, Mar. 01, 1957.

[31] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," On a Formal Model of Safe and Scalable Self-driving Cars, Jan. 01, 2018

[32] P. Koopman and M. Wagner, "Challenges in Autonomous Vehicle Testing and Validation," Challenges in Autonomous Vehicle Testing and Validation, Apr. 05, 2016.

[33] “The probabilistic data association filter,” The probabilistic data association filter | IEEE Journals & Magazine | IEEE Xplore.

[34] M. Gao, A. Tawari, and S. Martin, “Goal-oriented object importance estimation in on-road driving videos,” in ICRA. IEEE, 2019

[35] Z. Zhang, A. Tawari, S. Martin, and D. Crandall, “Interaction graphs for object importance estimation in on-road driving videos,” in ICRA. IEEE, 2020, pp. 8920–8927.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997

[37] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in ICRA. IEEE, 2018, pp. 4601–4607.

[38] J. Li, H. Ma, Z. Zhang, J. Li, and M. Tomizuka, “Spatio-temporal graph dual-attention network for multi-agent prediction and tracking,” IEEE Transactions on Intelligent Transportation Systems, 2021

[39] E. Aksoy, A. Yazıcı, and M. Kasap, “See, attend and brake: An attentionbased saliency map prediction model for end-to-end driving,” arXiv preprint arXiv:2002.11020, 2020.

[40] S. Martin, S. Vora, K. Yuen, and M. M. Trivedi, “Dynamics of driver’s gaze: Explorations in behavior modeling and maneuver prediction,” IEEE Trans. on Intell. Vehicles, vol. 3, no. 2, pp. 141–150, 2018

[41] V. Aravantinos and P. Schlicht, “Making the Relationship between Uncertainty Estimation and Safety Less Uncertain,” in Conf. on Design, Automation and Test in Europe, 2020.

[42] M. Hoss, M. Scholtes, and L. Eckstein. (2021) A Review of Testing Object-Based Environment Perception for Safe Automated Driving. Available at https://arxiv.org/abs/2102.08460

[43] Y. Guo, H. Caesar, O. Beijbom, J. Philion, and S. Fidler, “The efficacy of Neural Planning Metrics: A meta-analysis of PKL on nuScenes,” in IEEE/RSJ Int. Conf. on Intelligent Robots & Systems: Workshop on Benchmarking Progress in Autonomous Driving, 2020

[44] J. Philion, A. Kar, and S. Fidler, “Learning to Evaluate Perception Models Using Planner-Centric Metrics,” in IEEE Conf. on Computer Vision and Pattern Recognition, 2020.

[45] G. Volk, J. Gamerdinger, A. von Bernuth, and O. Bringmann, “A Comprehensive Safety Metric to Evaluate Perception in Autonomous Systems,” in Proc. IEEE Int. Conf. on Intelligent Transportation Systems, 2020.

[46] A. Bansal, J. Singh, M. Verucchi, M. Caccamo, and L. Sha, “Risk Ranked Recall: Collision Safety Metric for Object Detection Systems in Autonomous Vehicles,” in Mediterranean Conf. on Embedded Computing, 2021

[47] A. Bajcsy, K. Leung, E. Schmerling, and M. Pavone. (2021) Towards the Unification and Data-Driven Synthesis of Autonomous Vehicle Safety Concepts. Available at https://arxiv.org/abs/2107.14412.

[48] S. Shalev-Shwartz, S. Shammah, and A. Shashua. (2017) On a formal model of safe and scalable self-driving cars. Available at https://arxiv.org/abs/1708.06374