CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

PRED-RAG: a Predictive Radial Grid for Automotive Radar Multipath Identification

Identification of objects created by the radar multipath phenomenon, with focus on low computational complexity.

Master's thesis in Computer Science and Engineering

Andreas Karlsson
Erik Kindlund

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

PRED-RAG: a Predictive Radial Grid for Automotive Radar Multipath Identification

Identification of objects created by the radar multipath phenomenon, with focus on low computational complexity.

Andreas Karlsson
Erik Kindlund

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

PRED-RAG: a Predictive Radial Grid for Automotive Radar Multipath Identification
Identification of objects created by the radar multipath phenomenon, with focus on low computational complexity.
Andreas Karlsson
Erik Kindlund

Advisor: Niclas Carlström, Aptiv
Supervisor and Examiner: Ahmed Ali-Eldin Hassan, Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

PRED-RAG: a Predictive Radial Grid for Automotive Radar Multipath Identification
Identification of objects created by the radar multipath phenomenon, with focus on low computational complexity.
Andreas Karlsson
Erik Kindlund
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Automotive radar sensors are crucial for advanced driver assistance systems but are susceptible to the multipath phenomenon, where radio waves reflect multiple times between surfaces, creating false "ghost" objects that can trigger unnecessary safety interventions. Previous work relies on restrictive assumptions about reflection surfaces and environmental conditions, yielding solutions that perform well in specific scenarios but demonstrate limited generalization capabilities in the complex, diverse situations encountered during real-world driving. This thesis addresses the challenge of identifying radar multipath objects in real-time environments, focusing on developing an algorithm that maintains low computational complexity while achieving high accuracy. We established a development and evaluation pipeline using synthetic data together with a simulation framework, enabling data driven development of our algorithm. We propose the PRED-RAG algorithm, a novel approach that utilizes a radial grid structure combined with host motion prediction of static detections for enhanced high-level environment mapping. The algorithm identifies triplets consisting of a ghost object, reflection point and true object, then evaluates them using velocity-based criteria. When compared to a state-of-the-art algorithm, our approach demonstrates superior performance in both accuracy and computational efficiency across various driving scenarios. The PRED-RAG algorithm achieves 94.43% accuracy for high-priority objects compared to 39.26% for the baseline, with significantly better generalization capabilities, particularly in complex environments. The geometric properties employed in the grid-based approach effectively separate ghost objects from true objects while maintaining runtime performance suitable for real-time automotive applications. This work contributes to safer autonomous driving systems by reducing false objects that could lead to unnecessary emergency interventions.

Keywords: Automotive radar multipath, radial grid, host motion prediction, multi object tracking.

Acknowledgements

We would like to thank Aptiv for the support during this project and for trusting our ideas and intuitions throughout this thesis. A special thanks to our supervisor **Niclas Carlström** for all the valuable input and assistance. We would also like to thank our supervisor and examiner **Ahmed Ali-Eldin Hassan** for providing helpful feedback and insights.

Andreas Karlsson and Erik Kindlund
Gothenburg, 2025-08-05

Contents

Glossary

Ghost object Object created due to multipath detections.

Host The vehicle on which the sensor is mounted.

LiDAR Light detection and ranging sensor.

Multipath detection Detection occurring due to the multipath phenomenon regardless number of reflections, includes both type 1 and type 2 detections.

Radar Radio detection and ranging sensor.

Scan One measurement cycle of the radar.

Time of flight The time measured from when a radar wave leaves the sensor to when it comes back again.

Type 1 detection Multipath detection due to two reflections and thus DOD≠DOA.

Type 2 detection Multipath detection due to three reflection and thus DOD=DOA.

Acronyms

AEB Automatic Emergency Braking.

AES Automatic Evasive Steering.

CFAR constant false alarm rate.

DB Database.

DBSCAN density-based spatial clustering of applications with noise.

DOA direction of arrival.

DOD direction of departure.

EKF extended Kalman filter.

FCW forward collision warning.

FFT fast Fourier transform.

FMCW frequency modulated continuous wave.

FOV field of view.

IMU inertial measurement unit.

LOS line of sight.

MIMO multiple input multiple output.

NHTSA National Highway Traffic Safety Administrations.

RANSAC random sample consensus.

RCS radar cross section.

RDG radar data generator.

RMOT radar multi object tracker.

SNR signal to noise ratio.

TTC time to collision.

UKF unscented Kalman filter.

VAA virtual antenna array.

List of Figures

List of Tables

1

Introduction

In advanced driver assistance systems, or ADAS, automotive radars are widely used as one of the main sensor types as they have the advantage of being robust despite weather conditions and are relatively cheap, which makes them an appealing candidate for a fully autonomous driving system [1], [2]. Radars operate by emitting radio waves and measuring the time it takes for them to return (time of flight), which can be translated into a range between the sensor and the target [3]. The Doppler shift of the received signal, which is the change in frequency of a wave in relation to an observer who is moving relative to the source of the wave, can also be used to calculate the radial velocity, commonly called range-rate, of the wave's reflection point. In a multiple input multiple output (MIMO) radar, the direction of departure (DOD) and direction of arrival (DOA) can be estimated for each radio wave, which together with the range and range-rate resembles what is generally known as a radar detection [3]. In the best case, the radio wave has only reflected off one single point and in that case, the DOA and range resemble the true position of the detection relative to the sensor.

The multipath phenomenon occurs when the radio wave has reflected off multiple objects between the sending and receiving phase [4]. Multiple reflections will lead to an incorrect range estimation as the time of flight increases in comparison to a single reflection off one of the reflection points. If detections originating from the multipath phenomenon are not identified, and are treated as true detections when being processed by a multi-object tracking system, they will create so-called ghost objects. This is problematic since the ghost objects can propagate to safety-critical functions such as Automatic Emergency Braking (AEB) or Automatic Evasive Steering (AES) and at worst influence a decision that leads to an accident. Since the automotive driving environments are highly dynamic with many potential reflective surfaces such as guardrails, barriers, road signs and other vehicles, the multipath phenomenon is a considerable problem. Hence, an algorithm that can accurately identify such ghost objects in real-time is of high importance when developing safe ADAS systems.

Since the multipath phenomenon occurs due to radio wave reflections, if the environment is known, it is theoretically possible to identify reflective surfaces and thus trace the paths the radio wave could travel. This would be a resource-intensive task that would rely on accurate mapping of the environment, making it unsuitable for real-time applications. According to Baratam, many of the top automotive

manufacturers fail to detect and handle multipath detections [5]. Therefore there is a need for a multipath identification algorithm suitable for real-time systems.

1.1 Goals and Limitations

This work aims to develop an algorithm for radar multipath object identification with a focus on low computational complexity. Using object and detection level data, the algorithm shall identify hypothetical multipath triplets consisting of a potential ghost object, a reflection point, and a true object. To classify the hypothetical triplets, this work aims to identify correlation properties unique for objects originating due to multipath.

Autonomous driving data is highly susceptible to noise due to the dynamic environments a vehicle operates in and even with the best sensors and systems, there will always be a risk for measurement anomalies. Thus, this work is conducted under the following assumptions:

- Existing state-of-the-art radar multi object tracker (RMOT): The RMOT system used is assumed to be state of the art and without any implemented countermeasures for ghost objects.
- Optimal radar sensor: The only input noise is multipath detections.

To restrict the scope, the following limitations are applied:

- Static object distance threshold: Static objects within a close distance behind another static object are challenging to classify because of two reasons. Firstly, due to being static, their velocity properties related to multipath, described in Section 2.2.2, are all unusable due to the lack of a velocity vector, which implies there is nothing to compare. Secondly, true static objects will often be positioned close to each other but always in front of any corresponding ghost static object, seen from the sensors point of view. Hence, it can be assumed that ghost objects within a close distance behind true objects will most likely not pose any harm. Therefore, this work will consider static objects with a Euclidean distance lower than 2.0 meters behind another static object as not in scope.
- True and ghost RMOT object always present: For instances where there only exists a ghost object without a corresponding true object or vice versa, these will not be regarded or included in the performance evaluation due to the assumption of optimal RMOT.
- True and ghost RMOT object measurement updated the current scan: Detections has been associated to both objects in the current scan.
- Only multipath due to two and three reflections over at most two reflection points are covered: Higher orders of reflection count or additional reflection points imply increased complexity and can be considered highly unlikely, as the signal attenuates with each reflection [6].

Moreover, as there is a varying degree of danger a ghost object can impose, a priority scheme is defined, described in Section 5.3. The priority scheme aims to introduce a safety aspect to the algorithmic performance evaluation, where the highest priority class will be used for optimization, but performance metrics will be presented for all classes.

1.2 Ethical Considerations

The algorithm developed in this work performs a classification of RMOT objects as either true or false (ghost). In safety-critical automotive applications, it is essential to carefully balance the false positive rate and false negative rate, as both types of errors can lead to accidents. Misclassifying a true object as a ghost could prevent the AEB system from activating when needed, potentially causing a collision. Conversely, failing to identify a ghost object (classifying it as true) could trigger unnecessary AEB activation, which may lead to dangerous traffic situations. Therefore, optimizing this classification balance is crucial for the safe implementation of the system in real-world traffic scenarios.

1.3 Thesis Outline

The thesis is structured as follows: Chapter 2 introduces necessary radar basic knowledge to be able to understand the underlying causes of the multipath phenomenon, including the mathematical equations required to model multipath propagation. Further, Chapter 2 introduces basic RMOT concepts required to understand how the synthetic detections are used to create objects which the PRED-RAG algorithm operates on. Lastly, Chapter 2 explains the difficulty with multipath labeling in conventional datasets together with a short introduction of spatial data structures.

Chapter 3 sets the stage of what has previously been done on the topic both in the area of deterministic solutions and machine learning solutions.

Chapter 4 first explains the synthetic data generation together with the simulation framework utilized in this work. Thereafter, Chapter 4 thoroughly explains the PRED-RAG algorithm and state-of-the-art baseline algorithm.

Chapter 5 briefly introduces how the evaluation scenarios were constructed and the key properties for set 1 and set 2. Chapter 5 thereafter explains the ground truth mapping, priority scheme and the performance metrics evaluated. In Chapter 6, there is a short comparison between the baseline and PRED-RAG algorithm with the two following sections presenting the results more in depth. Finally, Chapter 7 discusses the results together with what might be improved in future work and presents some final conclusions.

In the Appendix, a variety of figures and tables have been placed; these are items that might be useful for anyone trying to further develop this work but were deemed not necessary enough to be part of the main thesis.

2

Background

2.1 Radar Basics

Automotive radar is one of the most established sensors in ADAS systems and the most common type of automotive radar used today is the MIMO frequency modulated continuous wave (FMCW) radar [2]. The general process flow of a MIMO FMCW radar can be seen in Figure 2.1.

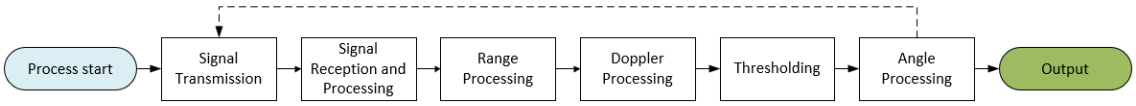


Figure 2.1: Process flow of an FMCW automotive radar.

2.1.1 Signal Transmission

The first step of the radar process is generating a frequency-modulated continuous wave, or chirp, signal. Such a signal can be seen in Figure 2.2 as the green line. The signal is shown as frequency over time where f_0 is the base transmission frequency, B is the bandwidth, T is the chirp duration, τ is the time delay between the transmitted and received signal, and f_β is the frequency difference between the transmitted and received signal. In a MIMO system, the signal also incorporates an additional frequency shift, unique for each transmitting antenna element [7], [3]. These unique shifts enable the receiving antennas to distinguish from which transmitting antenna the signal has originated, creating a virtual antenna array (VAA) which has the size of transmitting antenna elements times receiving antenna elements [8]. This in turn increases the resolution of the radar which makes it possible to determine the DOA of the signal more accurately [8]. The VAA is constantly active during a fixed period of sensing time, called a dwell, where the generated signal is continuously being sent out and received.

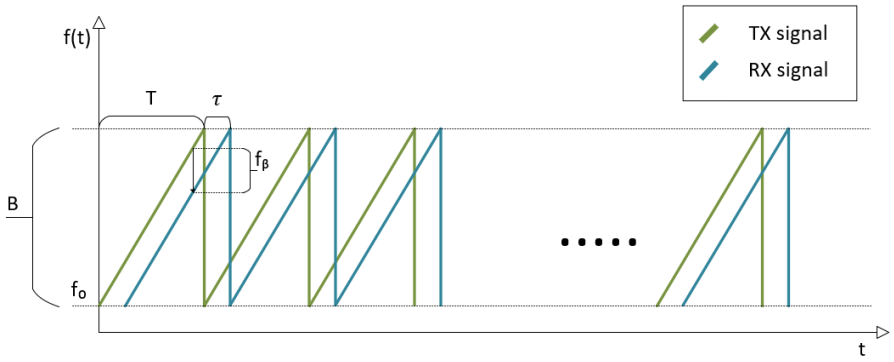


Figure 2.2: The basic shape of a transmitted and received FMCW radar signal.

2.1.2 Signal Reception and Processing

After the transmitted signal has propagated through the environment and reflected off different surfaces, the reflected signals will be received by the MIMO antenna array and passed on for further processing. To make it easier to work with the received signal, the bandwidth is reduced by mixing the received signal with the transmitted one, bringing the received signal down to base-band frequency [9]. The resulting signal is the frequency shift f_β. This allows the system to sample the signal without having to use high frequency sampling hardware which is considerably more expensive. After mixing the signal it is passed through a low pass filter to remove system noise and an amplifier where the filtered signal is amplified before being sampled by an analog to digital converter [9], [10].

2.1.3 Range Processing

When the signal has been sampled, it is processed to extract the peaks in the range spectrum. The range R is equal to the one-way distance traveled by the wave which can be computed as follows:

$$R = \frac{c\tau}{2} \tag{2.1}$$

where c is the speed of light. The time delay τ can, in turn, be approximated in terms of the frequency difference f_β as:

$$\tau = -\frac{f_\beta T}{B} \tag{2.2}$$

Inserting Equation 2.2 into 2.1 gives the following equation for the range R in terms of the sampled signal frequency shift f_β:

$$R = -\frac{c f_\beta T}{2B} \tag{2.3}$$

The range values of the signal peaks can then be extracted by performing a fast Fourier transform (FFT) on the sampled signal and converting the frequency axis into range using the equations above.

2.1.4 Doppler Processing

As mentioned before, the relative radial velocity of a radar detection is derived from the Doppler shift of the received signal. In Equation 2.3, the range is calculated using the frequency difference f_β. This calculation is correct based on the fact that the FFT produces peaks in the signal frequency regardless of phase shifts. In reality, f_β is a composition of the frequency f_R and the doppler shift f_D as:

$$f_\beta = f_R - f_D \tag{2.4}$$

Here, the frequency f_R is the only component dependent on the range of the reflection point according to the equation:

$$f_R = -\frac{B2R}{Tc} \tag{2.5}$$

The doppler shift f_D on the other hand is dependent on the radial velocity v_r of the reflection point relative to the sensor:

$$f_D = \frac{2v_r}{R} \tag{2.6}$$

The continuous ideal base-band signal can then be described as:

$$s(t) = e^{j2\pi(f_\beta t+\phi)} \tag{2.7}$$

where ϕ is the relative radial velocity-dependent phase shift.

To be able to calculate the range-rate, an FFT is performed across all chirps in the dwell, where the frequency axis is converted to radial velocity according to Equation 2.6. The output from the range FFT and Doppler FFT produces a 2-dimensional matrix called a range-doppler map, where the y-axis consists of range bins and the x-axis of range-rate bins. Each square, or cell, in the range-doppler map, has the size range resolution times range-rate resolution [9], [10]. One range-doppler map is created for each receiving antenna element in the VAA.

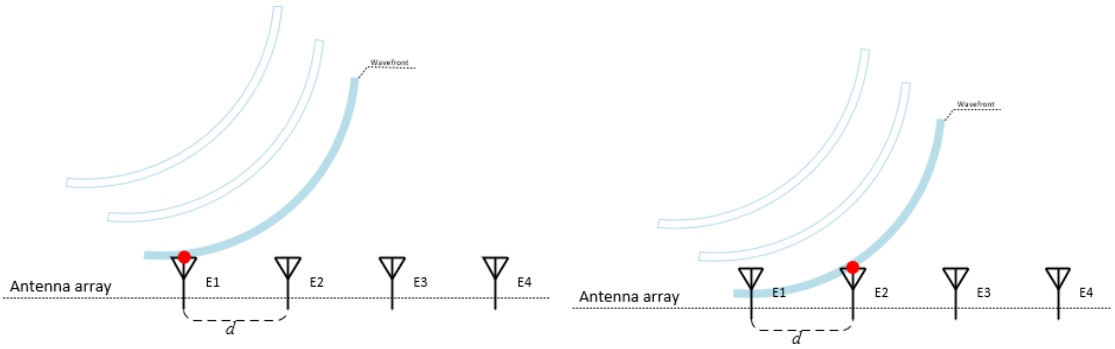
2.1.5 Thresholding

When the range-doppler maps have been created, thresholding is typically performed to select which of the cells should produce detections in the output. One of the most common thresholding methods is the cell-averaging constant false alarm rate (CFAR). Cell-averaging CFAR is performed in radar signal processing to avoid overloading the system with false alarms (creating detections from noise) under the influence of clutter [11], [12]. The cell-averaging also has another effect on the output which is limiting the number of detections produced from within a certain region in range-doppler space, usually within a range bin or range-doppler cell. This effect allows modern radars to run with high-resolution configurations without filling up the output detection buffer only using data from the closest targets.

The CFAR algorithm determines from which cells in the range-doppler map detections should be produced based on a statistical method. The job of the algorithm is to maintain a constant probability of false alarms despite the levels of interference continuously changing [10]. The cells selected are the same for all range-doppler maps across the VAA.

2.1.6 Angle Processing

After a set of cells has been selected by the CFAR algorithm, the cells are passed on to an angle estimation algorithm. A multitude of algorithms exists for estimating the angle of a radar detection, including established methods such as MUSIC and ESPRIT [13], [14], [15]. Even though the algorithms are different, the basic principle remains the same. When a radar wave is received by the VAA, the arrival angle of the wave will influence the phase shift induced in each antenna element since the wavefront will arrive at each element at a slightly different time. An illustration of a wavefront arriving at a four-element antenna array can be seen in Figure 2.3.



(a) State at time T. (b) State at time $T + \Delta T$.

Figure 2.3: Illustration of wavefront arriving at a four-element antenna array.

As seen in the figure the wavefront will in this case arrive at antenna element E1 first, then E2, E3 and finally E4. Since the distance d between adjacent antenna elements is small, the resulting time difference ΔT between signal arrivals is significantly smaller than the range-dependent time difference τ. Hence, the time difference ΔT will only influence the phase shift of the signal between the antenna elements. The angle-dependent phase shift peaks can then be extracted by performing an FFT for each cell across all antenna elements, or by using a more advanced algorithm [16], [13]. When the angle-dependent phase shift has been determined the angle of the detection can be calculated.

2.1.7 Radar Output

When the angles have been computed for the set of selected cells, a list of detections including the computed range, range-rate and angle, is constructed. Each cell can give rise to multiple detections, therefore the number of output detections might be higher than the number of cells in the range-doppler map.

2.2 Multipath Radar Data

As previously mentioned the multipath phenomenon occurs due to multiple reflections during a radio wave's propagation through the environment. In automotive radar

systems, the phenomenon is commonly divided into two different types, one where the DOD is different from the DOA and one where they are equal [17], [18], [19]. Here, the case where DOD≠DOA is referenced as a type 1 detection and the case where DOD=DOA: a type 2 detection. Figure 2.4 illustrates a sample multipath scenario along with the considered multipath effects. The radio wave emitted from sensor S propagates through the environment, reflecting off the target T and guardrail R in different ways before returning to the sensor. The propagation paths that have created each type of detection can be seen in Table 2.1.

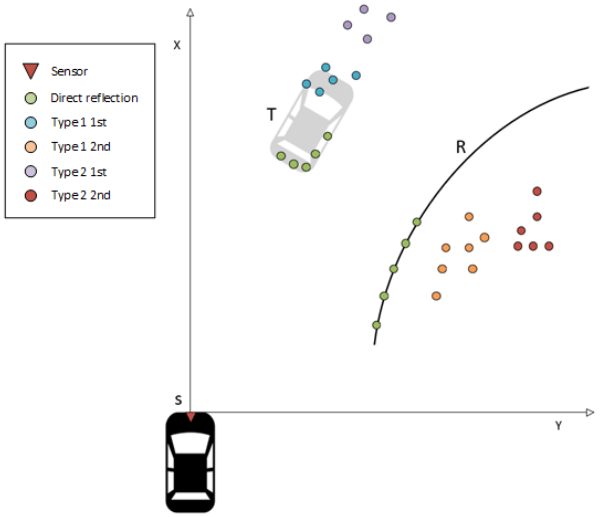


Figure 2.4: Illustration of produced radar detections in a common multipath scenario (not to scale).

Table 2.1: Propagation paths and corresponding detection type.

Propagation Path	**Detection Type**
s→T→s	Direct reflection
s→R→s	Direct reflection
s→R→T→s	Type 1 1st
s→T→R→s	Type 1 2nd
s→T→R→T→s	Type 2 1st
s→R→T→R→s	Type 2 2nd

2.2.1 Geometric Properties

In Figure 2.4 it can be seen that given two reflective surfaces in the radar field of view (FOV), here embodied as a car T and a guardrail R, the resulting detection output includes 4 different types of multipath. The first and second-order type 1 and type 2 respectively are very similar in their geometric correlation, the only difference being the inverted order of reflections on the propagation path. In Figure 2.5 a single case of each multipath detection type is shown.

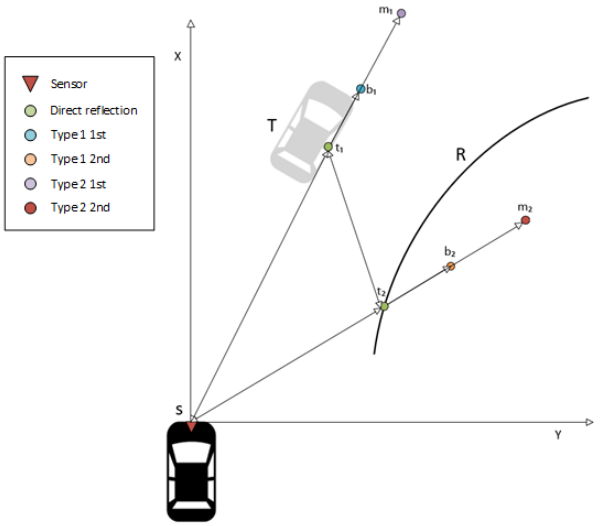


Figure 2.5: Geometric relation of each multipath detection type (not to scale).

From the figure, it can be seen that the vector from the sensor to the multipath detection has the same direction as the vector from the sensor to the last reflection point of the multipath detections propagation path. Since the range of a detection r is calculated as $r = \frac{ct}{2}$, from Equation 2.1, the vectors $\overrightarrow{sb_1}$, $\overrightarrow{sb_2}$, $\overrightarrow{sm_1}$ and $\overrightarrow{sm_2}$ can be computed given the vectors $\overrightarrow{st_1}$, $\overrightarrow{st_2}$ and $\overrightarrow{t_1t_2}$ as follows [10]:

$$\begin{aligned}
\overrightarrow{sb_1} &= \frac{||\overrightarrow{st_1}|| + ||\overrightarrow{st_2}|| + ||\overrightarrow{t_1t_2}||}{2} \cdot \frac{\overrightarrow{st_1}}{||\overrightarrow{st_1}||} \\
\overrightarrow{sb_2} &= \frac{||\overrightarrow{st_1}|| + ||\overrightarrow{st_2}|| + ||\overrightarrow{t_1t_2}||}{2} \cdot \frac{\overrightarrow{st_2}}{||\overrightarrow{st_2}||}
\end{aligned} \tag{2.8}$$

$$\begin{aligned}
\overrightarrow{sm_1} &= (||\overrightarrow{st_1}|| + ||\overrightarrow{t_1t_2}||) \cdot \frac{\overrightarrow{st_1}}{||\overrightarrow{st_1}||} \\
\overrightarrow{sm_2} &= (||\overrightarrow{st_2}|| + ||\overrightarrow{t_1t_2}||) \cdot \frac{\overrightarrow{st_2}}{||\overrightarrow{st_2}||}
\end{aligned} \tag{2.9}$$

2.2.2 Radial Velocity Properties

The radial velocity of a detection is derived from the Doppler shift of the received radio wave. The Doppler shift of the signal varies with each reflection along the propagation path, depending on the velocity of each reflecting surface [10]. Given a radio wave with the wavelength λ, the Doppler shift f_D after a reflection can be calculated as:

$$f_D = \frac{2||v_t||}{\lambda} cos(\psi) cos(\frac{\beta}{2}) \tag{2.10}$$

where v_t is the relative velocity of the reflection point compared to the sensor, ψ is the angle between the velocity vector of the reflection point and the bisector of the angle β, and β is the angle subtended between the incoming direction of the wave to the reflection point, the reflection point itself, and the outgoing direction of the wave from the reflection point [20]. For a direct reflection detection, the β angle is zero, and ψ is the angle between the velocity of the reflection point and the angle of the reflection point location relative to the sensor, leaving the equation:

$$f_D = \frac{2||v_t||}{\lambda} cos(\psi) \tag{2.11}$$

For multipath detections however, the Doppler shift is affected by multiple reflections resulting in a more complex relation to the radial velocity of the detection. Given each reflection point i, Equation 2.10 can be extended to handle multiple reflections:

$$f_D = \sum_{i=1}^{k} \frac{2||v_{ti}||}{\lambda} cos(\psi_i) cos(\frac{\beta_i}{2}) \tag{2.12}$$

From this equation, the range-rate $\dot{r}$ can be computed as:

$$v = \frac{f\lambda}{2} \Rightarrow \dot{r} = \sum_{i=1}^{k} ||v_{ti}|| cos(\psi_i) cos(\frac{\beta_i}{2}) \tag{2.13}$$

An illustration of the angles and velocity vectors can be seen in Figure 2.6 where the point S is the sensor, P_1 is the first reflection point, and P_2 the second.

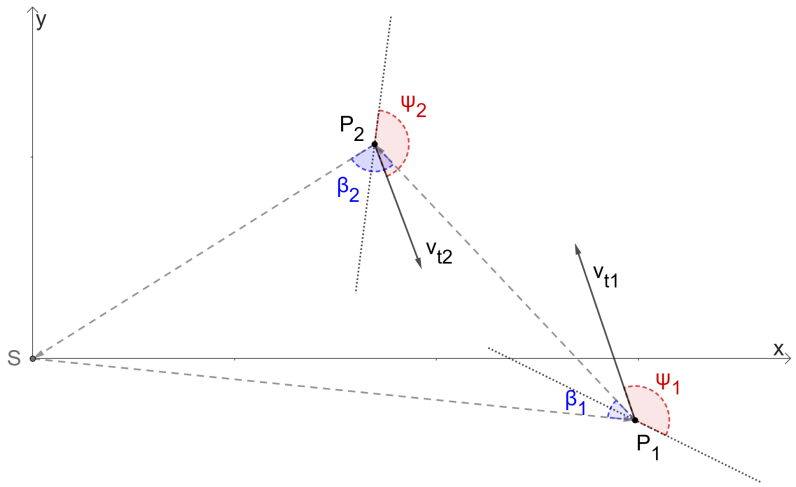


Figure 2.6: Illustration of relevant multipath velocity properties in a type 1 scenario.

In a type 2 scenario, the angle β_2 is zero and the point P_1 is used twice in the equation ($\mathrm{P}_3 = \mathrm{P}_1$) as the wave reflects off it two times. The geometry of a type 2 scenario is illustrated in Figure 2.7.

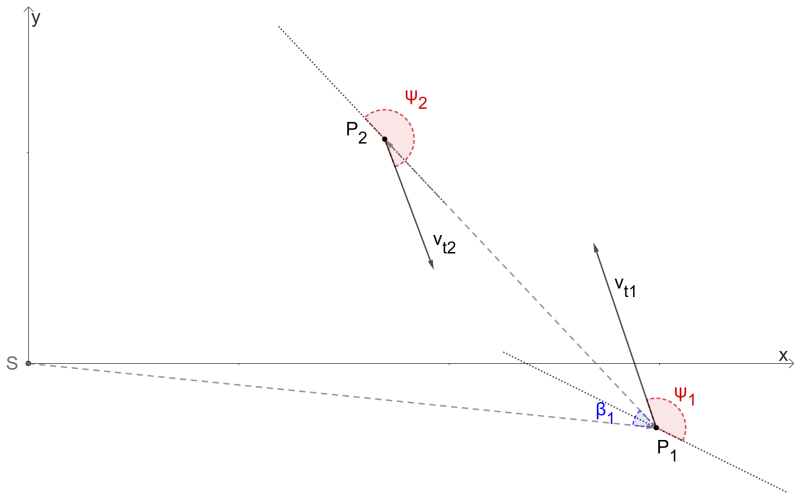


Figure 2.7: Illustration of relevant multipath velocity properties in a type 2 scenario.

Additionally, as the range-rate of a target is the relative change of range between the sensor and the target, the range-rate of a ghost object can also be calculated as the derivative of its range. For the type 1 scenario, the range-rate $\dot{r}$ can be found by differentiating Equation 2.8:

$$\dot{r} = \frac{1}{2}\left(\frac{(S-P_1)(v_s - v_{p1})}{\|S-P_1\|} + \frac{(P_1-P_2)(v_{p1}-v_{p2})}{\|P_1-P_2\|} + \frac{(P_2-S)(v_{p2}-v_s)}{\|P_2-S\|}\right) \tag{2.14}$$

and for the type 2 scenario, it can be found be differentiating Equation 2.9:

$$\dot{r} = \frac{(S-P_1)(v_s - v_{p1})}{\|S-P_1\|} + \frac{(P_1-P_2)(v_{p1}-v_{p2})}{\|P_1-P_2\|} \tag{2.15}$$

Here v_s is the velocity of the sensor, v_{p1} the absolute velocity of point P_1, and v_{p2} the absolute velocity of point P_2.

Even though the radar sensor can measure the Doppler shift of the received signals, the equations above show that to utilize these relations it is necessary to know the velocity vector of each reflection point. The velocity vector is generally not something that can be directly measured by a radar but it is possible to estimate it over time using a multi-object tracking system [21], [22].

2.3 Radar Multi-Object Tracking

Radar multi-object tracking is the process of clustering and filtering radar detections over time to remove measurement noise and enable estimation of properties that are not measurable directly by the radar [21], [22]. It is also one of the most common

processes used in the automotive industry to perceive the surrounding environment using radar data. The general process flow of the RMOT system can be described as seen in Figure 2.8. The input to an RMOT system is a list of radar detections from the latest available scan and host motion data such as velocity, gyro measurements, and steering angle. Over time, the data in the RMOT system will accumulate confidence while moving through the cycles and after a while form tracked objects. This time flow can be seen in Figure 2.9. RMOT systems commonly incorporate some type of Bayesian filter to track the objects. Filters commonly used for this purpose are for example the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [23].

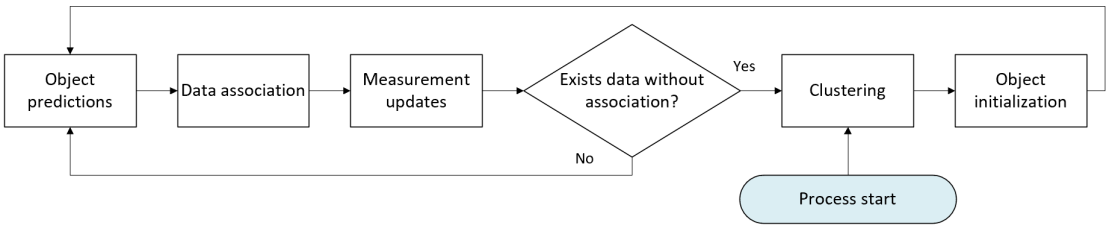


Figure 2.8: Radar multi-object tracker process flow.

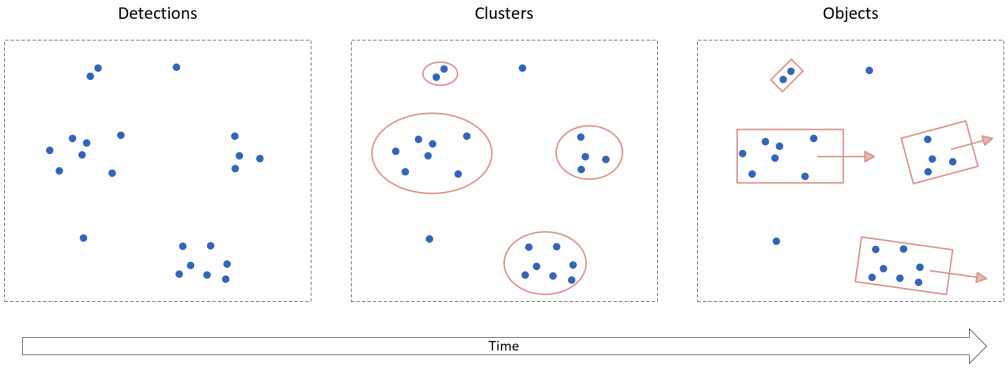


Figure 2.9: States of tracked data in an RMOT system.

2.3.1 Clustering

The entry point to the process loop is the clustering step, since on the first iteration there will be no historically tracked objects and therefore all input radar data will be unassociated. The clustering algorithm groups the radar detections based on properties like position and range-rate, trying to create one group of detections per actual target object present in the data, and single out any possible noise. One popular algorithm used for such clustering is the density-based spatial clustering of applications with noise (DBSCAN) [24], [25].

2.3.2 Object Initialization

After clusters of the input detections have been formed, the next step in the cycle is to initialize tracked objects from these clusters. All active clusters are evaluated to see if they match the properties of a possible real target. If a cluster is deemed to be likely to have come from a real target, a tracked object will be formed from the cluster and stored. A tracked object is usually represented by a filter motion model along with any potential other parameters which is preferably estimated over time [23]. The clusters that are not likely enough to have come from a real target could either be scrapped or stored to also be tracked over time as clusters. With the latter approach, a cluster can accumulate data over time, enabling a more confident decision about its handling in the subsequent processing cycle.

2.3.3 Object Predictions

Even if the entry point to the process loop on the first iteration is the clustering step, the actual first step on the loop in each forthcoming cycle is the object predictions. Commonly in this step, assuming the RMOT system utilizes a Bayesian filter, all currently tracked objects are brought into the current time by performing a prediction using the motion model of each object [26]. These predictions are done by taking the last known state of each object, which will be from the last cycle, and computing an estimate of how the state most likely has changed between the last time the state was updated and the current time. The change will then be applied to the state, bringing it forward in time to match the upcoming input radar measurement data. A visualization of the timeline can be seen in Figure 2.10.

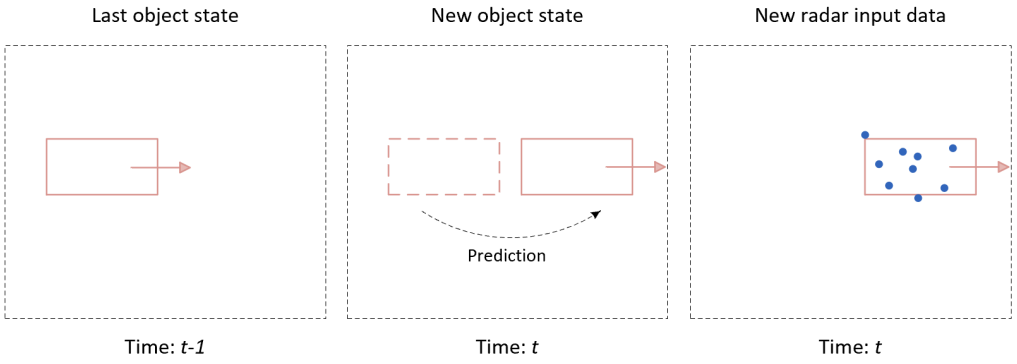


Figure 2.10: Timeline perspective of object prediction step.

2.3.4 Data Association and Measurement Updates

When the tracked objects have been brought in to current time, object states are predicted to the timeframe of the measurement enabling association of the radar detections to the objects [26]. All new radar detections are compared to the tracked objects and if a detection matches the properties of the tracked object in regard to position and range-rate, the detection will be associated to that object. After the detections have been associated with the tracked object's new predicted states, these

detections will be used to update the object states. The measurement update adjusts the predicted state according to the chosen measurement model, which estimates a more accurate approximation of the true object states whilst still being resistant to noise in the measurements [26].

2.4 Automotive Radar Datasets

Several well-established datasets have emerged in response to the rapid development of autonomous driving systems, including ZOD [27], nuScenes [28], and KITTI [29]. Most of the automotive datasets contain multi-modal sensor data from the most utilized sensor types, camera, LiDAR, radar and inertial measurement unit (IMU) with labels for some of the most common machine learning tasks such as 2D and 3D object detection, semantic segmentation, object tracking and lane detection [30], [1]. The labeling is often done manually or semi-automated to provide a high-level class depending on the task. For the case of object detection, bounding boxes with properties and class tags are provided for all available objects in the scene, and LiDAR is used as the reference sensor to ensure accurate metrics.

2.4.1 Multipath Labeling

To our knowledge, no public dataset currently exists with multipath annotations. Annotating radar detections for multipath is inherently hard due to the complex nature of the multipath phenomenon. Multipath detections may occur within the bounding box of a true object, and thus using true object bounding boxes as a ground truth would be inaccurate. Furthermore, multipath detections may be too few for some execution cycles to pass object initialization, thus those detections would be unclassified if object-level data is used as ground truth, leading to uncertainties in the data. As illustrated by Figure 2.11, the radar, which is usually mounted lower than the LiDAR, will have its FOV occluded by the lane barrier and thus no true detections from the object to the right will be created. However, due to multipath, the radar may produce multipath detections at roughly the same position as the object to the right. If the bounding boxes from the LiDAR data are used as reference true objects with the rationale that all detections within a true bounding box are true detections, this would lead to multipath detections being incorrectly classified as true. Further, there may be multipath detections that appear within the bounding box of the true object to the left, and the same applies here: multipath detection would be classified as true detections, leading to inconsistencies in the data labels.

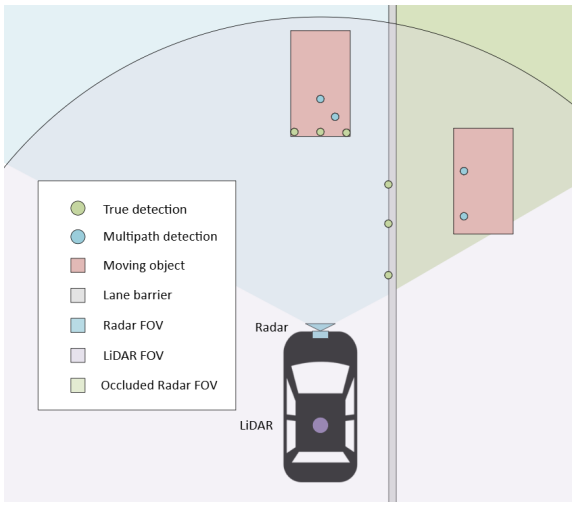


Figure 2.11: Radar and LiDAR FOV differences due to mounting position. Illustration of how a true object may cause a multipath detection to be classified as a true detection.

Radar multipath could theoretically be labeled by modeling the multipath propagation paths. In that case, the surrounding reflective surfaces must be known, which implies that the surrounding environment must be precisely mapped, and preferably the material properties should also be known. From what we know, this would be a demanding task and has not yet been done.

Since radar is prone to noise and sensitive to more than the multipath phenomenon, such as angle ambiguities and range-rate ambiguities, labeling radar detections manually is error-prone since there would exist multiple cases where the human eye cannot distinguish which phenomenon was the true cause of the erroneous detection [10].

2.5 Spatial Data Structures

For applications with positionally related data entities, spatial data structures organize these entities in a logical manner that preserves their positional relationships while enabling low-cost computational queries for entities meeting specific constraints [31]. One set of common spatial data structures are grids, they can be based on either Cartesian or Spherical coordinates in both two and three dimensions. Grids work by the principle of splitting up the surrounding area into bins, where each bin covers a certain area. Each entity is assigned the bin that corresponds to its location [32]. This implies that all entities that are within a certain area will be allocated to the same bin, which allows for a simple bin query. Compared to naive approaches, which would imply querying all entities and checking if they satisfy the condition, this reduces the computational complexity of querying entities by position from $\mathcal{O}(n)$ to $\mathcal{O}(1)$. In real-time applications where computations like these are performed, spatial data structures are crucial for meeting run-time constraints.

3

Related Work

3.1 Deterministic Solutions

Using deterministic algorithms is the most common way of mitigating multipath issues in automotive radar data today. One of the most popular techniques is filtering the radar data on signal processing level based on the radar waves DOD and DOA [33], [34]. By checking if the DOA angle is different from the DOD angle when a radar wave returns to the radar, the detection can directly be classified as a type 1 multipath detection. The problem with this technique, however, is that the type 2 multipath detections, where the DOA angle is still equal to the DOD angle, cannot be separated from regular true targets. To be able to handle type 2 multipath detections as well, different algorithms have been developed which make use of the geometric properties of the multipath scenario to identify the false targets [17], [35], [36], [19]. The general structure of these algorithms is to produce candidate pairs of either detections or objects, one hypothesized as true and the other as a false target caused by multipath. After producing such pairs, all pairs are evaluated against some criteria coupled to their spatial relation to each other, the host vehicle, and in some cases the surrounding environment, to decide whether or not to classify a detection or object as caused by multipath.

The algorithm described by Liu et al. [17], filters out detections based on their range difference between each other. A detection is classified as multipath if it is within a threshold further away from the sensor than its closest neighboring detection towards the sensor. While the assumption that the range difference between the true detection and the corresponding multipath detection is limited, is valid, the problem with this algorithm arises if any other true target appears within that threshold. In Figure 3.1 a common multipath scenario is shown in which according to Equation 2.8, the range difference between the true detection and corresponding type 1 multipath detection is 6.5 meters. Assuming a threshold around this value would lead to false positives since multiple targets can exist within this range difference, as illustrated by true target 2 in the figure where the range difference is only 5 meters.

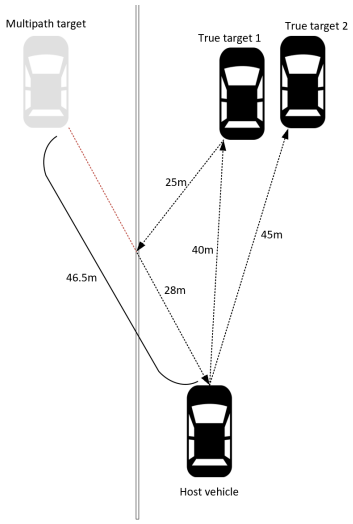


Figure 3.1: Example of multipath scenario and corresponding ranges according to the multipath geometry.

Looking at a method presented by Roos et al. [35], it stands out from the other geometric algorithms as it only operates on the properties of one object or cluster of detections instead of a pair. From a complexity point of view, this is desirable since it is worst case linear in complexity with regards to the number of objects or clusters. The algorithm estimates both a bounding box and a velocity vector for each detection cluster. When the bounding box orientation significantly deviates from the estimated velocity direction, the cluster is classified as being caused by multipath. One issue with this type of algorithm is that it requires high-resolution detections scattered in patterns that match the true shape of the vehicle. If that cannot be achieved, the bounding box estimation will be prone to large errors causing the accuracy of the algorithm to drop. An illustration of this issue can be seen in Figure 3.2 where the bounding box and velocity vector are estimated from a cluster of detections on a true target. In the figure it is clearly shown that even though the target is real, the difference between the estimated bounding box orientation and the estimated velocity direction can be large depending on how the detections are scattered [35].

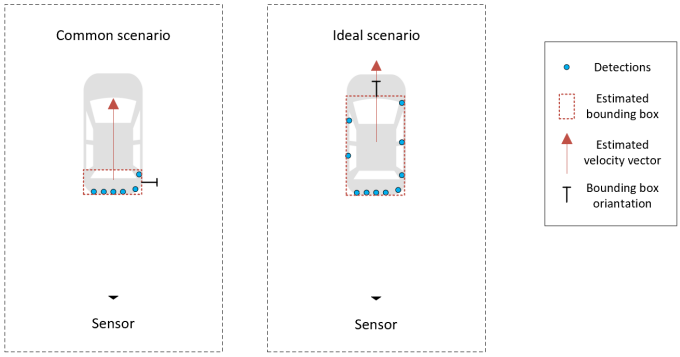


Figure 3.2: Examples of bounding box and velocity vector estimations from two clusters of detections on the same target.

Further algorithms have been developed where the reflection surface is estimated and used in the evaluation of possible multipath targets. One such method is presented by Grebner et al. [36], which uses a random sample consensus (RANSAC) algorithm to estimate a reflective surface from the radar data. To filter out multipath targets, targets on the host side of the reflective surface are mirrored onto the other side and any matching overlapping targets on the other side will be degraded in probability based on how many times they match over time. When a target probability is degraded below a threshold, it gets classified as a multipath target. Although this method promises highly accurate results it is only capable of estimating one reflective surface in the whole scene which is insufficient in many scenarios [36]. The method also assumes multiple radar sensors, which makes it incompatible with the wide range of single radar systems that exist. This also means that if there are multiple reflective surfaces present, the algorithm will struggle with producing an estimate that does not try to combine all reflective surfaces into one.

Longman et al. [19], [37], presented an algorithm that instead calculates potential reflection surfaces between a set of object pairs . The set of pairs is constructed from the assumption that the range difference between a true target and its corresponding multipath target is limited and the fact that the multipath target will be further away from the sensor than its corresponding true target. After all potential pairs along with their reflective surfaces are computed, they are evaluated against several correlation criteria to decide whether the hypothesized false target in the pair is a ghost target. The overall flow of the algorithm can be seen in Figure 3.3.

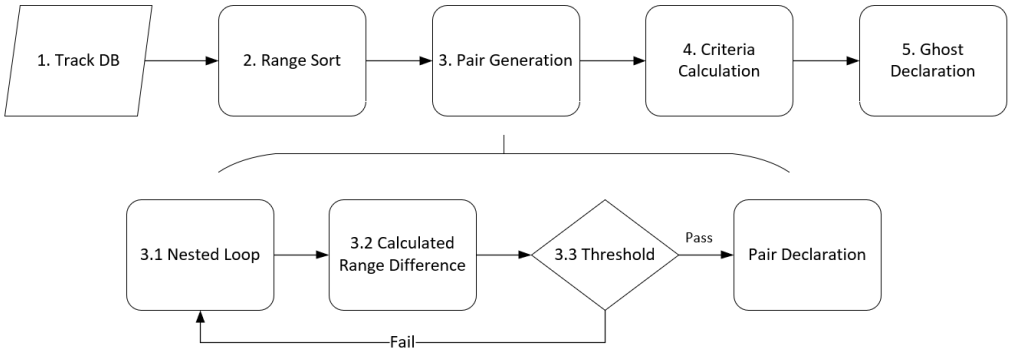


Figure 3.3: Multipath mitigation algorithm flow, from [19].

The correlation criteria are calculated as follows. It is assumed that in the track Database (DB), object position and velocity histories are available for up to T time steps back in time. The criteria build on the assumption that for the distance covered during T time steps, a reflective surface can be considered linear. Hence, $0 <= i <= T$ where i is every time step in the available history and $i = 0$ is the current time. For each pair, a reflection point R_i and slope angle α_i are calculated for each point in history available in the track DB. Here α_i is the angle between the direction of travel and the reflective surface at point R_i. Using first-order linear regression, a reflection line L is then calculated for the set of reflection points $\mathbf{R} = (R_0, R_1, ..., R_T)$. From this data, three criteria are derived. The first is called ANG and is calculated as the

mean difference between every angle in $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, ..., \alpha_T)$ and the angle of L. The second criterion is called MSD and it is the mean square distance of the calculated reflection points in $\mathbf{R}$ to the line L. Lastly, a third criterion on the reflection surface data is introduced which is called PER. PER is the percentage of reflection points in $\mathbf{R}$ which lie above a given threshold distance from the line L. The method states that a pair that consists of a multipath target and its actual corresponding true target, a so-called true pair, will produce low values for the criteria ANG, MSD and PER, whilst a pair consisting of either two true targets, two multipath targets or a multipath target and a non-corresponding true target, a so-called false pair, will produce higher values [19].

After the reflective surface criteria have been calculated, the method describes a set of further criteria that may be used to identify true pairs in the set. The first one, here denoted as MSE_v, is calculated as:

$$\text{MSE}_v = E((\widehat{\mathbf{v_{tx}}} - \mathbf{v_{tx}})^2 - (\widehat{\mathbf{v_{ty}}} - \mathbf{v_{ty}})^2)$$

where $\widehat{\mathbf{v_t}}$ is the set of estimated velocity vectors at each time step i of the true object, and $\mathbf{v_t}$ is the set of actually tracked velocity vectors of the true object [37]. The estimated velocity vector is calculated by first estimating the heading of the true object from the measured radial velocity of the two objects in the pair and the previously estimated reflection surface L, then calculating a velocity magnitude of the true object, and finally projecting that magnitude on to the estimated heading of the true object. Similarly as for MSD, a percentile criterion, PER_v, may be used for the velocity estimates in $\widehat{\mathbf{v_t}}$.

Next up, a criterion for the distance traveled of each object is introduced, here denoted as DRV. This criterion is calculated as:

$$\text{DRV} = |l(\mathbf{x_t}) - l(\mathbf{x_g})|$$

where

$$l(\mathbf{x}) = \sum_{i=0}^{T-1} ||x_{i+1} - x_i||$$

and $\mathbf{x_t}$ and $\mathbf{x_g}$ are vectors of the historic positions of the true and ghost objects respectively.

Furthermore, a criterion that compares the velocity magnitude of the multipath and true object is described, here denoted as MSE_{vm}, and is calculated as:

$$\text{MSE}_{vm} = E((||\mathbf{v_t}|| - ||\mathbf{v_g}||)^2)$$

One more criterion regarding the velocity is then introduced, here denoted as MSE_{va}, which is the difference in direction between the true object velocity and and estimated velocity calculated by mirroring the ghost object velocity over the reflective surface L. The criterion is calculated as:

$$\text{MSE}_{va} = E\left(\left(\cos^{-1}\left(\frac{\mathbf{v_t}}{||\mathbf{v_t}||} \cdot \mathbf{S}_r\right) - \cos^{-1}\left(\frac{\mathbf{v_g}}{||\mathbf{v_g}||} \cdot \mathbf{S}_r\right)\right)^2\right)$$

where $\mathbf{S}_r$ is the normalized vector corresponding to the reflective surface as:

$$\mathbf{S}_r = \begin{pmatrix} \cos \boldsymbol{\alpha} & \sin \boldsymbol{\alpha} \end{pmatrix}$$

Lastly, two final criteria are described. One, here denoted as IANG, is calculated as:

$$\text{IANG} = E(\underline{\boldsymbol{\theta}})$$

where $\underline{\boldsymbol{\theta}}$ is the vector of computed incident angles between the radar waves DOD and the reflective surface L at each time step i, and along with that another percentile criterion PER_i for the incident angles in $\underline{\boldsymbol{\theta}}$. The IANG criterion is range-dependent as Longman et al. [37], state that the incident angle can be higher at lower ranges.

Similarly as for ANG, MSD and PER it is stated that all these criteria should be low for a true pair. However, it is also noted that they may be combined and used in different ways based on the scenario encountered to get the best performance.

3.2 Machine Learning Solutions

Machine learning has proven powerful for applications which are complex to model mathematically, and with the current evolution of machine learning, nothing seems impossible as long as high-quality, labeled datasets exist for the task. As discussed in Section 2.4, there is a notable absence of high-quality datasets with multipath labeling—a critical prerequisite for effective machine learning implementations since such algorithms require annotated examples to learn the distinguishing features in the data [38]. Despite this, there are previous works that have tried to circumvent the issue by different means.

Using LiDAR point clouds, two different approaches have emerged by Alhayek et al. [39], and Danino et al. [40]. Alhayek et al. used LiDAR to label objects with the assumption that the LiDAR's high accuracy will only generate true objects. Thereafter, they trained a neural network to detect radar objects that did not exist in the LiDAR ground truth. Danino et al. developed a method for automatic labeling of automotive radar datasets using LiDAR point cloud to estimate reflective surfaces which makes it possible to identify potential multipath detections. Both of these approaches show promising results but face the sensor FOV inconsistency problem described in Section 2.4.1.

Chen et al. [41], proposed a method that leverages automatic feature extraction from the PointNet network combined with a post-extraction differentiation method [42]. Targets and ghosts are classified by their correlation between features. Some of the features used are detection range, azimuth, velocity, signal to noise ratio (SNR), number of points within distance threshold, and distance to reflective surface [41]. Using an FMCW radar statically mounted on top of a lane barrier, the authors claim 96% accuracy; however, these results are from a relatively small dataset in a very limited scenario.

For indoor environments, Feng et al. [43], developed a deep neural network that uses the spatial signals from the MIMO radar's virtual channel as an input to a

fully connected network which classifies the signals as true targets or ghosts. They evaluated their approach on both simulated and real-world data where they observed slightly better results on the simulated data.

4

Methods

4.1 Matlab Simulations

To develop a multipath identification algorithm, annotated data representing the problem is needed on which the algorithm can be tested and evaluated. Since there are no real-world datasets with good multipath annotations, data was instead generated using the autonomous driving toolbox in Matlab [44].

4.1.1 Configuring Driving Scenarios

Using the Matlab autonomous driving toolbox, driving scenarios were generated to reflect common real-world driving. The scenarios were set up using Matlab's driving scenario designer [45], in which it is possible to insert roads, barriers and different types of actors such as vehicles or pedestrians. Only cars and trucks were used as actors in the scenarios as the multipath issue appears mostly when the wave has reflected off larger road users [5]. The generated scenarios include different amounts of surrounding traffic, curved and straight roads together with lane merging and junctions, and different configurations of guardrails. Detailed descriptions of all scenarios used can be found in Appendix A.2.

4.1.2 Synthetic Multipath Data Generation

To each scenario, a radar data generator (RDG) [46], with the ability to simulate multipath behavior, can be attached to the host vehicle. The RDG can be tuned using a set of parameters to match a desired functionality. The parameters used for the simulations can be seen in Table 4.1. The output from the RDG is a list of detections for each simulation time step. Each detection has the simulated range, range-rate, azimuth- and elevation angle attached, as well as descriptions of the propagation path that created the detection. The propagation path data is not used as input to the multipath identification algorithm, however, it is essential for evaluating the performance of the algorithm as it can act as deterministic ground truth.

Table 4.1: Parameter settings used in the radar data generator, parameters not shown are left as default.

Parameter	**Value**	**Unit**	**Comment**
ScanConfig	No Scanning	-	-
UpdateRate	20	Hz	Radar cycle frequency
MountingLocation	[3.729, 0, 0.5]	m	[x, y, z] Relative host rear axle center ground
CenterFrequency	77^9	Hz	
HasRangeRate	true	-	-
HasElevation	true	-	-
HasNoise	false	-	-
HasGhosts	true	-	Enables multipath detections
HasFalseAlarms	false	-	-
FieldOfView	[120, 30]	degrees	[azimuth, elevation]
RangeLimits	[0, 250]	m	[min, max] Detectable range
RangeRateLimits	[-150, 150]	m/s	[min, max] Detectable range-rate
ReferenceRange	5	m	-
ReferenceRCS	-30	dB	-
DetectionProbability	0.9	-	Probability of detecting a target with ReferenceRCS at ReferenceRange
AzimuthResolution	0.5	degrees	-
ElevationResolution	4	degrees	-
RangeResolution	0.5	m	-
RangeRateResolution	0.1	m/s	-
TargetReportFormat	Detections	-	-
MaxNumReportsSource	property	-	-
MaxNumReports	4000	-	-

One problem discovered with the RDG is that simulating detections using resolution parameters similar to modern automotive radars, results in all of the detections ending up only on the closest targets even though the detection range limit is much further away. First, to circumvent this problem, the max number of detections outputted from the RDG was set to 4000. This allowed the RDG to output data up to a range similar to a real-world automotive radar. By filtering the outputted detections post-simulation using a fixed number of randomly selected detections per resolution cell, the detection count was reduced to a level often found in a real sensor. This way, the RDG can be configured with high resolution and still give a more realistic and evenly distributed detection spread as output.

After running the simulations, the data produced by each driving scenario was converted into a binary format compatible as input to an existing C++ RMOT simulation framework.

4.1.3 Simulation Framework

An RMOT simulation framework written in C++ was utilized to get object-level data based on all of the synthetic detections. This enabled the use of time-filtered properties, such as mentioned in Section 2.3, which could be useful in the algorithm development.

4.2 Algorithm Development

The algorithmic development can be divided into two parts. The first part is triplet identification: the task of identifying a corresponding reflection point and true object from a potential ghost object using the geometric relations described in Section 2.2.1. After a triplet has been found, the next step is to evaluate whether or not the triplet is an actual multipath triplet or not.

4.2.1 Multipath Triplet Identification

In the best case, all three objects needed for solving the geometric multipath equation are active in the current execution cycle, but sometimes due to different types of thresholding, one or two objects are missing. Commonly it is the reflection object since it is often an object like a barrier or a post which has a lower radar cross section (RCS) than for example a car or a truck [10]. Objects with low RCS will return a lower amplitude response and in high-density driving environments, lower amplitude peaks run the risk of being filtered out in the radar signal thresholding described in Section 2.1.5. To circumvent this, stationary detection predictions were introduced where all stationary detections from each execution cycle are predicted according to the current sensor host motion over four cycles ahead. This increases the likelihood of stationary detections being present for multipath triplet identification.

4.2.1.1 Radial Grid

As the multipath triplet identification relies heavily on the geometric properties described in Section 2.2.1, it is important to be able to efficiently search for objects or detections that fit these properties. To achieve this, a spatial data structure in the form of a radial grid was introduced. A radial grid has multiple benefits when it comes to multipath triplet identification.

Firstly, it helps cluster detections into cells which is beneficial when trying to determine if an object or detection is occluded by something else from the sensors point of view. Having the detections in cells can reduce the complexity of searching at a single angle at a time and reduce the risk of missing a target which only got one detection but in reality is bigger than a single point in space.

Secondly, the radial grid structure further reduces the computational complexity when determining whether an object or detection is occluded or within LOS, as it enables sequential processing of cells within individual azimuth bins.

Finally, using a grid also allows for utilizing lazy updates with time stamped input data. This keeps the complexity of updating the grid in the order of number of input detections which is considered low compared to other spatial data-structures [47].

In Table 4.2 the parameters of the radial grid are presented.

Table 4.2: Radial Grid variables and parameters.

az_i	Azimuth bin at resolution index i.
r_j	Range bin at range index j.
N_i	Number of azimuth bins at azimuth resolution i.
$\mathbf{k} = \{k_1, k_2, ..., k_{M_{tholds}}\}$	Vector of azimuth resolution thresholds [m].
G_r	Grid max range.
G_a	Grid azimuth FOV.
res_r	Grid range resolution.
$res_{az,i}$	Grid azimuth resolution at resolution index i.
R	Number of range bins. Fixed according to G_r/res_r
M_{tholds}	Number of resolutions thresholds

A modulation scheme was implemented to maintain consistent spatial resolution of each cell by varying the number of azimuth bins across different range intervals. Specifically, for any range r where $k_i \leq r < k_{i+1}$, the system utilizes N_i azimuth bins. This approach ensures that each cell maintains approximately equal area throughout the detection range, compensating for the natural angular spread that occurs with increasing distance. The number of azimuth bins is calculated according to:

$$N_i = 2^{(i-1)} N_0 \tag{4.1}$$

where N_0 is the initial number of azimuth bins. The total number of cells in the grid is calculated as:

$$\sum_{i=0}^{M_{tholds}} N_i(k_i - k_{i-1}) \tag{4.2}$$

As illustrated by Figure 4.1, within the first azimuth resolution threshold k_1, there is $N = N_0$ number of azimuth bins, and for every forthcoming azimuth resolution threshold, the number of azimuth bins is doubled. The correspondence between range bin index r_j and physical range r can simply be determined using:

$$r_j(r) = \left\lfloor \frac{r}{res_r} \right\rfloor, \quad \text{if } r \geq 0. \tag{4.3}$$

Combining az_i and r_j gives us the grid cell $c_{i,j}$.

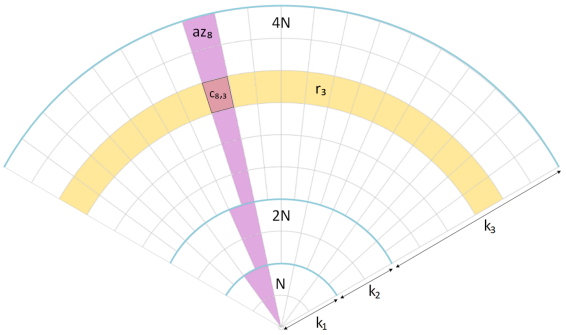


Figure 4.1: Radial grid structure visualization. The yellow region represents a range bin, while the purple section indicate an azimuth bin. Blue lines denote the azimuth resolution thresholds (k_i). The illustration demonstrates how the number of azimuth bins increases with range, where N represents the initial number of azimuth bins at the closest range.

4.2.1.2 Host Motion Prediction

The host motion is predicted using a state-space model presented in Table 4.3. All values use the rear axle center as a reference point and the non-zero initial values are collected from either the input simulation data or scenario configuration.

Table 4.3: State-space model used for host motion prediction.

Parameter	**Unit**	**Description**	**Initial value**
x	m	Host rear axle center position	-3.729
y	m	Host rear axle center position	0
speed	m/s	Host speed	From input
acceleration_x	m/s^2	Host acceleration in x-direction	From input
acceleration_y	m/s^2	Host acceleration in y-direction	From input
angle	rad	Host rotation angle	0
yaw_rate	rad/s	Host yaw rate	From input
sideslip	rad	Host sideslip angle	0

In each scan, a new host state is initialized and predicted four scans forward. The prediction for each time step is stored in a list to later be used when predicting the position of stationary detections. The host motion prediction is described in Algorithm 1 below where atan2 is defined as:

$$\text{atan2}(x,\ y) = \begin{cases} \arctan\left(\frac{x}{y}\right), & y > 0 \\ \arctan\left(\frac{x}{y}\right) + \pi, & x \geq 0, y < 0 \\ \arctan\left(\frac{x}{y}\right) - \pi, & x < 0, y < 0 \\ \frac{\pi}{2}, & x > 0, y = 0 \\ -\frac{\pi}{2}, & x < 0, y = 0 \\ 0, & x = 0, y > 0 \\ \pi, & x = 0, y < 0 \\ 0, & x = 0, y = 0 \end{cases}$$

Algorithm 1 Host motion prediction.

$H = [5]$ ▷ List to store predicted host states of size 5
$T = 0.05$ ▷ Time step for one scan given the RDG cycle frequency
$h_0 =$ Initial host state
$H[0] \leftarrow h_0$
for $i = 1$ **to** 4 **do**
  $h \leftarrow H[i-1]$
  h.speed $\leftarrow h$.speed $+$ h.acceleration_x $*$ T
  h.angle $\leftarrow h$.angle $+$ h.yaw_rate $*$ T
  h.sideslip $\leftarrow$ atan2(h.acceleration_y $*$ T, h.speed)
  h.x $\leftarrow h$.x $+$ h.speed $*$ cos(h.angle $+$ h.sideslip) $*$ T
  h.y $\leftarrow h$.y $+$ h.speed $*$ sin(h.angle $+$ h.sideslip) $*$ T
  $H[i] \leftarrow h$
end for

4.2.1.3 Predictive Grid Population

In each scan, the grid is populated using the simulated input detections where each detection is placed in the cell spanning its position. All stationary detections are added both to its current cell and to all cells spanning its predicted positions over the next four scans. The detections are predicted by translating their current position using the origin position shifts computed in the host motion prediction. An illustration of how the grid is populated can be seen in Figure 4.2. Pseudo code for the grid population can be found in Algorithm 2.

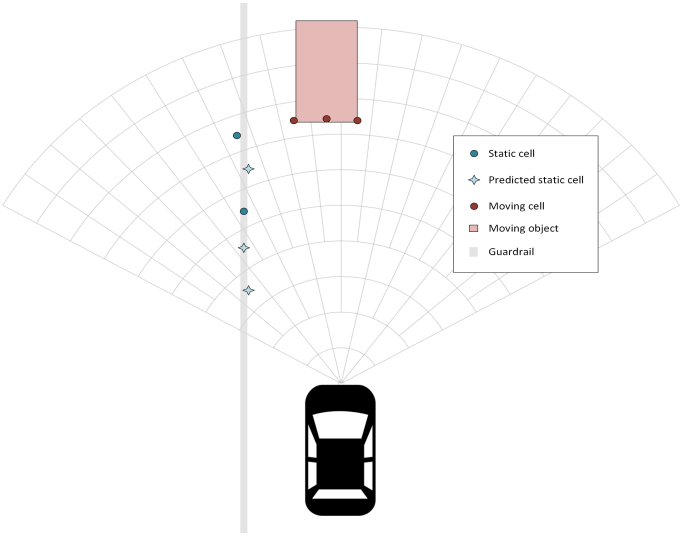


Figure 4.2: Simplified illustration of the radial predictive grid.

Algorithm 2 Grid Population Algorithm

```
DB_d = [N_d]                                   ▷ Detection database of size N_d
Grid = [M_g]                                   ▷ Initialize empty grid of size M_g
for i = 1 to N_d do
    D ← DB_d[i]                                ▷ Get detection from database
    idx ← CalculateGridIndex(D.range, D.azimuth)
    if D.moving then
        Grid[idx].moving.add(D)                ▷ Insert moving detection
    else
        Grid[idx].static ← D                   ▷ Insert static detection
        for j = 1 to M_p do                    ▷ Generate predicted detections
            pred_idx ← CalculatePredictedIndex(D)
            Grid[pred_idx].pred ← D            ▷ Insert prediction
        end for
    end if
end for
```

When querying a grid cell, predicted detections are only used if no current stationary detection is present, and the timestamp the detection was predicted into matches the timestamp of the current execution cycle.

4.2.1.4 LOS Identification

After populating the grid, each azimuth bin is iterated sequentially from the origin outward to identify cells containing LOS detections. Any detections that are in the first occupied bin from the origin and out are classified as in LOS. After that, any detections that occupy a cell further away in the current bin will not be classified as in LOS unless the difference in either azimuth or elevation angle is larger than a range-dependent threshold for all other identified LOS detections in the bin. Pseudo code for the LOS calculation can be found in Algorithm 3.

Algorithm 3 LOS Calculation Algorithm

```
Grid = [M_g]                                              ▷ Populated grid
LOS = [M_az][ ]                          ▷ Initialize LOS array, max size M_az × M_LOS
for i = 1 to M_tholds do
    for a = 0 to AZ_sizes[i] − 1 do                       ▷ Iterate through azimuth bins
        for r = 0 to M_r − 1 do                           ▷ Iterate through range bins
            idx ← CalculateGridIndex(r, a)
            az_bin ← GetAzimuthBin(r, a)
            for all cell_item ∈ Grid[idx] do              ▷ Check all items in cell
                if IsOutsideThreshold(cell_item, LOS[az_bin]) then
                    LOS[az_bin].add(cell_item)            ▷ Update LOS list for azimuth bin
                end if
            end for
        end for
    end for
end for
```

4.2.1.5 Forming Multipath Triplets

To form the multipath triplets, all active objects are tested as potential ghost objects. From the geometric properties described in Section 2.2.1, it is known that a ghost object appears behind a reflection surface. Therefore, all LOS points in the ghost object azimuth bin between the ghost object and the origin are iterated and tested as possible reflection points. For each reflection point, a sweep is performed over all cells that fulfill the geometric properties for the location of the true object. As there is no way of telling the angle of the reflection surface at this stage, the sweep is performed for all possible reflection angles in the plane. In Figure 4.3 an illustration of the sweeping process is shown. Point G is the reference point for the ghost object, point B is the reflection point under test, point S is the sensor, and points t_1 through t_4 are all the possible positions of a true object for the current sweep angle α.

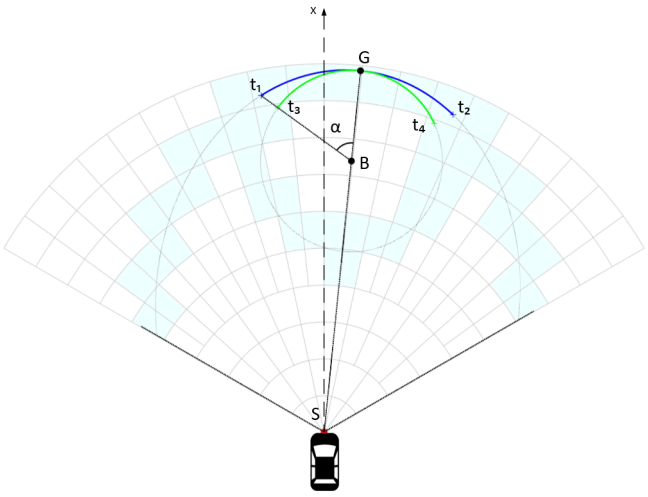


Figure 4.3: Crude illustration of the sweeping process.

To compute all possible positions of the points t_1 through t_4, the angle α is swept from 0 to π. For each step of the angle, the positions of the points t_1 through t_4 are computed as follows. Starting with the points t_1 and t_2, which are the points corresponding to a type 1 multipath case, it is known from Equation 2.8 that the range from the sensor to point t_1 and t_2 respectively, here denoted as D_{T1}, can be described as:

$$\begin{aligned} g &= ||\overrightarrow{SG}||,\ b = ||\overrightarrow{SB}|| \\ r &= ||\overrightarrow{Bt_1}|| = ||\overrightarrow{Bt_2}|| \\ D_{T1} &= ||\overrightarrow{St_2}|| = ||\overrightarrow{St_1}|| = 2g - b - r \end{aligned} \tag{4.4}$$

The distance r can also be described using the cosine theorem:

$$\begin{aligned} D_{T1}^2 &= b^2 + r^2 - 2bcos(\pi - \alpha)r \Rightarrow \\ r &= \frac{2g(g-b)}{bcos(\alpha) - b + 2g} \end{aligned} \tag{4.5}$$

Substituting Equation 4.5 into 4.4, the range to the points t_1 and t_2, D_{T1}, can be computed.

$$D_{T1} = 2g - b - \frac{2g(g-b)}{bcos(\alpha) - b + 2g} \tag{4.6}$$

Using the computed range D_{T1}, the angle of the points t_1 and t_2 can be derived from the difference in angle between the ghost or reflection point and either of the true points, here denoted as δ_{T1}.

$$\begin{aligned} \delta_{T1} &= |\angle t_1SB| = |\angle t_2SB|, \\ r^2 &= D_{T1}^2 + b^2 - 2D_{T1}bcos(\delta_{T1}) \Rightarrow \\ \delta_{T1} &= cos^{-1}\left(\tfrac{r^2 - b - D_{T1}^2}{-2D_{T1}b}\right), \end{aligned} \tag{4.7}$$

$$\angle t_1SX = \angle BSX + \delta_{T1}, \quad \angle t_2SX = \angle BSX - \delta_{T1}$$

Similarly, the positions of the type 2 multipath points, t_3 and t_4, can be computed using the same logic. Combining Equation 2.9 and the cosine theorem, the range from the sensor to point t_3 and t_4 respectively, here denoted as D_{T2}, can be described as:

$$D_{T2} = \sqrt{b^2 + (g-b)^2 - 2bgcos(\pi - \alpha)} \tag{4.8}$$

The angle of the points can then be computed, again using the cosine theorem.

$$\begin{aligned} \delta_2 &= |\angle t_1SB| = |\angle t_2SB| \\ r^2 &= D_{T2}^2 + b^2 - 2D_{T2}bcos(\delta_{T2}) \Rightarrow \\ \delta_{T2} &= cos^{-1}\left(\tfrac{r^2 - b - D_{T2}^2}{-2D_{T2}b}\right), \end{aligned} \tag{4.9}$$

$$\angle t_3SX = \angle BSX + \delta_{T2}, \quad \angle t_4SX = \angle BSX - \delta_{T2}$$

When the radial coordinates of the possible true object positions t_1 through t_4 have been computed, the coordinates are converted to grid indices using Equation 4.3. As computing the positions t_1 through t_4 is a computationally heavy process, a lookup table was constructed to be able to iterate all matching cells given a ghost and a reflection cell. If any detections associated to an object are present in the queried grid cells, the associated objects are used to form multipath triplets. Pseudo code for the triplet identification as well as ghost classification can be found in Algorithm 4.

Algorithm 4 Triplet Identification Algorithm

```
DB = [N_o]                                          ▷ Track database of size N_o
LOS = [M_az][M_LOS]                                 ▷ LOS objects/detections per azimuth bin
Sweep = [M_g][M_r]                                  ▷ Lookup table for grid sweep
for i = 1 to N_o do
    G ← DB[i]                                       ▷ Get potential ghost object from database
    is_ghost = false
    highest_prob = 0
    for all R ∈ LOS[G.az_bin] do                    ▷ For each LOS point in current azimuth bin
        if R.range < G.range then
            for all T ∈ Sweep[G.idx, R.r_bin] do    ▷ For each true object candidate
                r̂r ← CalculateRangeRate(G, R, T)    ▷ Using Eq. 2.13
                x ← |r̂r − G.rr|                     ▷ Calculate range-rate difference
                p_i ← CalculateProbability(x)       ▷ Using Eq. 4.11
                if p_i > highest_prob then
                    is_ghost = p_i > t_i            ▷ Evaluate against threshold
                    highest_prob = p_i
                end if
            end for
        end if
    end for
    G.is_ghost = is_ghost
end for
```

4.2.2 Multipath Triplet Evaluation

After a potential multipath triplet has been identified, it is evaluated against the velocity properties described in Section 2.2.2 to decide whether or not the triplet is a true multipath triplet.

4.2.2.1 Triplet Categories

Given the two types of multipath under consideration (type 1 and type 2), each with different geometric relations, the triplets can be categorized into two corresponding subsets. Additionally, as the triplets will be evaluated against velocity-based properties, it makes sense to further split the subsets based on each members motion status. This categorization serves as an analytical framework, grouping triplets which are more likely to follow the same patterns and thus make each triplet easier to classify.

An illustration of the categorization can be seen in Figure 4.4 where a total of 16 categories are established.

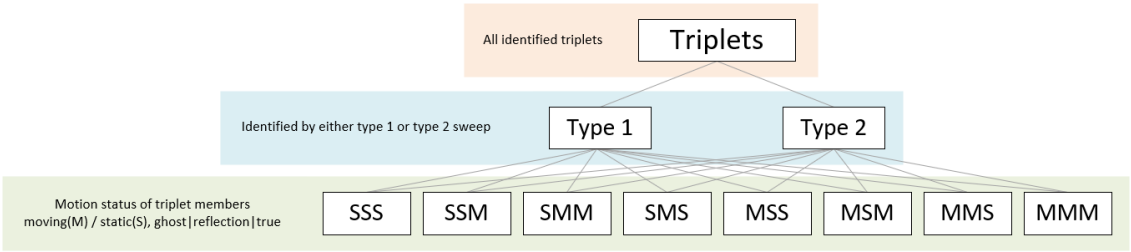


Figure 4.4: Triplet categorization.

4.2.2.2 Velocity-Based Criteria

For each triplet, the theoretical ghost range-rate is calculated using Equation 2.13 and compared to the measured range-rate of the ghost object. The difference between the theoretical and measured value is here denoted as range-rate difference. For a true triplet the range-rate difference should be small, whereas for false triplets the difference should generally be larger. However, since the velocity vectors of the objects are just estimations, there is no definitive split between the differences calculated for true and false triplets.

To accommodate for errors in the velocity estimations, a statistical approach is used where a probability is calculated for each triplet. Firstly, using the ground truth data from the simulations, exponential distributions of the range-rate differences were approximated for the true and false triplets in each category. The distributions X_i^t and X_i^f for each category i are defined as follows:

$$\begin{aligned} \lambda_i^t &= \frac{1}{E(\hat{X}_i^t)} \\ X_i^t(x) &= \lambda_i^t \ * \ e^{-\lambda_i^t \ * \ x} \\ \lambda_i^f &= \frac{1}{E(\hat{X}_i^f)} \\ X_i^f(x) &= \lambda_i^f \ * \ e^{-\lambda_i^f \ * \ x} \end{aligned} \tag{4.10}$$

Here x is the range-rate difference, $\hat{X}_i^t$ is the distribution of measured range-rate differences for all true triplets in class i, and $\hat{X}_i^f$ is the distribution of measured range-rate differences for all false triplets in class i.

The probability is then calculated using the range-rate difference x, for the triplet under evaluation, according to the equation below.

$$\begin{aligned} p_i^t &= X_i^t(x) \\ p_i^f &= X_i^f(x) \\ p_i &= \frac{p_i^t}{p_i^t + p_i^f} \end{aligned} \tag{4.11}$$

Given the probability p_i from Equation 4.11, a threshold value for each of the triplet categories is then tuned to maximize the triplet classification accuracy.

4.2.3 Ghost Object Classification

After the probabilities have been calculated for each triplet evaluated using a potential ghost object, the object's class is determined by the triplet with the highest probability. Since one potential ghost can participate in multiple triplets, this statistical approach enables classification based on the most probable triplet category.

The object under test is classified as a ghost if the probability of the most probable triplet is above it's corresponding threshold and hence is classified as a true triplet.

4.3 Baseline Algorithm

The algorithm proposed by Longman et al. [19] was chosen as the baseline algorithm for multiple reasons. Firstly, the algorithm classifies multipath at object level which is desirable since the algorithm proposed in this paper also operates in that scope. Longman's algorithm can also classify objects independently of the number of reflection surfaces apparent in the surrounding environment which again puts it in the same scope as the algorithm proposed in this paper. Finally, the algorithm promises high performance which makes it qualified to be used as a state-of-the-art benchmark.

4.3.1 Implementation

To be able to set a benchmark, the state-of-the-art algorithm was implemented in the same C++ RMOT framework used in the development of the algorithm proposed in this paper. As the literature does not state any actual values used as thresholds for the given criteria, the algorithm was first run exhaustively to log statistics for all necessary parameters. Further, since there are no details mentioned on how many historic data points to use in the equations, it was assumed that at least three historic data points are required for each object to be valid in the pair generation step. This is because at least three points are required to perform the linear regression used in the criteria. For long-lived objects, the last ten data points were used. Statistics were collected over two different sets of driving scenarios, one composed of scenarios similar to the ones used in the literature presenting the state-of-the-art algorithm, and the other composed of more varying scenarios reflecting everyday normal urban driving. For each set of statistics, the parameter thresholds were optimized to provide maximized discrimination between the true and false multipath pairs.

5

Evaluation Setup

5.1 Scenarios

To evaluate both the PRED-RAG algorithm and the baseline algorithm, fifteen driving scenarios were constructed to represent real-world driving. Each scenario aims to catch a limited set of multipath situations, meanwhile the whole collection of scenarios aims to cover a variety of urban driving situations. Each scenario is described in more detail in Section A.2.

5.1.1 Set 1

Set 1 consists of four scenarios where there are few reflective surfaces, few targets and only targets moving in similar direction as host. The set is similar to the scenarios that the authors of the baseline algorithm evaluated their algorithm on. Details of set 1 is found in Appendix A.1.

5.1.2 Set 2

Set 2 represents a more general and thorough scenario composition than set 1. It consists of twelve scenarios with mixed complexity and diverse traffic patterns to better represent condition encountered in everyday driving. Details of set 2 is found in Appendix A.2.

5.2 Ground Truth Mapping

The synthetic radar data includes propagation path descriptions for each detection. Direct reflections contain a single target ID, while multipath detections list all reflection points. This information provides essential ground truth for evaluating multipath identification algorithms. Although ground truth exists at the detection level, evaluation was conducted at the object level to match the algorithms' operation. To address objects with mixed detection types, the following criterion was established: true objects must have at least one true detection, while ghost objects must have none. Using propagation path information, ground truth data was also created defining all true multipath pairs—each ghost object linked to its corresponding true object.

As the ground truth provided by the simulations is at detection level, and since objects can have a mix of detection types associated with them, there are still occasions where this ground truth criterion fails. For example, objects with mixed detection types, including direct reflections, can still be regarded as ghost objects if their properties diverge significantly from what they would have been if only direct reflection detections were used. To solve this issue, another criterion was established which checks if all associated direct reflections have different motion status compared to all associated multipath detections. If they do, the object is labeled as a ghost even though it has direct reflections associated to it.

However, even with the specific countermeasure described above, there are still cases where the ground truth labeling fails. If a ghost object forms overlapping with a true object, there is a risk that the ghost object gets direct reflection detections associated with it which actually belong to the true object. This can lead to objects being wrongly labeled and, in turn, influence the measured accuracy of the evaluated algorithm. The risk of this happening though is small, and a random sample evaluation of the ground truth labels shows that they are still above 99% correct for prio 4 objects.

5.3 Priority

As established previously, there is a varying degree of danger a ghost object can impose. Different priorities for all objects were therefore introduced to be able to evaluate the performance of the algorithm from a safety-focused perspective. To be able to divide the objects into priorities, a rectangular zone was introduced that is 50 meters long and 14 meters wide in both directions laterally, with origin at the sensor midpoint. In Figure 5.1 an illustration of the priority zone is shown.

The rationale for choosing the length of the zone comes from the National Highway Traffic Safety Administrations (NHTSA) requirement that a forward collision warning (FCW) system should act latest at 2.1 seconds time to collision (TTC) at a speed of 20 meters per second approaching a stationary target [48]. Adding some overhead to the resulting range of 42 meters, it can be assumed that a target should be identified at a minimum of 50 meters range to safely pass the requirement. For choosing the width of the zone, it was reasoned that everything within at most four lanes to each side must be regarded as important. In Sweden a road lane is at maximum 4 meters wide which implies that $3 * 4$ meters on both sides of sensor origin must be covered. To include the host lane, $0.5 * 4$ meters is added to both sides resulting in 14 meters of width [49].

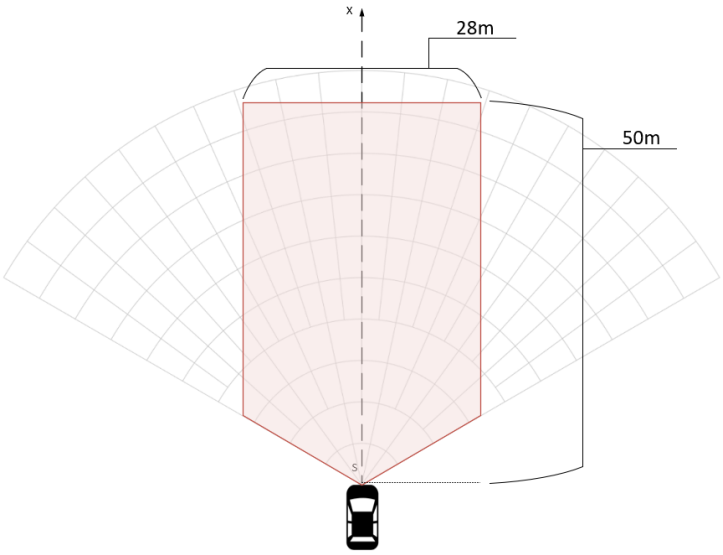


Figure 5.1: Illustration of priority zone.

From the zone, four different priority classes were define with 1 being the least important and 4 being the most important:

1. Static object not in zone.
2. Static object in zone.
3. Moving object not in zone.
4. Moving object in zone.

Each priority class describes a ghost object's importance in relation to its position and motion status. In general, static objects pose very little danger as they cannot be predicted to move into the host path. Secondly, due to the geometric properties of multipath, they will always be occluded by their reflection object, and therefore they are regarded as unimportant since it can be assumed that no false static objects will be between a host vehicle and a true static object. Since it is still of interest to evaluate the performance including static objects both within and outside of the zone separately, they are two separate priority classes.

Moving objects can be predicted to move into the host path and are therefore important to identify. In general, objects at great distances will have fewer detections and motion prediction confidence, therefore the risk of measurement inaccuracies is higher and the criterion may become misleading. To be able to evaluate performance for moving objects within and outside of the zone, they are divided into two separate priority classes. The purpose of the priority scheme is to help differentiate algorithmic performance depending on ghost object identification importance.

5.4 Identification Performance

To measure the identification performance for both the PRED-RAG algorithm and baseline algorithm, the algorithm's classification results are compared with the ground truth matching. In the comparison, objects meeting the requirements outlined in Section 1.1 are referred to as "in scope". Performance metrics for objects in scope will be presented in Section 6.3, while metrics for all objects can be found in Appendix A.4. Further, a classification is regarded as correct if an actual ghost object is classed as a ghost at least once during the current execution cycle. This implies that even if the algorithm assigns different classifications to the same object when paired with different true objects, successfully classifying it once is sufficient for it to be considered a true positive. For instances where the algorithm has classified an object at least once as a ghost object, but it is not marked as a ghost in the ground truth, it will lead to a false positive. An object that is marked as a ghost in the ground truth but does not appear or is never classified as a ghost will be counted as a false negative. According to the counting described above, recall, precision, accuracy, and f1-score are calculated as performance metrics.

5.5 Runtime and Complexity

As a measurement of the real-time performance for both the PRED-RAG algorithm and baseline algorithm, average runtime and theoretical complexity will be computed. The algorithm's runtime is measured using a simulation framework written in C++ on a PC. This implies that runtime may vary depending on system load, therefore the measurements are averaged over 10 executions for the whole set of scenarios evaluated. Even though the hardware is not the same as a production system would use, the comparison between the two algorithms gives an indication of which one is likely to be better in terms of runtime. Besides execution time, theoretical worst-case complexity will be derived as a universally comparable metric. The hardware used to run the algorithms can be seen in Table 5.1.

Table 5.1: Evaluation hardware.

CPU	Intel Core Ultra 7 165H, 3800 Mhz, 16 Cores, 22 Logical Processors
Physical Memory	64 Gb
Operating System	Windows 10 Enterprise

6

Results

6.1 Comparison

Performance evaluation on the less complex scenario set (set 1) reveals that both algorithms perform with high accuracy, as shown in Figure 6.6 and 6.1, with the PRED-RAG algorithm demonstrating a slight advantage. When evaluated on the more general scenario set (set 2), the PRED-RAG algorithm exhibits significantly better generalization capabilities. Table 6.1 clearly demonstrates that the PRED-RAG algorithm outperforms the baseline algorithm across all performance metrics, regardless of priority level, when tested on set 2. Table 6.2 presents the runtime mean and standard deviation for both algorithms when run on the two scenario sets. The baseline algorithm achieves the lowest runtime on scenario set 1 whilst the PRED-RAG algorithm demonstrates better computational efficiency on scenario set 2.

Table 6.1: Performance metrics for both algorithms on scenario set 2 in scope objects.

	Priority	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Baseline	4	39.26%	90.07%	4.74%	9.01%
	1-4	56.97%	42.13%	24.52%	31.0%
Proposed	4	94.43%	96.43%	94.72%	95.57%
	1-4	89.23%	91.80%	79.79%	85.38%

Table 6.2: Mean scan runtime.

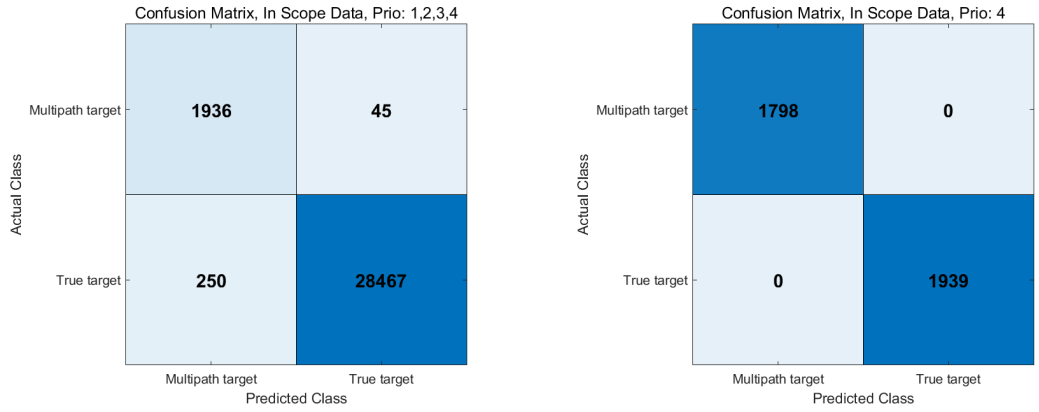
	Scenario Set	**Runtime mean [ms]**	**Runtime standard deviation [ms]**
Baseline	1	4.77	0.85
	2	12.21	14.1
Proposed	1	6.98	1.54
	2	7.36	4.43

6.2 Baseline Algorithm

According to the optimized thresholds described in Section 4.3.1, the baseline algorithm was evaluated under three conditions: first on scenario set 1 and 2 using parameter thresholds optimized for set 1, and then on scenario set 2 with thresholds optimized for set 2.

6.2.1 Identification Performance Evaluation

The identification performance on scenario set 1 with parameter thresholds optimized for that set can be seen in Figure 6.1. As seen in the figure, the algorithm performs very well on the scenarios in set 1 with an accuracy of 100% for priority level 4. The statistics from scenario set 1 with the thresholds used and the results for all priority levels can be found in Appendix A.3.

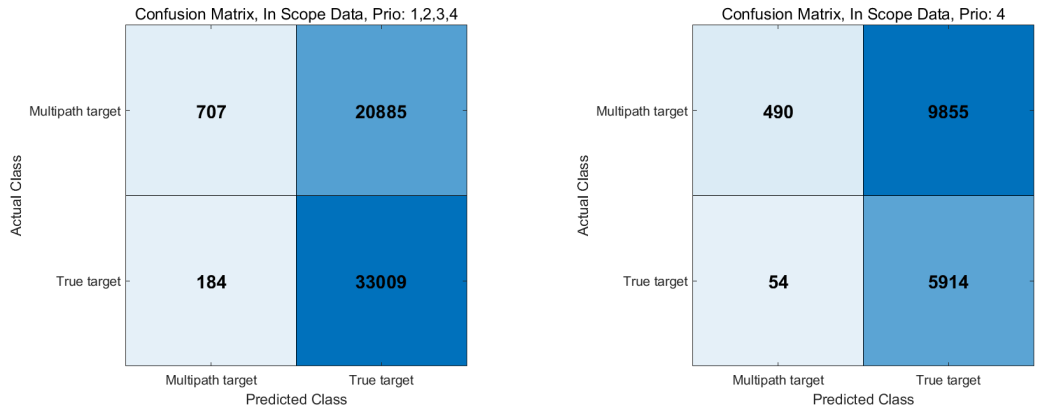


(a) Performance over all priority levels.

(b) Performance over priority level 4.

Figure 6.1: Performance of the baseline algorithm on scenario set 1.

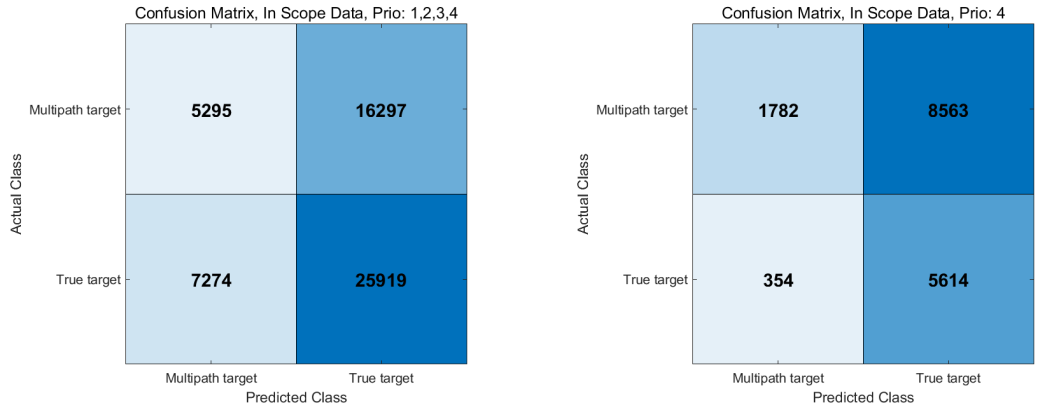
When the baseline algorithm is evaluated on the more general set of scenarios (set 2), but with the threshold optimized for set 1, there is a clear degradation in performance. The results for priorities 1 through 4 and 4 alone from the run on set 2 can be seen in Figure 6.2.



(a) Performance over all priority levels. (b) Performance over priority level 4.

Figure 6.2: Performance of the baseline algorithm on scenario set 2 using threshold from scenario set 1.

With set 2 optimized thresholds, it can be seen in Figure 6.3 that there is a slight increase in performance. However, it is still clear that the algorithm struggles with the more general scenarios. The statistics from scenario set 2 along with the thresholds used and the results for all priority levels can be found in Appendix A.3.



(a) Performance over all priority levels. (b) Performance over priority level 4.

Figure 6.3: Performance of the baseline algorithm on scenario set 2 using tuned parameters.

6.2.2 Complexity

According to Figure 3.3, first of all, sorting of all objects in the track DB is performed. Sorting of a nearly sorted list can be done in the best case with $\mathcal{O}(n)$ complexity using insertion sort but with $\mathcal{O}(n^2)$ for an unsorted list [50]. Under the assumption that the tracking database is kept nearly sorted between each execution cycle, insertion sort would be the fastest sorting algorithm. As shown in Algorithm 5, in the pair generation, there is a nested loop where the object pairs range difference is compared

with the requirement that a true object and a ghost object cannot be too far apart. Thus the number of comparisons, which is also the max number of pairs, becomes:

$$\text{Max number of pairs} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j \Rightarrow \frac{(n-1)n}{2} \tag{6.1}$$

The complexity of the pair generation step is thereby $\mathcal{O}(n^2)$. In the criteria evaluation step, constant complexity can be assumed for every pair since every criterion can be calculated in constant time. This leads to a total assumed and worst-case complexity of $\mathcal{O}(n^2)$.

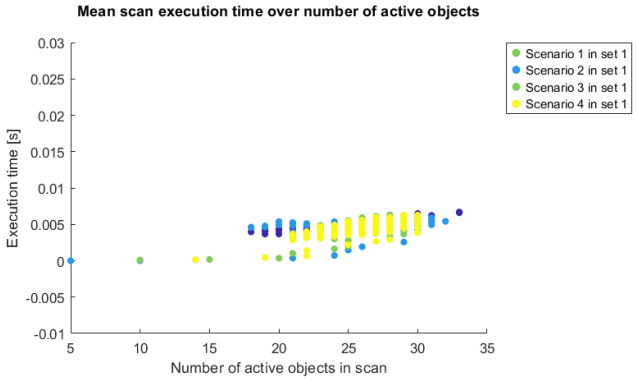
Algorithm 5 Longman et al. [19] pair generation.

```
DB = [n]                                   ▷ Sorted Track database of size n
Pairs = [ ]                                ▷ List of object pairs
T = Range difference threshold
for i = 1 to N do
    for j = i to N do
        r = DB[j].range − DB[i].range      ▷ Range difference
        if r < T then
            Pairs.add([i, j])
        end if
    end for
end for
```

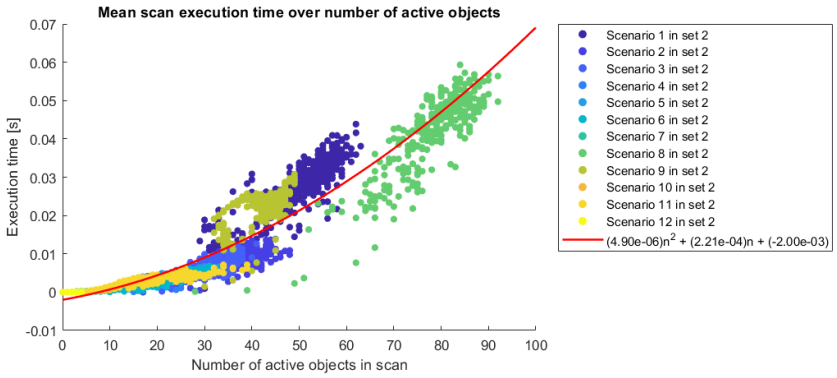
6.2.3 Runtime

Looking at Table 6.2, where the runtime mean and standard deviation for one scan when running the algorithm ten times on the two scenario sets are presented, it is clear that the mean runtime is much lower for the scenarios in set 1 where the algorithm also performed well in regard to accuracy.

In Figure 6.4 the mean scan runtime over the number of active objects in the scan is presented for the two sets of scenarios. As the figure shows, the runtime for scenario set 1 is clustered around the same area which is expected since the scenarios in the set are very similar. Looking at Figure 6.4a however, the quadratic complexity derived in Algorithm 5 becomes prominent.



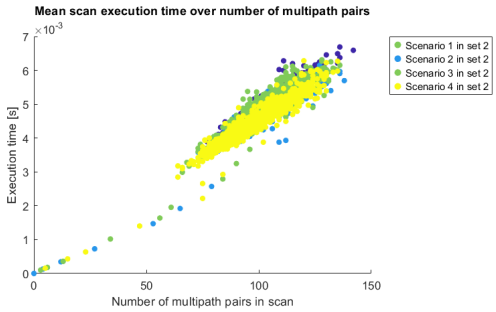
(a) Scenario set 1.



(b) Scenario set 2.

Figure 6.4: Baseline algorithm mean runtime per scan over the number of currently active objects.

Figure 6.5 shows the mean scan runtime over the number of multipath pairs generated by the nested pair generation loop seen in Figure 3.3. As mentioned above, the execution time for computing and evaluating the criteria of a multipath pair can be assumed as constant, resulting in the linear correlation seen in the figure below.



(a) Scenario set 1.

(b) Scenario set 2.

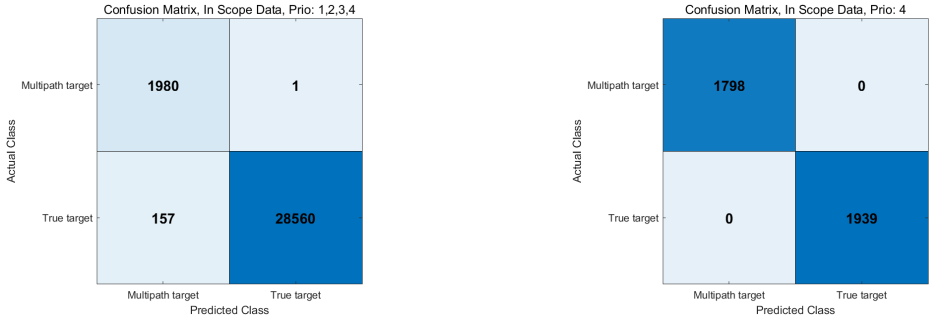
Figure 6.5: Baseline algorithm mean runtime per scan over the number of multipath pairs generated.

6.3 PRED-RAG Algorithm

Similarly as for the baseline algorithm, the PRED-RAG algorithm was evaluated on both scenario set 1 and 2. A key advantage demonstrated during evaluation is that the PRED-RAG algorithm performs effectively across both scenario sets without requiring any tuning, unlike the baseline algorithm which, even after tuning, struggles with the more general data in set 2.

6.3.1 Identification Performance Evaluation

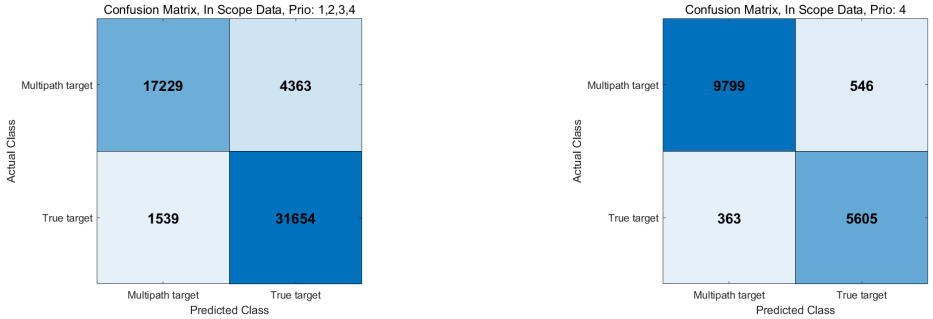
As seen in Figure 6.6, on scenario set 1, the PRED-RAG algorithm has equal performance for level 4 priority as the baseline algorithm and slightly better performance over all priority levels.



(a) Performance over all priority levels. (b) Performance over priority level 4.

Figure 6.6: Performance of the PRED-RAG algorithm on scenario set 1.

Evaluation on scenario set 2 reveals that the PRED-RAG algorithm outperforms the baseline algorithm. As seen in Figure 6.7, the PRED-RAG algorithm has a slightly higher number of true targets classed as ghosts (false positives), but significantly less ghost targets classed as true targets (false negatives), for priority level 4. Across all priority levels, the PRED-RAG algorithm demonstrates a substantially lower number of false negatives and false positives compared to the baseline algorithm and thus achieves higher accuracy than the baseline approach.

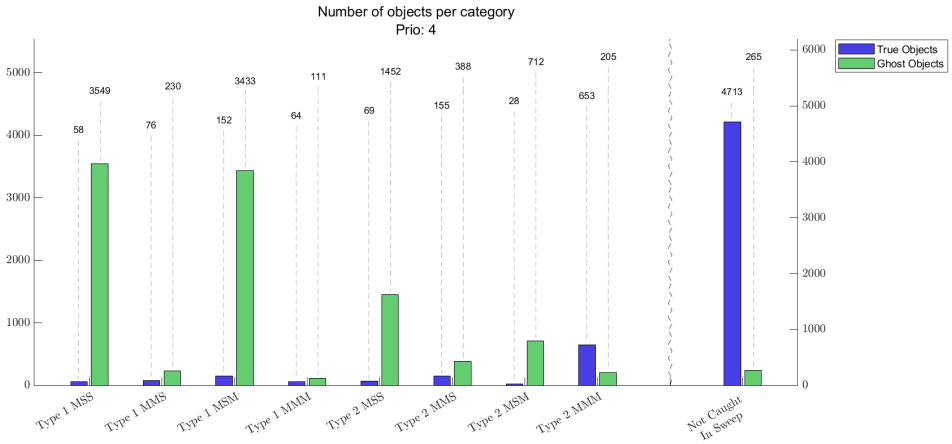


(a) Performance over all priority levels. (b) Performance over priority level 4.

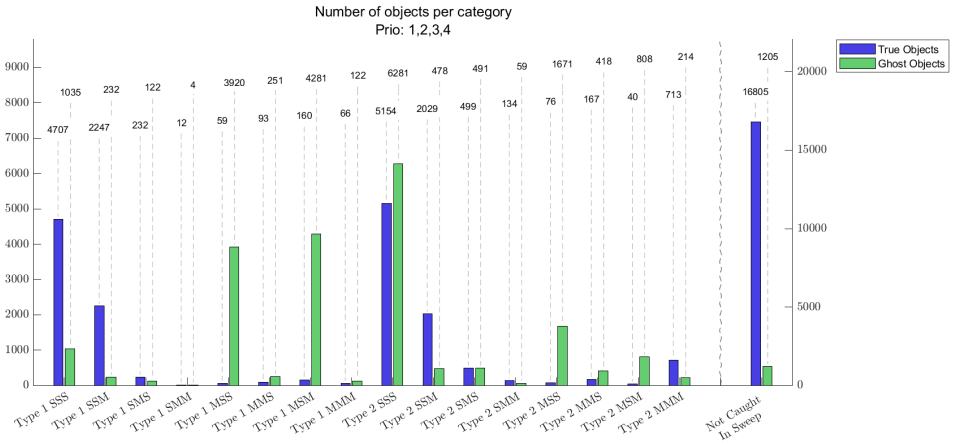
Figure 6.7: Performance of the PRED-RAG algorithm on scenario set 2.

In Figure 6.8 the number of objects per category is presented for prio 4 and prio 1 through 4 respectively. As shown in the figures, the geometric properties employed in the grid sweep algorithm effectively form triplets, separating the majority of ghost objects from true objects.

Figure 6.9 shows the number of classifications made for each type of reflection point. The majority of classifications are made using static reflection points, second to most using reflection points predicted from historic detections, and least using moving reflection points.



(a) Prio 4



(b) Prio 1 through 4

Figure 6.8: Number of in-scope objects evaluated by the range-rate criterion per class.

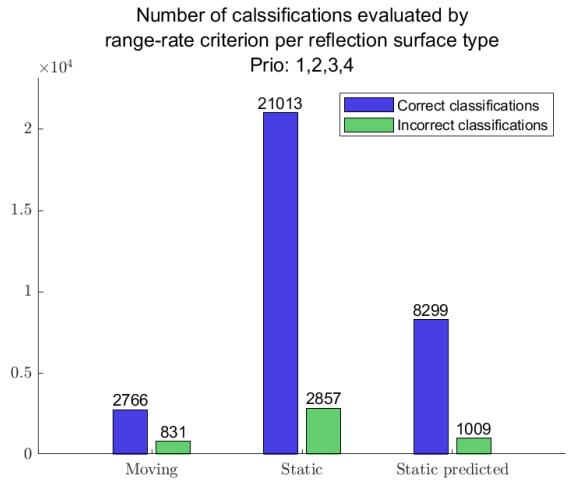
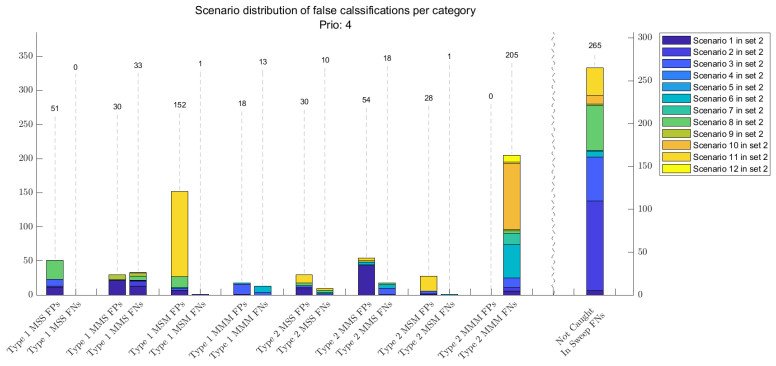
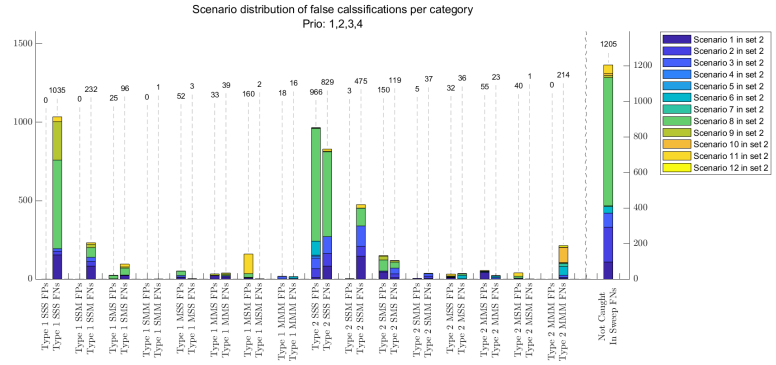


Figure 6.9: Number of classifications made per different type of reflection point, scenario set 2.

Figures 6.10 and 6.11 show the distributions of false and true classifications respectively across triplet categories and scenarios. For moving objects, scenario 10 and 11 are the worst performing, whilst for static objects, scenario 8 is the largest contributor to false classifications. However, scenario 8 is also overrepresented in regard to static objects in general.

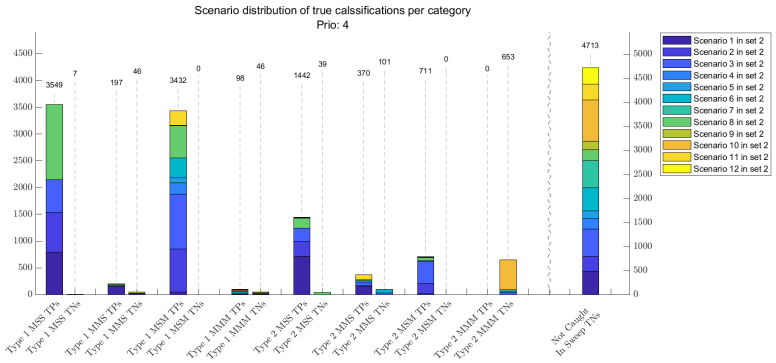


(a) Prio 4

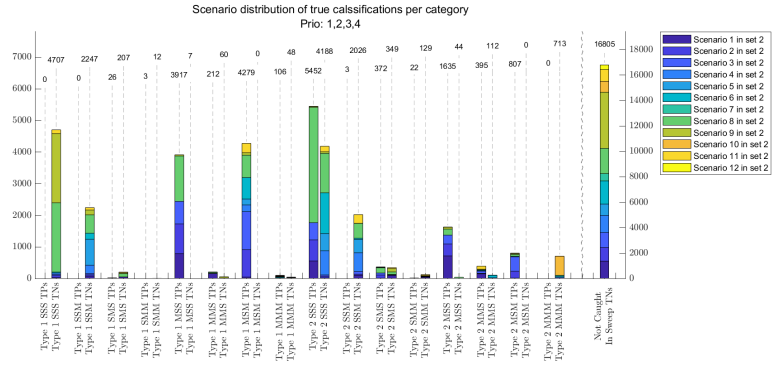


(b) Prio 1 through 4

Figure 6.10: Distribution of false classifications across categories and scenarios.



(a) Prio 4



(b) Prio 1 through 4

Figure 6.11: Distribution of true classifications across categories and scenarios.

6.3.2 Complexity

Table 6.3 shows the variables used in the following complexity derivation. In the grid population scheme, only one static detection and one predicted detection is allowed per cell. For moving detections only one detection per object is allowed per cell.

Table 6.3: Variables used in the complexity analysis of the grid-based multipath detection algorithm.

Variable	Used value	Comment
M_{az}	40	Maximum number of azimuth bins
M_r	55	Maximum number of range bins
M_g	1690	Total number of grid cells
N_o	200	Maximum number of objects
N_d	500	Maximum number of detections
M_p	4	Number of predictions for static detections
AZ_{sizes}	[5, 10, 20, 40]	Azimuth bin sizes
M_{tholds}	4	Number of azimuth thresholds
M_c	4	Maximum number of items in a grid cell
M_{LOS}	15	Maximum number of LOS points
M_{sweep}	214	Maximum number of cells in sweep

Under the assumption that the detections are evenly distributed spatially, there is at most N_o/M_g number of objects per grid cell. This implies that at most $2 + M_g/N_o$ detections can be contained within one cell. This allows for the approximation of $M_c = 2 + \lceil M_g/N_o \rceil$. Further, the LOS list is bounded to 15 elements.

From Figure 6.12 it can be seen that the algorithm is structured as three distinct steps:

1. Grid Population step: Algorithm 2 shows that populating the grid requires N_d number of iterations, one for each detection, plus M_p number of iterations for every static detection. Since M_p is constant, the overall complexity for the grid population is $\mathcal{O}(N_d)$.
2. LOS Processing: Algorithm 3 is a grid traversal of all cells, including a comparison between the current cell and the content of the LOS list. This can be reduced to M_g number of cell visits, with up to $M_c * M_{LOS}$ number comparisons, leading to a worst-case complexity of $\mathcal{O}(M_g * M_c * M_{LOS})$. Since $M_c * M_{LOS}$ is relatively small and bounded, the complexity can be simplified into $\mathcal{O}(M_g)$.
3. Triplet Identification and Classification: Algorithm 4 depends on N_o, M_{LOS} and M_{sweep}. The number of cells that must be visited varies depending on grid size, ghost object cell and current LOS point cell, but is limited by an upper bound due to the geometric constraints described in Section 4.2.1.5. In the worst case, the objects are positioned such that $N_o * M_{LOS} * N_o$ comparisons are required, resulting in $\mathcal{O}(N_o^2)$ complexity since M_{LOS} is bounded.

From this, it can be concluded that the PRED-RAG algorithm worst-case complexity is:

$$\mathcal{O}(N_d) + \mathcal{O}(M_g) + \mathcal{O}(N_o^2) = \mathcal{O}(N_d + M_g + N_o^2) \tag{6.2}$$

If M_g is kept constant during runtime, the worst-case complexity is reduced to:

$$\mathcal{O}(N_d + M_g + N_o^2) = \mathcal{O}(N_d + N_o^2) \tag{6.3}$$

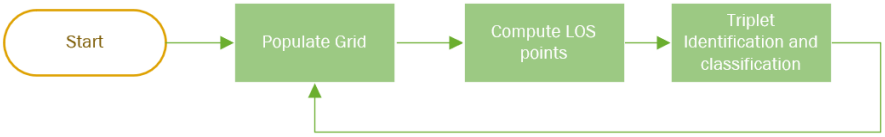


Figure 6.12: Process flow of the PRED-RAG algorithm.

6.3.3 Runtime

As shown in Table 6.2, the PRED-RAG algorithm outperforms the baseline when running on scenario set 2, whilst being slightly slower than the baseline on scenario set 1. For the PRED-RAG algorithm, the difference in runtime between the two scenario sets is lower compared to the baseline, indicating less dependency on input data.

In Figure 6.13 and 6.14 the mean scan runtime over the number of active objects in the scan is presented for the two sets of scenarios.

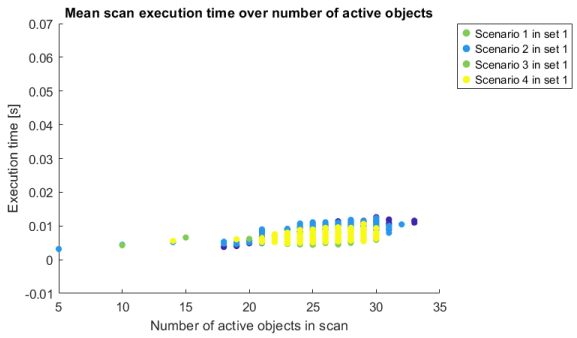


Figure 6.13: PRED-RAG algorithm mean runtime per scan over the number of currently active objects, scenario set 1.

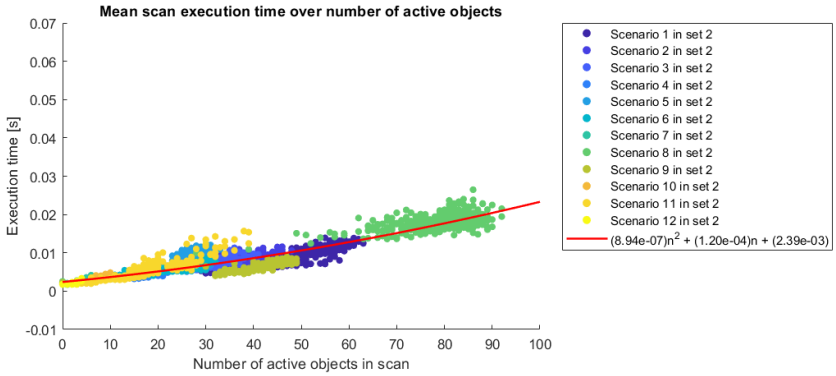
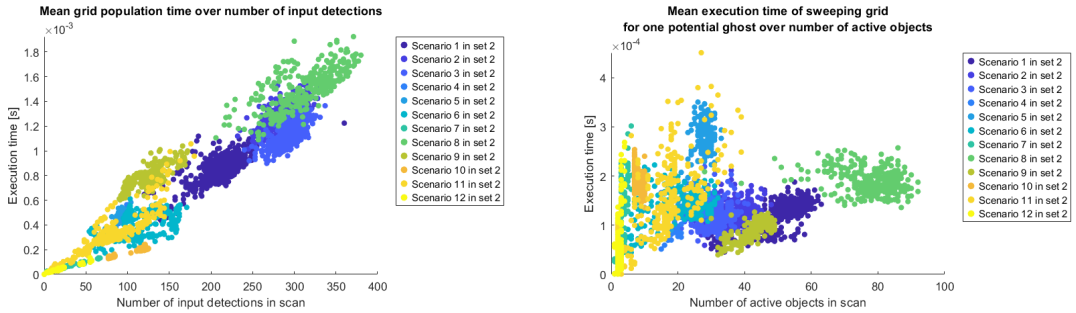


Figure 6.14: PRED-RAG algorithm mean runtime per scan over the number of currently active objects, scenario set 2.

As for the baseline, the runtime over scenario set 1 is clustered around the same area. Similarly, the quadratic complexity derived in Section 6.3.2 becomes evident in Figure 6.14, though the PRED-RAG algorithm exhibits significantly lower quadratic and linear coefficients in its measured runtime compared to the baseline. When comparing Figures 6.4, 6.13 and 6.14, the larger constant term in Equation 6.2, M_g, of the derived complexity for the PRED-RAG algorithm becomes apparent. Consequently, the minimum runtime remains higher than the baseline, which lacks large constant terms in its complexity.

Figure 6.15 shows the measured runtime for the grid population and triplet identification steps in the PRED-RAG algorithm process flow. The linear component in Equation 6.3, N_d, is clearly visible in Figure 6.15a. However, the quadratic component, N_o^2, cannot easily be distinguished in Figure 6.15b.



(a) Grid population step.

(b) Triplet identification step.

Figure 6.15: PRED-RAG algorithm mean runtime of process steps over the number of active objects in the scan, scenario set 2.

In Figure 6.16 the mean runtime for calculating the LOS points per scan is shown. As derived in Section 6.3.2, the worst-case complexity of the LOS calculation step is constant as the number of LOS points are bounded. Prior to reaching this upper bound, however, performance exhibits a clear dependency on the quantity of active objects and their spatial distribution throughout the grid. As illustrated in the figure, execution time initially increases with object count before gradually stabilizing as the number approaches higher values, demonstrating a convergence towards the worst-case constant complexity.

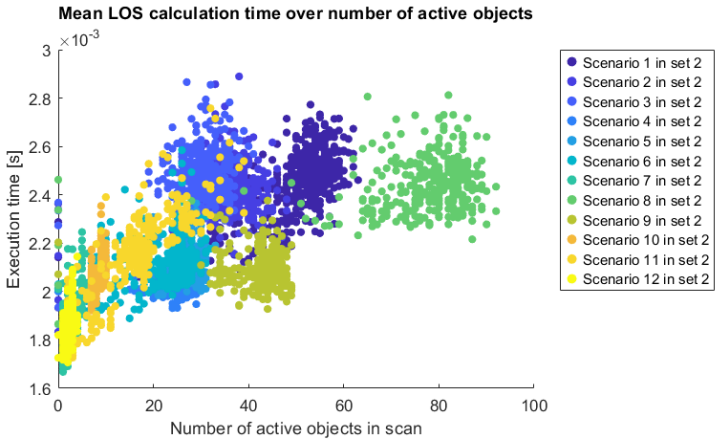


Figure 6.16: PRED-RAG algorithm mean runtime of LOS calculation per scan over the number of currently active objects, scenario set 2.

7

Conclusion

7.1 Discussion

While the PRED-RAG algorithm demonstrated promising results in the simulation environment, several important considerations must be addressed before deployment in real-world automotive applications. The controlled nature of simulation inevitably abstracts certain complexities inherent to actual radar data processing.

7.1.1 Simulated Detection Data

Radar waves generating multipath detections undergo multiple reflections and travel greater distances compared to those producing true detections. Consequently, multipath detections typically render lower signal amplitudes than true detections within the same range span, due to this increased signal attenuation. The detections produced by the Matlab simulation environment however, have no extractable information regarding the amplitude. This means that in a real-world application, many of the multipath detections would be filtered out in the thresholding stage described in Section 2.1.5. It is hence reasonable to assume that the proportion of multipath objects produced using the simulated detections, around 40%, is unreasonably large compared to what might be seen in the real world.

State-of-the-art radar systems likely incorporate sophisticated filtering algorithms that evaluate whether the DOA matches the DOD, as detailed in Section 3.1. If such filtering is indeed standard in modern automotive radars, then a majority of the type 1 multipath detections would be eliminated at the signal processing stage, before ever reaching the RMOT system.

The disproportionate representation of multipath objects in the dataset could potentially bias algorithm parameter optimization toward overclassifying objects as ghosts. However, since true objects still holds a majority, this imbalance is unlikely to be problematic as long as the algorithm maintains a low false positive rate. Rather than viewing this as a limitation, this distribution can be leveraged as an advantage—it effectively increases the test case density for multipath scenarios while keeping the amount of simulated scenario time at a reasonable level.

7.1.2 Matlab Driving Scenarios

The driving scenario generator is a great, easy to use tool for creating realistic traffic situations. However, as the simulator only provides models for road users, roads, and barriers, the level of realism is still limited. Real-world environments contain significantly greater complexity, featuring numerous off-road elements such as buildings, posts, vegetation, and other forms of infrastructure that contribute to radar reflections. Furthermore, the simulator's default models lack variation within each object class, whereas in the real world nothing is identical. This homogeneity in the simulations potentially limits the algorithm's exposure to the variability it would encounter in deployment.

Another data limitation arises from the velocity profiles used in the simulations, where most actors move at similar constant speeds with minimal acceleration and deceleration events. This considerably differs with real-world driving conditions, where vehicles typically maintain varied speeds and undergo frequent changes in acceleration as they navigate traffic. This lack of variability in the velocity data potentially limits the algorithms performance, specifically regarding the range-rate difference calculation. As vehicles in real-world traffic typically travel at varying speeds, a distinction should most likely be possible to make between a ghost object and a true object, given the velocity properties described in Section 2.2.2. Ghost objects will consistently follow these properties, whereas true objects would only coincidentally match the same characteristics. However, as the variability of the velocity profiles used in the simulations is very limited, the distinction between a ghost and a true object based on the range-rate difference becomes challenging. This can be seen when comparing Figure 6.10a with Figure 6.8a. For a majority of the categories, the best accuracy is achieved by classifying all evaluated triplets as either strictly true or strictly false. This is a clear indication that for this data, the range-rate criterion is insufficient.

Looking at the categories with the highest false classification count, type 2 MSM and type 2 MMM, scenarios 10 and 11 are the largest contributors. Scenario 10, described in Section A.2.13, is a straight road with multiple cars in close proximity and very low velocity variations. The bad performance in this scenario is expected given the problems described above. However, this is still a quite common scenario in the real world where the velocity differences will naturally be low.

Scenario 11, described in Section A.2.14, is a merging scenario where both targets in front of the host are driving at the same speed during the time that there is a guardrail separating the two. The velocity changes induced by the bend in this case do not help as the car opposite the guardrail mimics the movements of a ghost object spawned from the target in the host path. In the real world however, the two targets are more likely to have differing speeds which in theory should improve the performance of the PRED-RAG algorithm.

Looking at the scenario distribution over true classifications for the categories where the range-rate difference criterion is working, type 2 MSS and type 2 MMS seen in Figure 6.11a, scenarios 1, 2, 3 and 8 are the largest contributors. All these scenarios are set in corners or bends which induce continuous changes in the velocity vectors

of all the actors. This further indicates that the range-rate difference criterion is indeed dependent on variations in the velocity profiles of the actors.

7.1.3 PRED-RAG Algorithm Performance

The overall performance of the PRED-RAG algorithm is really good compared to the selected state-of-the-art baseline. As shown in Figure 6.8, the main performance factor by a margin is the geometric grid sweep. This outcome is expected since the geometric properties of multipath detections are less susceptible to errors than the velocity properties. This difference arises because radar systems directly measure target positions, whereas velocities can only be estimated through measurements taken over time. Using predictions to enhance the detection capabilities of static reflection surfaces also show promising results as seen in Figure 6.9.

The PRED-RAG algorithm demonstrates significantly lower execution time compared to the baseline algorithm. While both approaches face the inherent complexity of the multipath problem, the proposed solution achieves better measured performance despite maintaining the same worst-case quadratic complexity. As revealed by the comparison of Figures 6.4 and 6.14, the measured coefficients of both the quadratic and linear components in the PRED-RAG algorithm's complexity are considerably low relative to the baseline. This is expected since, as mentioned in Section 6.3.2, the number of cells that must be visited varies depending on initial cells used for the grid sweep and the positions of all other objects in the scene. To reach quadratic complexity even for one potential ghost object, every object not part of the initial ghost and LOS point pair must line up to match the geometric properties of multipath described in Section 2.2.1. This is highly unlikely, and that a whole scan would run at quadratic complexity is approximately impossible, as no objects in the track DB can change position during the triplet identification step to subsequently match the geometric properties for every combination of potential ghost and LOS reflection point in the scene.

7.1.4 Future Work

To further evaluate the performance of the PRED-RAG algorithm, testing on real-world data would be desirable. This would give a more accurate indication of how the algorithm handles the high complexity of the environment as well as how it is affected by noise in the data. By doing this, one can more accurately identify the strengths and weaknesses of the algorithm in real-world scenarios. Similarly, to increase the credibility of the algorithm runtime measurements, the evaluation should be done on more appropriate hardware than the equipment utilized in this study.

Beyond extending the evaluation methodology, further improvements exist for refining the algorithmic approach itself. Firstly, the current grid implementation adapts based on cell area, however, a systematic investigation into optimal grid configurations could yield substantial gains. Specifically, conducting a comprehensive parametric analysis of grid layouts, considering factors such as cell size and distribution based

on range and angle, could simultaneously improve both computational efficiency and solution accuracy. Similarly, investigating how one can, during runtime, adapt the grid size based on for example current detection range, host speed, or other indicators of the current driving environment could further enhance the effectiveness of the grid.

Secondly, the current method only utilizes one criterion based on the velocity of the triplet members. Investigating the possibility of using more velocity-based criteria, derived from modeling how the velocity vectors are influenced over time based on the different types of multipath influences, could enhance the confidence in the calculated probability of whether an object is a ghost or not.

Finally, with real-world data as input, the algorithm will most likely have access to other measured detection parameters such as signal amplitude and estimated target RCS. Incorporating such measurements in to the calculated probability of an object could also increase the classification accuracy of the algorithm.

7.2 Summary

This work proposes an algorithm for identifying ghost objects created from multipath radar detections. The results presented in the thesis show that the PRED-RAG algorithm clearly outperforms the state-of-the-art baseline both in regard to accuracy and computational complexity when evaluated on the simulated datasets. Using host motion predictions to project static detections into future time steps successfully enhance the classification capabilities of the algorithm whilst keeping the computational complexity low. Utilizing a grid to cluster detections and query objects based on spatial properties further reduces the processing demand, limiting the inherent quadratic worst-case complexity of the multipath problem. The proposed approach also show significant generalization capabilities compared to the state of the art as no strict assumptions are made on the members of a multipath triplet.

Bibliography

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020. DOI: 10.1109/ACCESS.2020.2983149.

[2] C. Waldschmidt, J. Hasch, and W. Menzel, "Automotive radar — from first efforts to future systems," *IEEE Journal of Microwaves*, vol. 1, no. 1, pp. 135–148, 2021. DOI: 10.1109/JMW.2020.3033616.

[3] S. Sun, A. P. Petropulu, and H. V. Poor, "Mimo radar for advanced driver-assistance systems and autonomous driving: Advantages and challenges," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 98–117, 2020. DOI: 10.1109/MSP.2020.2978507.

[4] A. Nüsslein, "Multipath modelling for 60 ghz short range radar," eng, Bachelor Thesis, Lund University, Lund, Sweden, 2021.

[5] S. Baratam, *Limiting radar disturbances and artefacts in tunnels - the effect on adas and autonomous functions*, Webinar, Accessed: 2024-12-05, Nov. 2024. [Online]. Available: https://lnkd.in/ddt-ZC64.

[6] R. Feng, E. D. Greef, M. Rykunov, H. Sahli, S. Pollin, and A. Bourdoux, "Multipath ghost recognition for indoor mimo radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–10, 2022. DOI: 10.1109/TGRS.2021.3109381.

[7] J. Li and P. Stoica, *MIMO Radar Signal Processing*. John Wiley & Sons, Jan. 2008, ISBN: 0-470-39148-0. DOI: 10.1002/9780470391488. [Online]. Available: https://doi.org/10.1002/9780470391488.

[8] H. Abdullah, M. Mabrouk, A. Abd-Elnaby Kabeel, and A. Hussein, "High-resolution and large-detection-range virtual antenna array for automotive radar applications," *Sensors*, vol. 21, no. 5, 2021, ISSN: 1424-8220. DOI: 10.3390/s21051702. [Online]. Available: https://www.mdpi.com/1424-8220/21/5/1702.

[9] M. Kronauge and H. Rohling, "New chirp sequence radar waveform," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 4, pp. 2870–2877, 2014. DOI: 10.1109/TAES.2014.120813.

[10] Richards, M. A., Scheer, J. A., Holm, and W. A., *Principles of Modern Radar, Volume I - Basic Principles*. SciTech Publishing, 2010, ISBN: 978-1-891121-52-4. [Online]. Available: https://app.knovel.com/hotlink/toc/id:kpPMRVIBP8/principles-modern-radar/principles-modern-radar.

[11] J. M. Yang and W. W. Kim, "Performance analysis of a minimum selected cell-averaging cfar detection," in *2008 11th IEEE International Conference*

on Communication Technology, 2008, pp. 442–445. DOI: 10.1109/ICCT.2008.4716288.

[12] S. Watts, "The performance of cell-averaging cfar systems in sea clutter," in *Record of the IEEE 2000 International Radar Conference [Cat. No. 00CH37037]*, 2000, pp. 398–403. DOI: 10.1109/RADAR.2000.851867.

[13] M. Schoor and Y. Bin, *High-resolution angle estimation for an automotive fmcw radar sensor*, Jan. 2007.

[14] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986. DOI: 10.1109/TAP.1986.1143830.

[15] R. Roy and T. Kailath, "Esprit-estimation of signal parameters via rotational invariance techniques," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984–995, 1989. DOI: 10.1109/29.32276.

[16] H. Singh and A. Chattopadhyay, "Multi-target range and angle detection for mimo-fmcw radar with limited antennas," in *2023 31st European Signal Processing Conference (EUSIPCO)*, 2023, pp. 725–729. DOI: 10.23919/EUSIPCO58844.2023.10289869.

[17] C. Liu, S. Liu, C. Zhang, Y. Huang, and H. Wang, "Multipath propagation analysis and ghost target removal for fmcw automotive radars," in *IET International Radar Conference (IET IRC 2020)*, vol. 2020, 2020, pp. 330–334. DOI: 10.1049/icp.2021.0554.

[18] L. Wang, S. Giebenhain, C. Anklam, and B. Goldluecke, "Radar ghost target detection via multimodal transformers," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7758–7765, 2021. DOI: 10.1109/LRA.2021.3100176.

[19] O. Longman, S. Villeval, and I. Bilik, "Multipath ghost targets mitigation in automotive environments," in *2021 IEEE Radar Conference (RadarConf21)*, 2021, pp. 1–5. DOI: 10.1109/RadarConf2147009.2021.9455253.

[20] NATO Science and Technology Organization, "Bistatic radar: Principles and practice," NATO Science and Technology Organization, Tech. Rep. RTO-EN-SET-086, 2006. [Online]. Available: https://www.sto.nato.int/publications/STO%20Educational%20Notes/RTO-EN-SET-086/EN-SET-086-04.pdf.

[21] K. Thormann, M. Baum, and J. Honer, "Extended target tracking using gaussian processes with high-resolution automotive radar," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 1764–1770. DOI: 10.23919/ICIF.2018.8455630.

[22] A. Manjunath, Y. Liu, B. Henriques, and A. Engstle, "Radar based object detection and tracking for autonomous driving," in *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, 2018, pp. 1–4. DOI: 10.1109/ICMIM.2018.8443497.

[23] L. Hammarstrand, L. Svensson, F. Sandblom, and J. Sorstedt, "Extended object tracking using a radar resolution model," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2371–2386, 2012. DOI: 10.1109/TAES.2012.6237597.

[24] S. Lim, S. Lee, and S.-C. Kim, "Clustering of detected targets using dbscan in automotive radar systems," in *2018 19th International Radar Symposium (IRS)*, 2018, pp. 1–7. DOI: 10.23919/IRS.2018.8448228.

[25] K. Khan, S. Rehman, K. Aziz, S. Fong, S. Sarasvady, and A. Vishwa, "Dbscan: Past, present and future," *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, pp. 232–238, 2014. DOI: 10.1109/ICADIWT.2014.6814687.

[26] S. Särkkä, *Bayesian Filtering and Smoothing* (Institute of Mathematical Statistics Textbooks). Cambridge University Press, 2013.

[27] M. Alibeigi et al., *Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving*, 2023. arXiv: 2305.02008 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2305.02008.

[28] H. Caesar et al., *Nuscenes: A multimodal dataset for autonomous driving*, 2020. arXiv: 1903.11027 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1903.11027.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[30] M. Liu et al., "A survey on autonomous driving datasets: Statistics, annotation quality, and a future outlook," *IEEE Transactions on Intelligent Vehicles*, pp. 1–29, 2024. DOI: 10.1109/TIV.2024.3394735.

[31] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering 4th Edition*. Boca Raton, FL, USA: A K Peters/CRC Press, 2018, p. 1200, ISBN: 978-1-13862-700-0.

[32] H. Samet, *Spatial data structures*. 1995.

[33] Y. Li and X. Shang, "Multipath ghost target identification for automotive mimo radar," in *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, 2022, pp. 1–5. DOI: 10.1109/VTC2022-Fall57202.2022.10012904.

[34] L. Zheng, J. Long, M. Lops, F. Liu, X. Hu, and C. Zhao, "Detection of ghost targets for automotive radar in the presence of multipath," *IEEE Transactions on Signal Processing*, vol. 72, pp. 2204–2220, 2024. DOI: 10.1109/TSP.2024.3384750.

[35] F. Roos, M. Sadeghi, J. Bechter, N. Appenrodt, J. Dickmann, and C. Waldschmidt, "Ghost target identification by analysis of the doppler distribution in automotive scenarios," in *2017 18th International Radar Symposium (IRS)*, 2017, pp. 1–9. DOI: 10.23919/IRS.2017.8008128.

[36] T. Grebner, F. Konrad, D. Schwarz, and C. Waldschmidt, "Detection and backprojection of ghost targets within a network of radar sensors," in *2023 20th European Radar Conference (EuRAD)*, 2023, pp. 43–46. DOI: 10.23919/EuRAD58043.2023.10289553.

[37] O. Longman, S. Villeval, and I. Bilik, *Multipath ghost mitigation in vehicle radar system*, Feb. 2023.

[38] S. Mohammed et al., *The effects of data quality on machine learning performance*, 2024. arXiv: 2207.14529 [cs.DB]. [Online]. Available: https://arxiv.org/abs/2207.14529.

[39] S. Trelsmo and F. Alhayek, "Radar ghost detection and elimination," Examensarbete för masterexamen, M.S. thesis, Chalmers, Göteborg, Sweden, Aug. 2022. [Online]. Available: https://hdl.handle.net/20.500.12380/305301.

[40] S. Danino and I. Bilik, "Automated labeling of automotive radar azimuth multipath," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 7630–7634. DOI: 10.1109/ICASSP48485.2024.10446232.

[41] Y. Chen, M. Yang, X. Wang, and B. Chen, "Multipath model and ghost suppression for millimeter wave traffic radar," in *2024 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, 2024, pp. 1–5. DOI: 10.1109/ICSIDP62679.2024.10868312.

[42] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, 2017. arXiv: 1612.00593 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1612.00593.

[43] R. Feng, E. De Greef, M. Rykunov, H. Sahli, S. Pollin, and A. Bourdoux, "Multipath ghost classification for mimo radar using deep neural networks," in *2022 IEEE Radar Conference (RadarConf22)*, 2022, pp. 1–6. DOI: 10.1109/RadarConf2248738.2022.9764274.

[44] MathWorks, *Highway vehicle tracking with multipath radar reflections*, Accessed: 2024-12-05, 2024. [Online]. Available: https://se.mathworks.com/help/driving/ug/multipath-radar-detection-and-tracking.html.

[45] Mathworks. "Driving scenario designer," Accessed: May 5, 2025. [Online]. Available: https://www.mathworks.com/help/driving/ref/drivingscenariodesigner-app.html.

[46] Mathworks. "Radardatagenerator," Accessed: May 5, 2025. [Online]. Available: https://www.mathworks.com/help/radar/ref/radardatagenerator.html.

[47] J. L. Bentley and J. H. Friedman, "Data structures for range searching," *ACM Comput. Surv.*, vol. 11, no. 4, pp. 397–409, Dec. 1979, ISSN: 0360-0300. DOI: 10.1145/356789.356797. [Online]. Available: https://doi.org/10.1145/356789.356797.

[48] R. Krishnan and S. Bhada, "Integrated system design and safety framework for model-based safety assessment," *IEEE Access*, vol. 10, pp. 1–1, Jan. 2022. DOI: 10.1109/ACCESS.2022.3193495.

[49] Trafikverket, "Vägars och gators utformning," Trafikverket, Tech. Rep. TRVINFRA-00396, Nov. 2024, Version 1.0.

[50] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Commun. ACM*, vol. 23, no. 11, pp. 620–624, Nov. 1980, ISSN: 0001-0782. DOI: 10.1145/359024.359026. [Online]. Available: https://doi.org/10.1145/359024.359026.

A

Appendix

A.1 Scenario Sets

Table A.1: Scenario set 1.

#	**Scenario**	**Number of Scans**	**Reference Section**
1	Highway 1 Target Lane Change 1	189	A.2.5
2	Highway 1 Target Lane Change 2	217	A.2.6
3	Highway 1 Target Long	787	A.2.7
4	Highway 1 Target Curvy	787	A.2.10

Table A.2: Scenario set 2.

#	**Scenario**	**Number of Scans**	**Reference Section**
1	Tight Corner 1 Target	480	A.2.1
2	Sweeping Bend 1 Target	340	A.2.2
3	Sweeping Bend 2 Targets	340	A.2.3
4	Highway 1 Target	218	A.2.4
5	Highway 1 Target Lane Change 1	189	A.2.5
6	Highway Multiple Targets	297	A.2.8
7	Highway No Guardrail	297	A.2.9
8	Highway 1 Target Curvy Overtake	320	A.2.11
9	Junction Targets All Directions	276	A.2.12
10	Low Speed Queue	196	A.2.13
11	Highway 1 Target Merge	321	A.2.14
12	Rural Road Multiple Targets	191	A.2.15

A.2 Scenarios

A.2.1 Tight Corner 1 Target

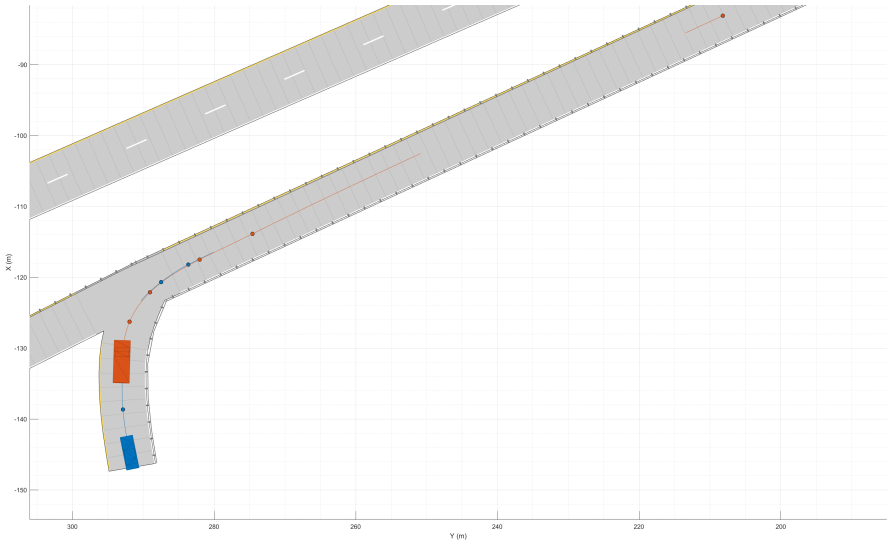


Figure A.1: Scenario Tight Corner 1 target: Host vehicle is driving behind a truck starting at a tight curve onto a highway ramp. Both host vehicle and truck is driving at 4 m/s. On both sides of the road there are guardrails, which cause multipath detections to appear on both sides of the truck.

A.2.2 Sweeping Bend 1 Target

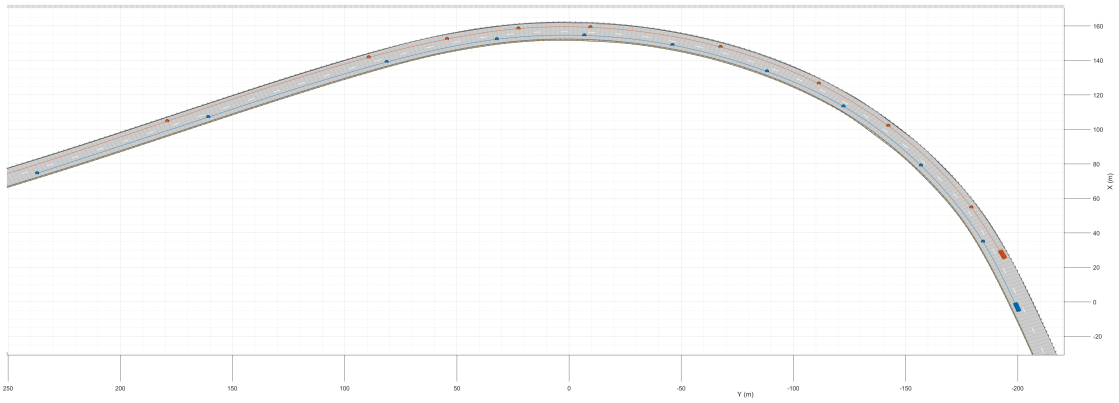


Figure A.2: Scenario Sweeping Bend 1 target: Host vehicle (blue) is driving in the left lane while a car (red) is driving in the right lane 25 m ahead. The road turns slightly left and both vehicles are driving at 30 m/s. On both sides of the road there are guardrails, which cause multipath detections to appear on both sides of the car.

A.2.3 Sweeping Bend 2 Targets

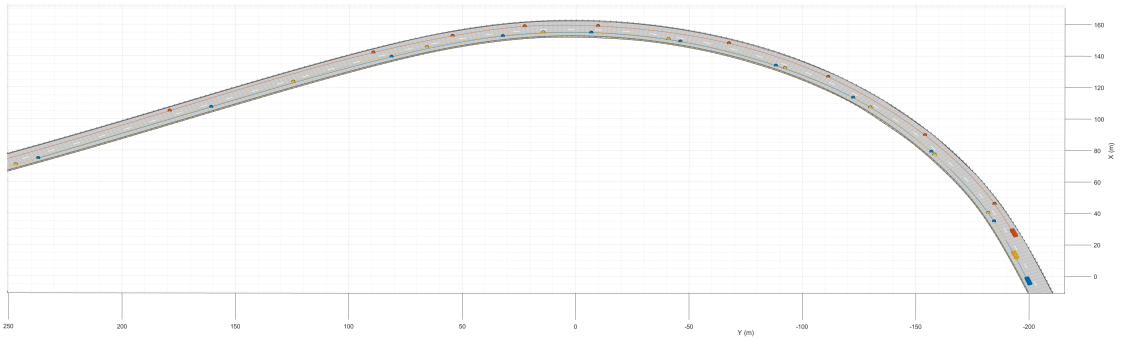


Figure A.3: Scenario Sweeping Bend 2 target: Host vehicle (blue) is driving in the left lane 15 m behind a car (yellow). In the right lane another car (red) is driving 25 m ahead. The road turns slightly left and all vehicles is driving at 30 m/s. On both sides of the road there are guardrails, which cause multipath detections to appear on both sides of the cars.

A.2.4 Highway 1 Target

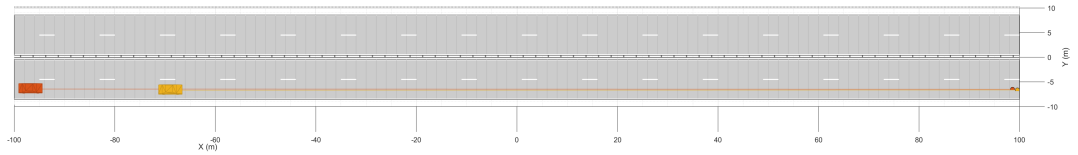


Figure A.4: Scenario Highway 1 target: Host vehicle (blue) is driving 30 m behind a car (yellow), both are traveling at 15 m/s. On the side of the left lane is a guardrail, which cause multipath detections to appear to the left of the guardrail.

A.2.5 Highway 1 Target Lane Change 1

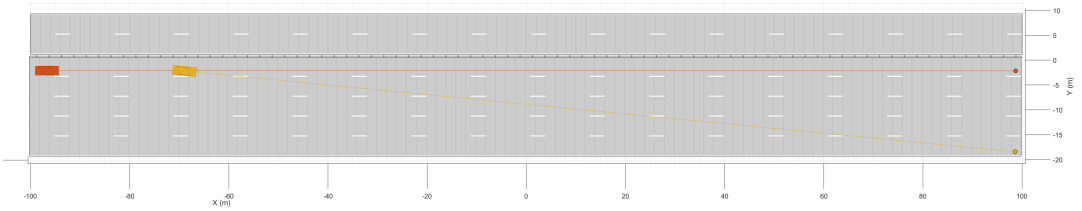


Figure A.5: Scenario Highway 1 target Lane Change 1: Host vehicle (red) is driving 30 m behind a car (yellow) which is changing from the leftmost to the rightmost lane on a five lane highway, both are traveling at 15 m/s. On the side of the leftmost lane is a guardrail, which cause multipath detections to appear to the left of the the guardrail.

A.2.6 Highway 1 Target Lane Change 2

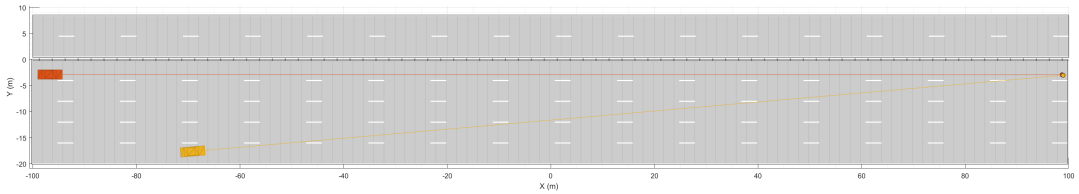


Figure A.6: Scenario Highway 1 target Lane Change 2: Host vehicle (red) is driving 30 m behind a car (yellow) which is changing from the rightmost to the leftmost lane on a five lane highway, both are traveling at 15 m/s. On the side of the leftmost lane is a guardrail, which cause multipath detections to appear to the left of the the guardrail.

A.2.7 Highway 1 Target Long

Exactly the same layout as Highway 1 Target described in Section A.2.4, but 2000 meters long instead of 200 meters.

A.2.8 Highway Multiple Targets

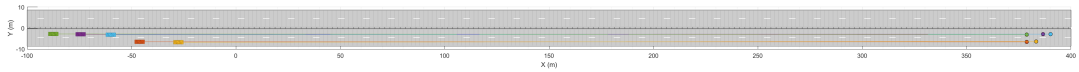


Figure A.7: Scenario Highway Multiple Targets: Host vehicle (red) is driving 30 meters behind a car (yellow) in the right lane. Three cars (green, purple and blue) are overtaking in the left lane at a speed of 30 m/s meanwhile host and yellow car is traveling at 20 m/s. On the side of the left lane is a guardrail, which cause multipath detections to appear to the left of the the guardrail.

A.2.9 Highway No Guardrail

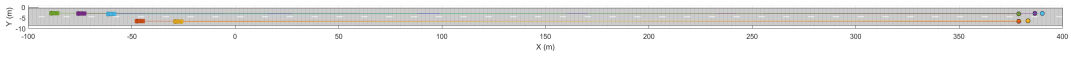


Figure A.8: Scenario Highway No Guardrail: Host vehicle (red) is driving 30 meters behind a car (yellow) in the right lane. Three cars (green, purple and blue) are overtaking in the left lane at a speed of 30 m/s meanwhile host and yellow car is traveling at 20 m/s. There are no guardrails on the side of the road.

A.2.10 Highway 1 Target Curvy

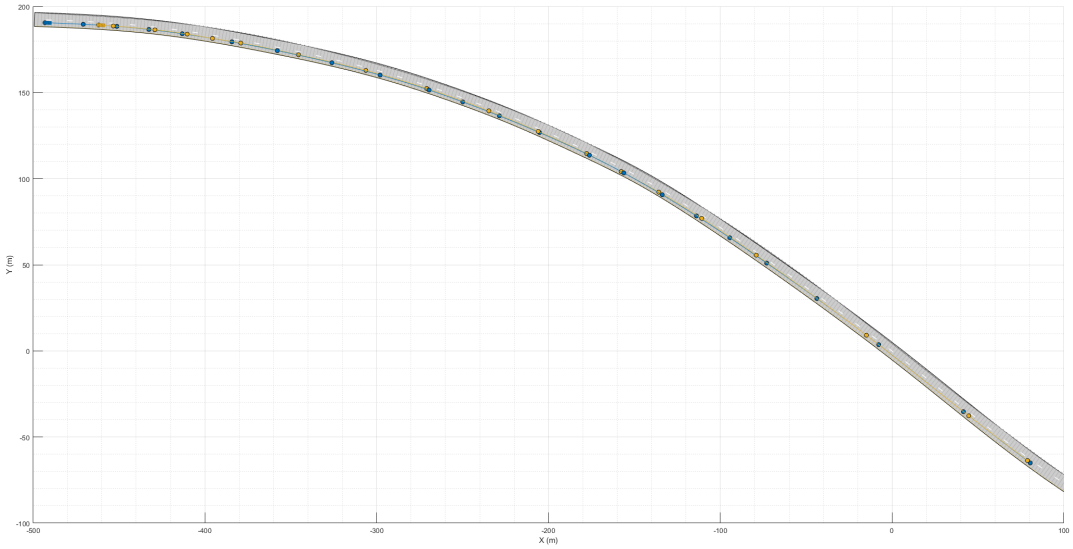


Figure A.9: Scenario Highway 1 Target Overtake: Host vehicle (blue) is driving 30 meters behind car (yellow) in the right lane. Both vehicles are driving at 15 m/s. There is a guardrail on the left side of the road, causing multipath detections to appear to the left of the guardrail.

A.2.11 Highway 1 Target Curvy Overtake

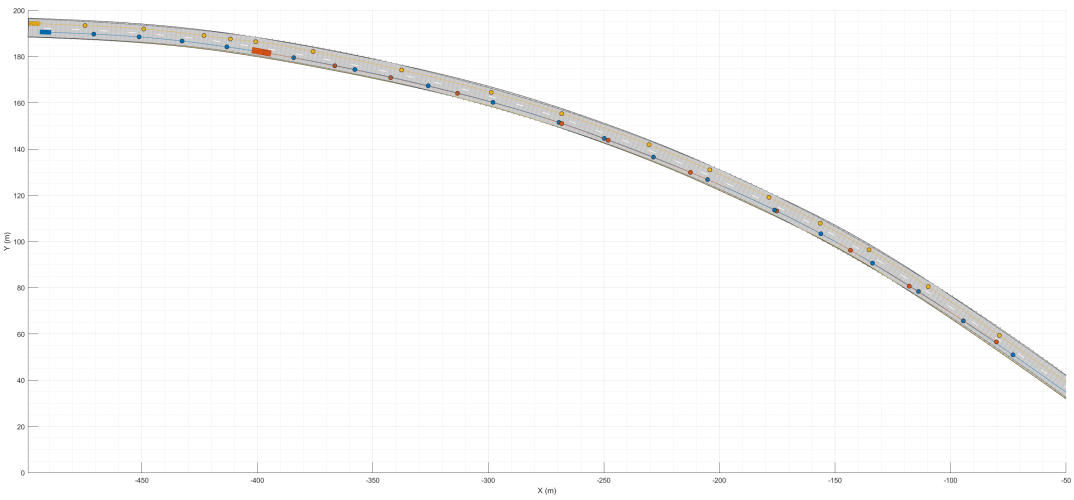


Figure A.10: Scenario Highway 1 Target Overtake: Host vehicle (blue) is driving 70 meters behind a truck (red) in the right lane. A car (yellow) is overtaking in the left lane at 30 m/s. Host is driving at 24 m/s and the truck initially drives at 20 m/s and later increases to 24 m/s. There are guardrails on both sides of the road causing multipath detections to appear on both sides of the road.

A.2.12 Junction Targets All Directions

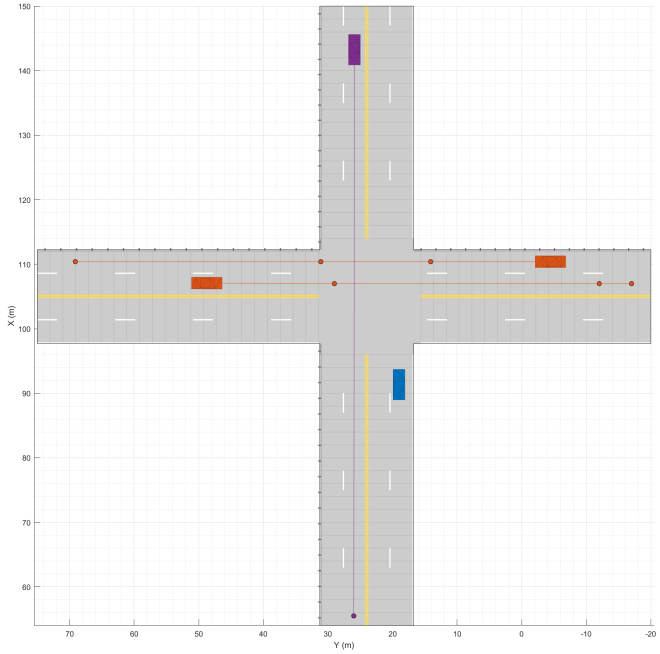


Figure A.11: Scenario Junction Targets All Directions: Host vehicle (blue) is standing still at a four way junction where two cars (red) are passing through the junction one at a time traveling 20 m/s. In the opposite lane of host a car (purple) is passing through the junction at 20 m/s after both red cars have passed. There are guardrails on the left side of host, right side of purple and to the right of the upper red car which cause complex multipath detection patterns.

A.2.13 Low Speed Queue

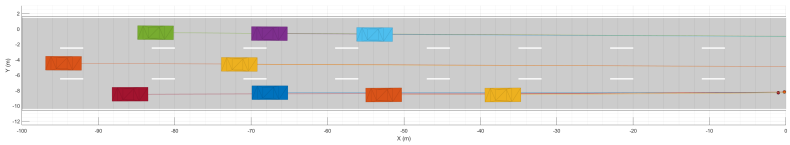


Figure A.12: Scenario Low Speed Queue: Host is driving in the middle lane behind a car (yellow), both traveling at 4 m/s. In the left lane three cars (green, purple, blue) are also driving at 4 m/s. In the right lane, four cars are traveling at 3 m/s. This scenario aims to simulate a typical low speed queue situation, where multipath detections occur from reflections between the vehicles.

A.2.14 Highway 1 Target Merge

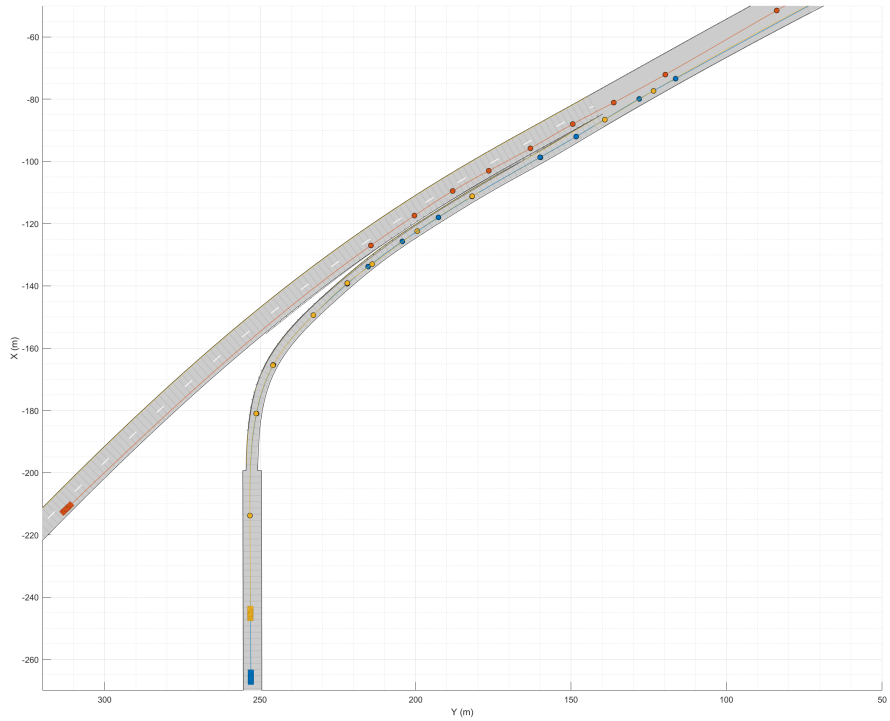


Figure A.13: Scenario Highway 1 Target Merge: Host (blue) is driving behind a car (yellow) on a highway merging ramp. When the ramp merges with the highway, another car (red) is positioned to the side of the yellow car. On the ramp there is a guardrail to the right, which causes ghost objects to the left of the guardrail. All vehicles are driving at 15 m/s.

A.2.15 Rural Road Multiple Targets

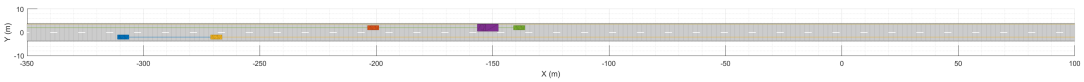


Figure A.14: Scenario Rural Road Multi Targets: Host (blue) is driving behind a car (yellow) on a two lane road where two cars (red and green) and a truck (purple) passes in the opposite direction on the left. The host, yellow car, red car and truck are driving at 10 m/s, meanwhile the green car is driving at 8 m/s. In this scenario, multipath detections appear due to reflections between the vehicles.

A.3 Baseline Statistics and Results

Table A.3: Criteria thresholds tuned for scenario set 1.

Criteria	**Threshold**	**Comment**
Range Diff	2.1 m	-
ANG	0.91 rad	-
MSD	0.04 m	-
PER	0.3	For threshold 1.6 rad
MSE_v	600 m/s	-
PER_v	0.3	For threshold 600 m/s
MSE_{vm}	0.2 m/s	-
MSE_{va}	0.02 rad	-
DRV	0.73 m	-
IANG	min(1.17, -0.182*range + 6.3) rad	Range dependent threshold
PER_i	0.3	For threshold 1.17

Table A.4: Criteria thresholds tuned for scenario set 2.

Criteria	**Threshold**	**Comment**
Range Diff	10 m	-
ANG	0.91 rad	-
MSD	0.04 m	-
PER	0.3	For threshold 1.6 rad
MSE_v	255 m/s	-
PER_v	0.3	For threshold 255 m/s
MSE_{vm}	20 m/s	-
MSE_{va}	2.5 rad	-
DRV	0.73 m	-
IANG	-0.182*range + 6.3 rad	Range dependent threshold
PER_i	0.3	For threshold 1.17

A.3.1 Baseline Scenario Set 1 Statistics

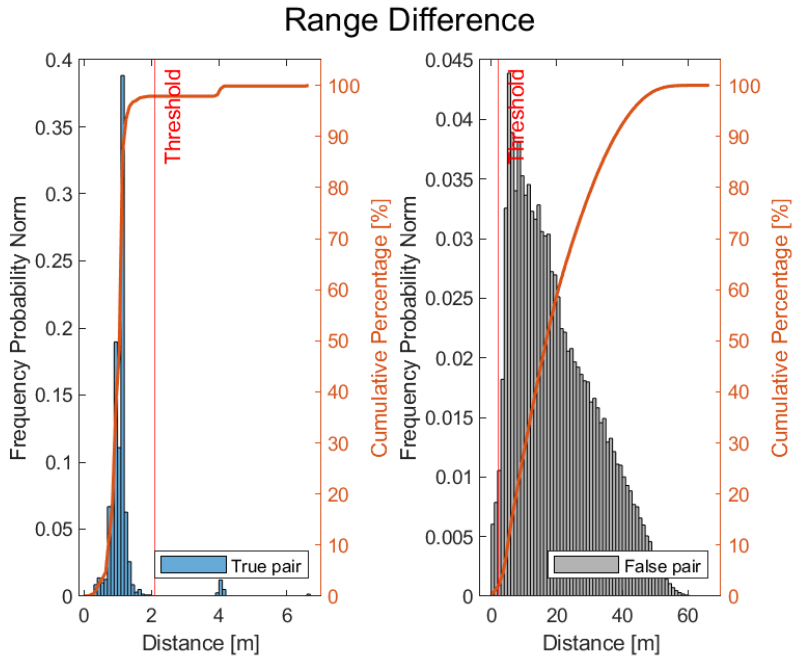


Figure A.15: Statistics and threshold for in scope data from scenario set 1, all priority levels. Pair range difference.

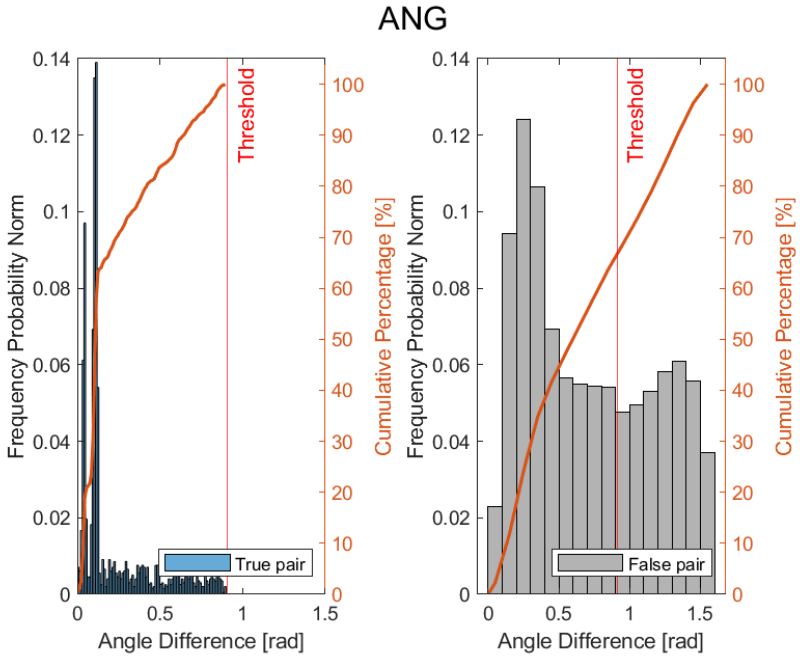


Figure A.16: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria ANG.

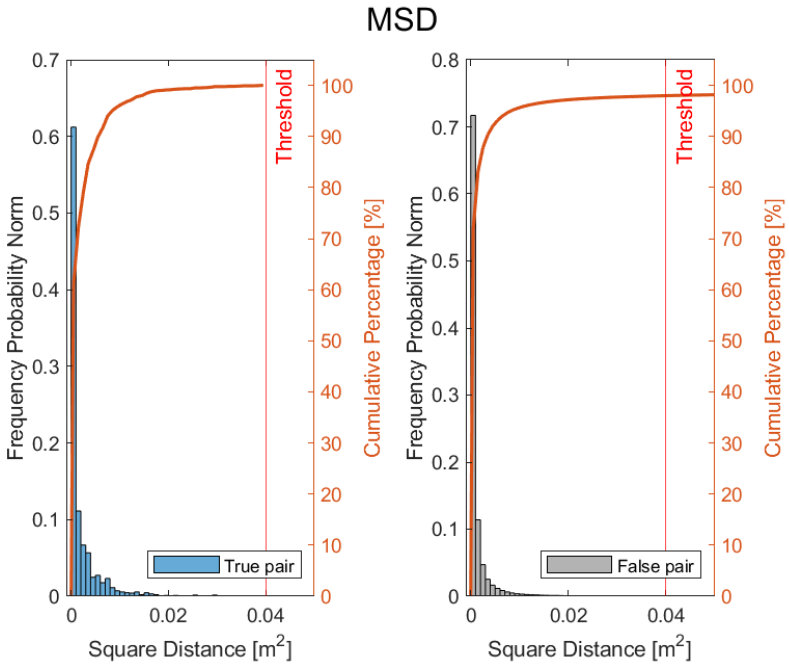


Figure A.17: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria MSD.

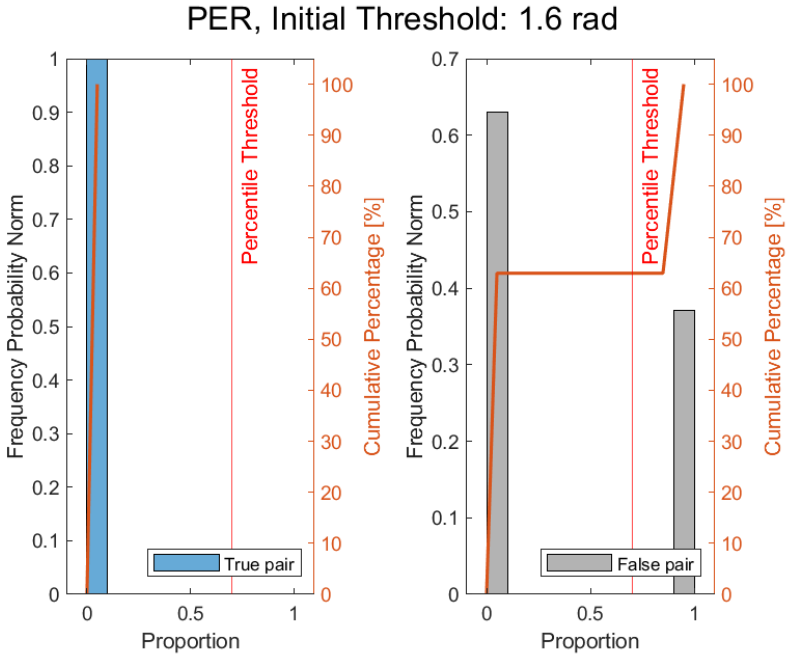


Figure A.18: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria PER.

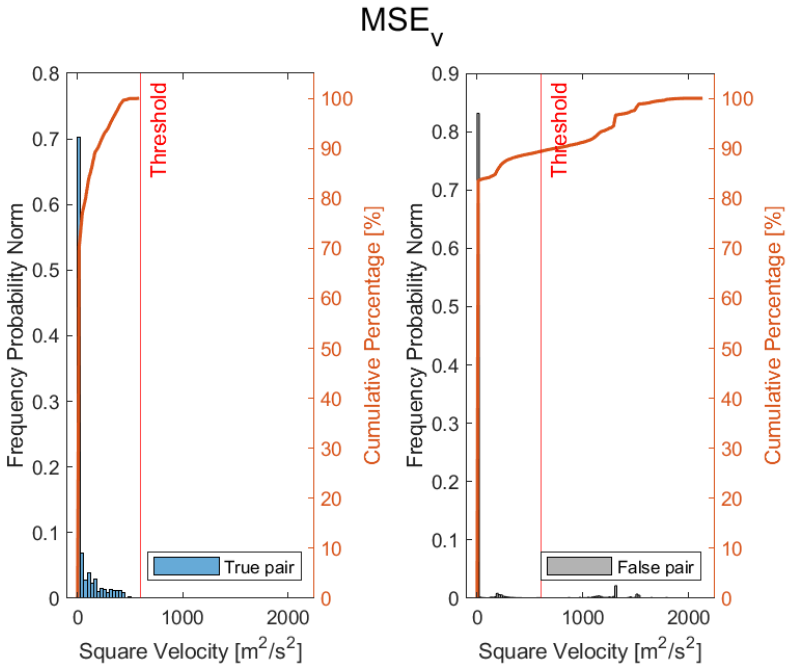


Figure A.19: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria MSE_v.

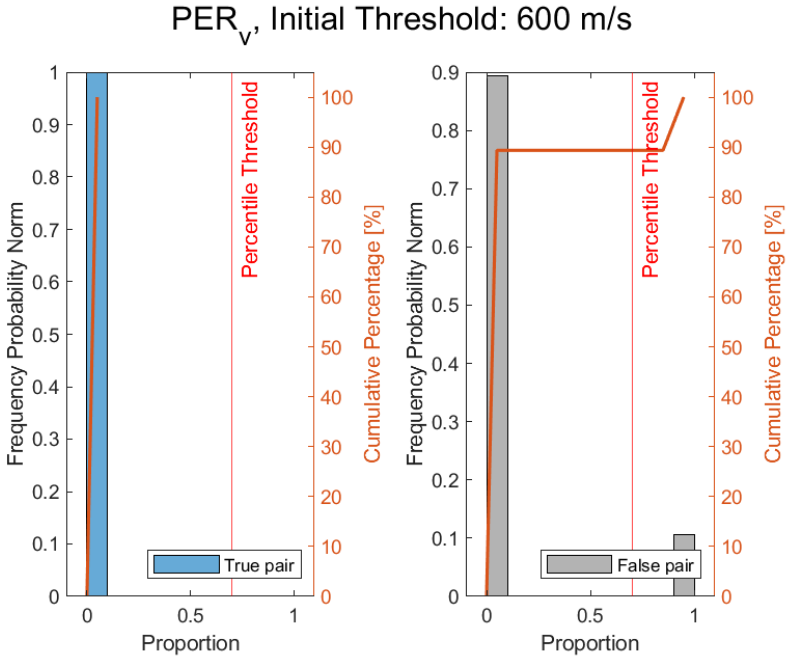


Figure A.20: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria PER_v.

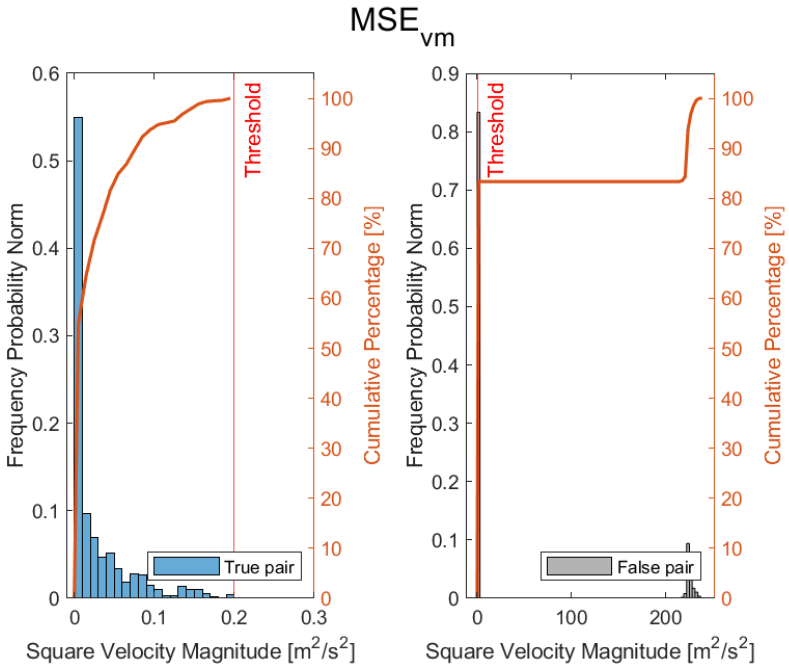


Figure A.21: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria MSE_{vm}.

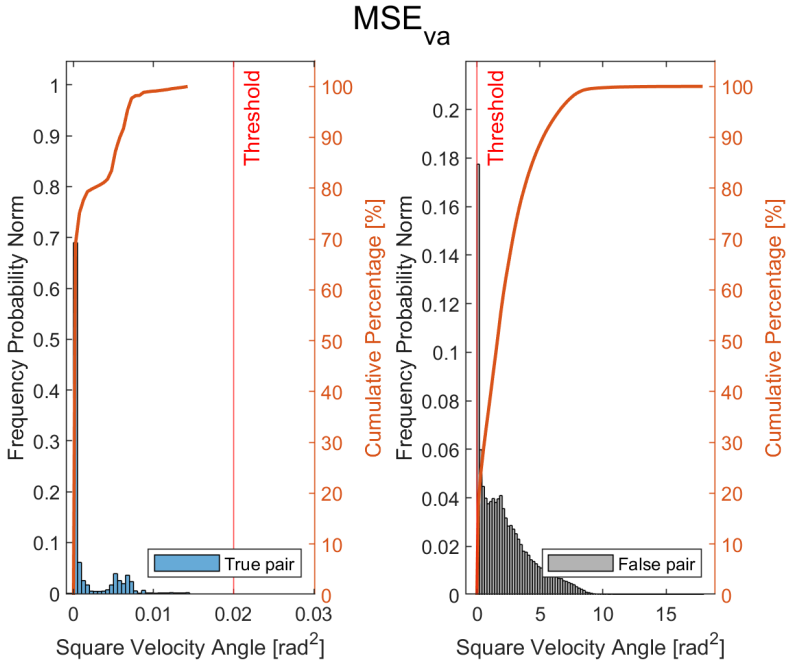


Figure A.22: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria MSE_{va}.

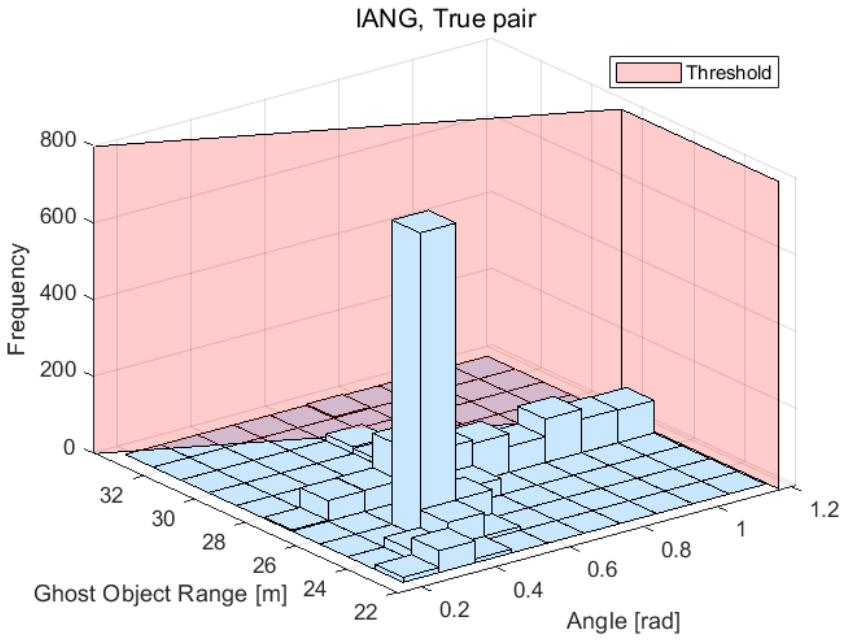


Figure A.23: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria IANG, true pairs.

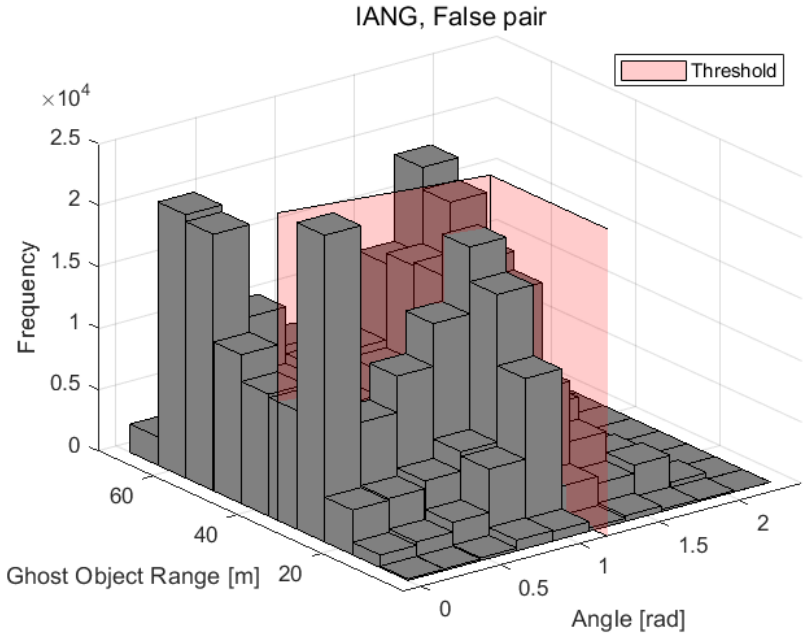


Figure A.24: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria IANG, false pairs.

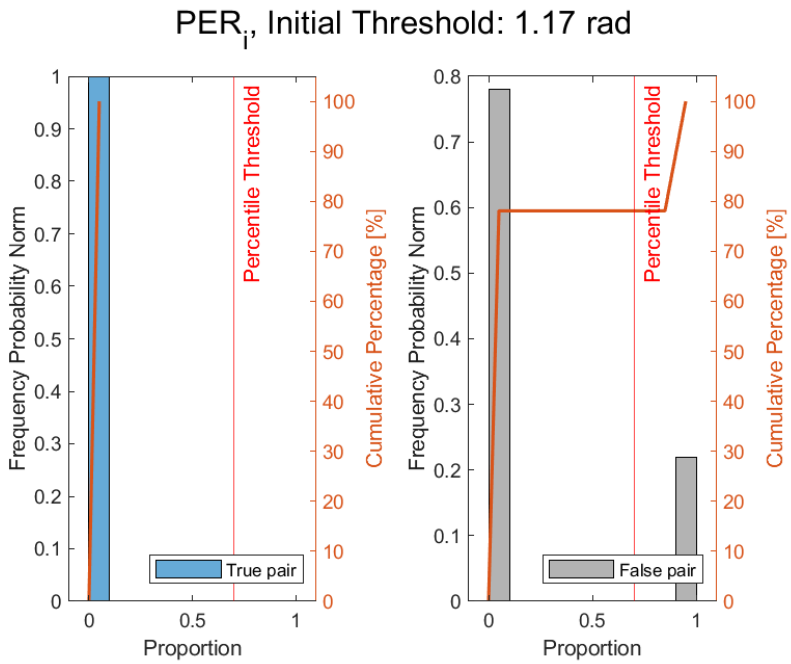
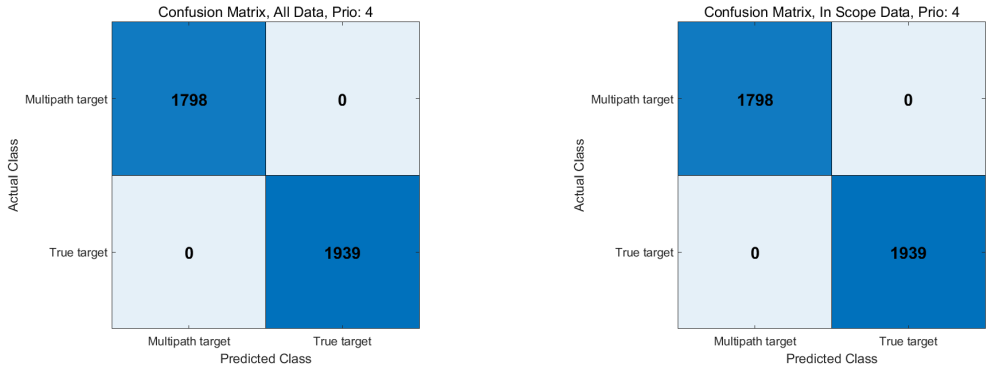


Figure A.25: Statistics and threshold for in scope data from scenario set 1, all priority levels. Criteria PER_i.

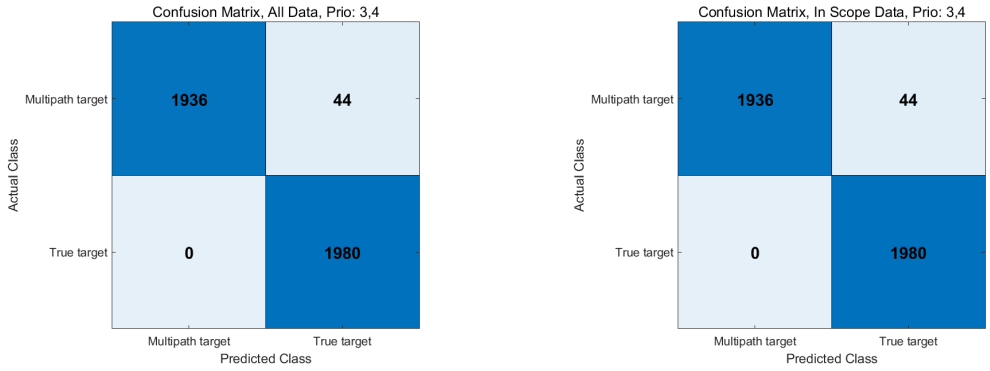
A.3.2 Baseline Scenario Set 1 Results



(a) Performance over all data

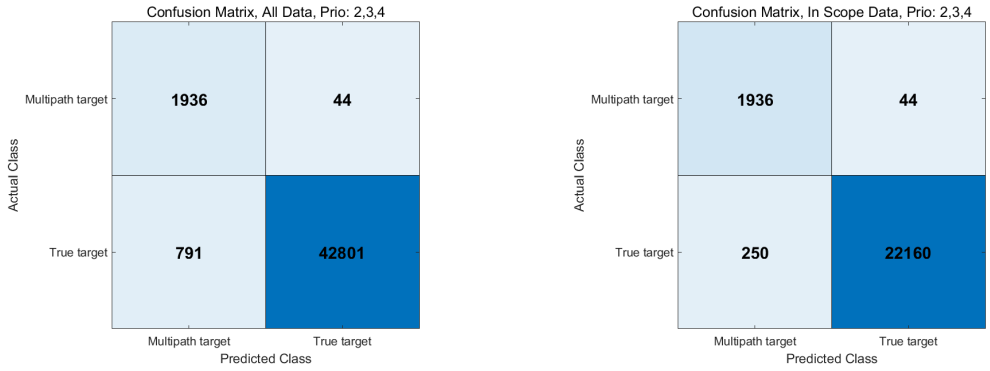
(b) Performance over in scope data

Figure A.26: Performance of state of the art algorithm on scenario set 1, priority level 4.



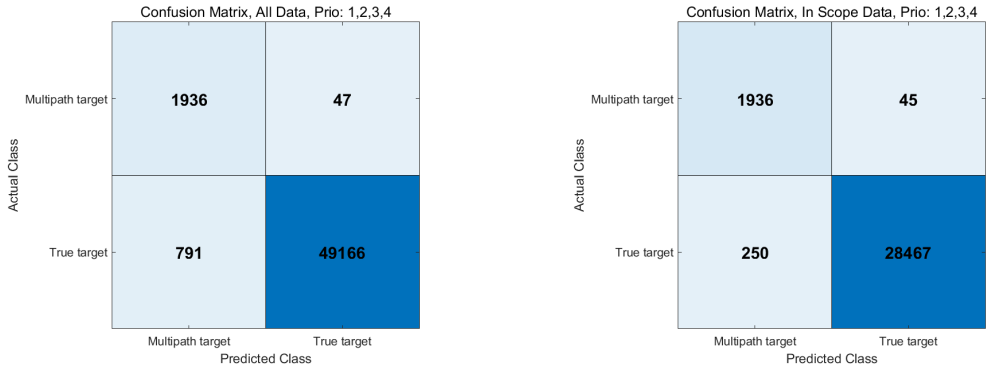
(a) Performance over all data (b) Performance over in scope data

Figure A.27: Performance of state of the art algorithm on scenario set 1, priority levels 3 through 4.



(a) Performance over all data (b) Performance over in scope data

Figure A.28: Performance of state of the art algorithm on scenario set 1, priority levels 2 through 4.



(a) Performance over all data (b) Performance over in scope data

Figure A.29: Performance of state of the art algorithm on scenario set 1, priority levels 1 through 4.

A.3.3 Baseline Scenario Set 2 Statistics

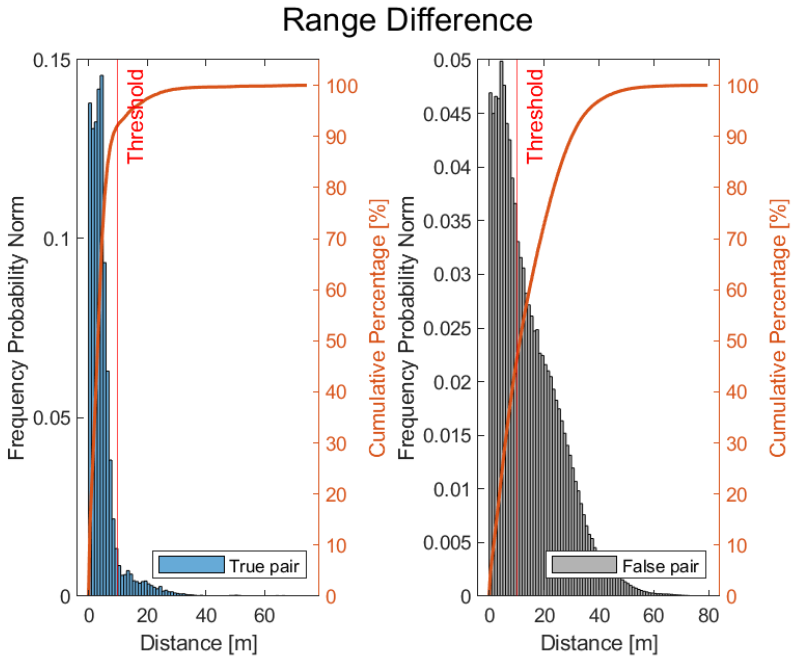


Figure A.30: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Pair range difference.

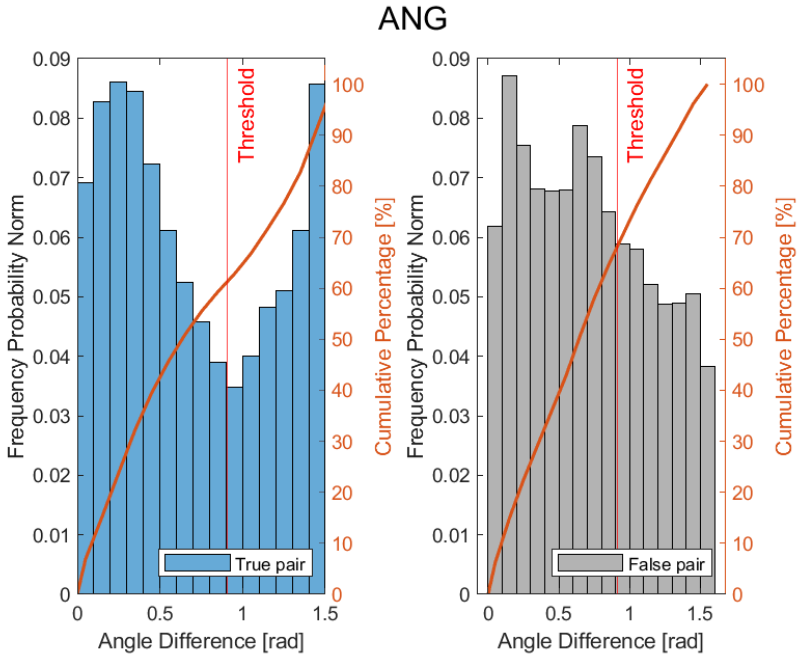


Figure A.31: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria ANG.

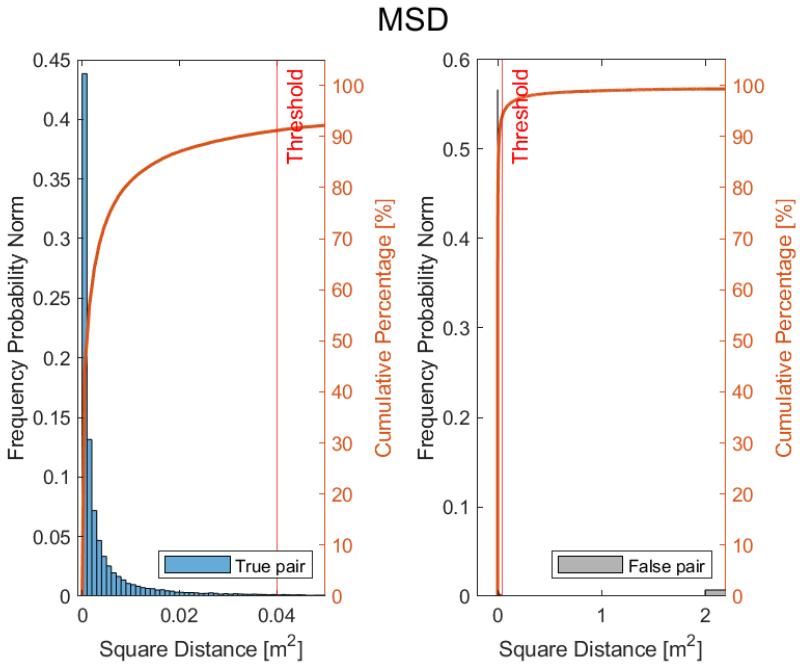


Figure A.32: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria MSD.

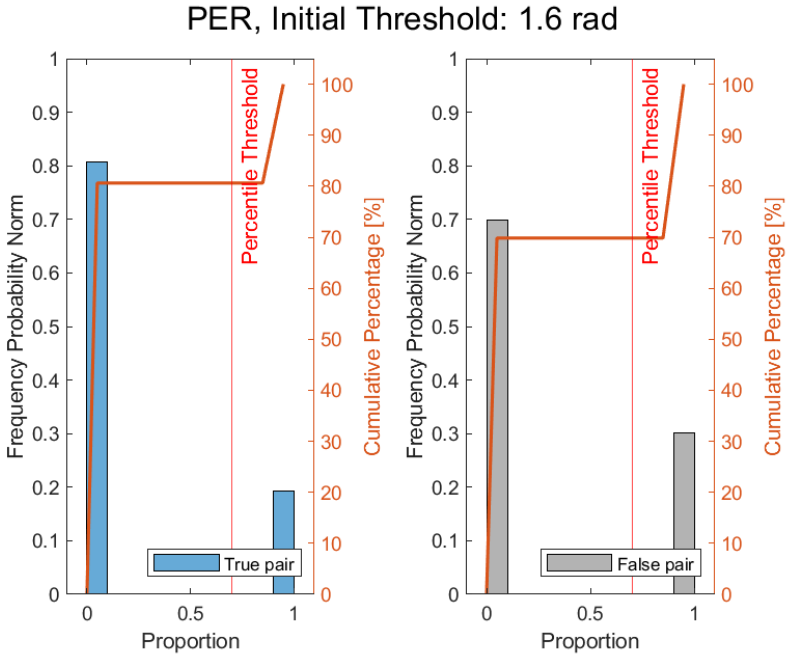


Figure A.33: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria PER.

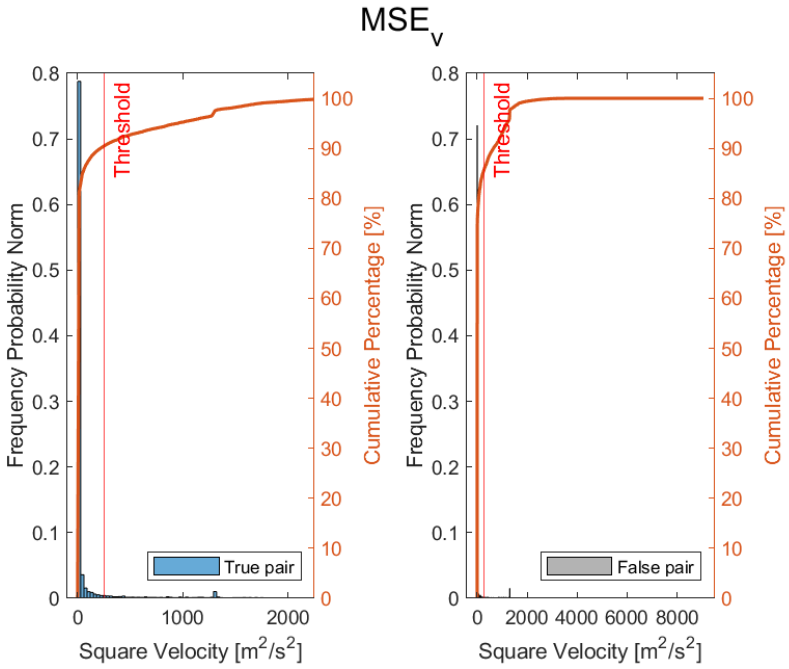


Figure A.34: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria MSE_v.

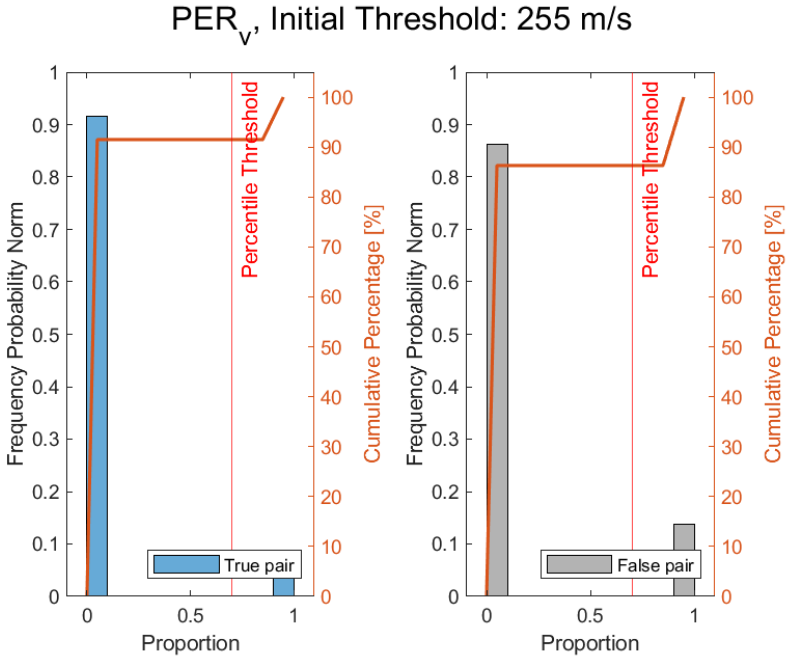


Figure A.35: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria PER_v.

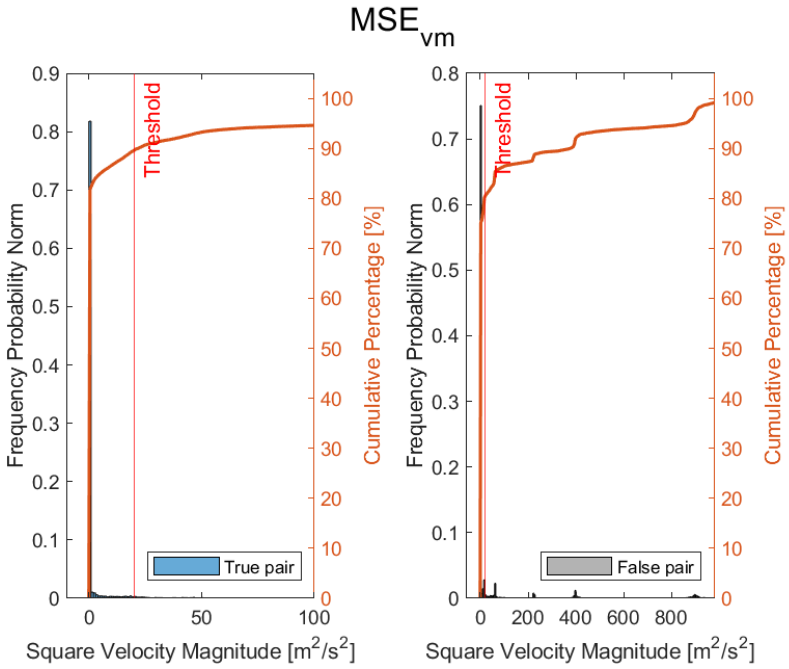


Figure A.36: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria MSE_{vm}.

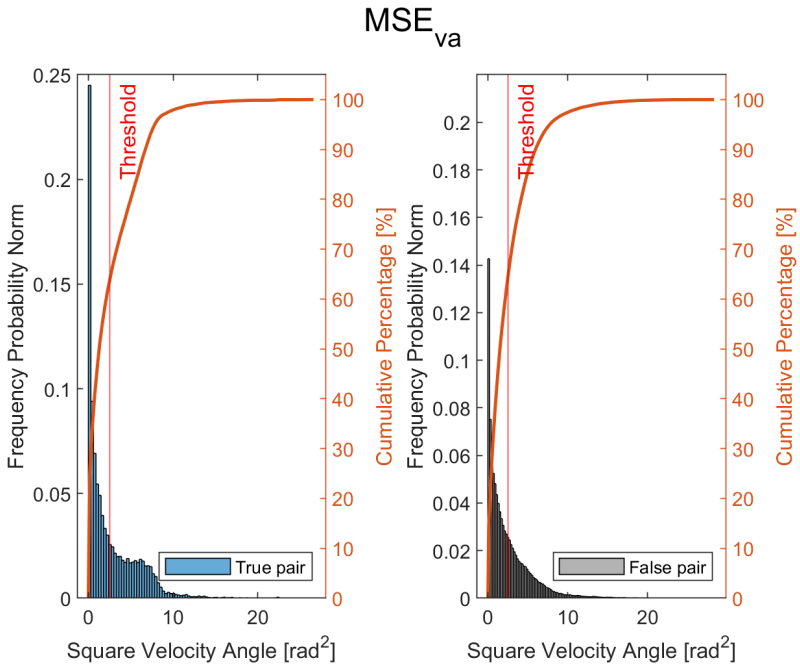


Figure A.37: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria MSE_{va}.

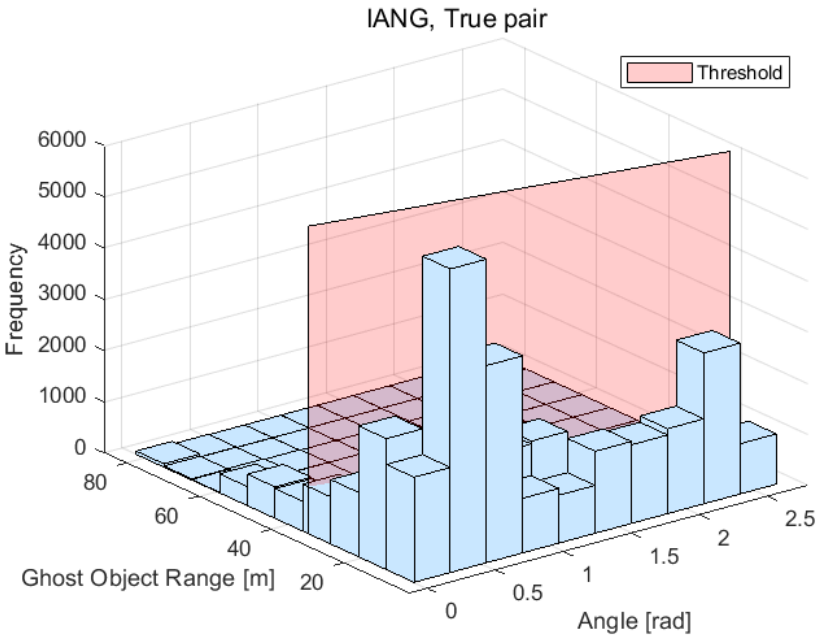


Figure A.38: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria IANG, true pairs.

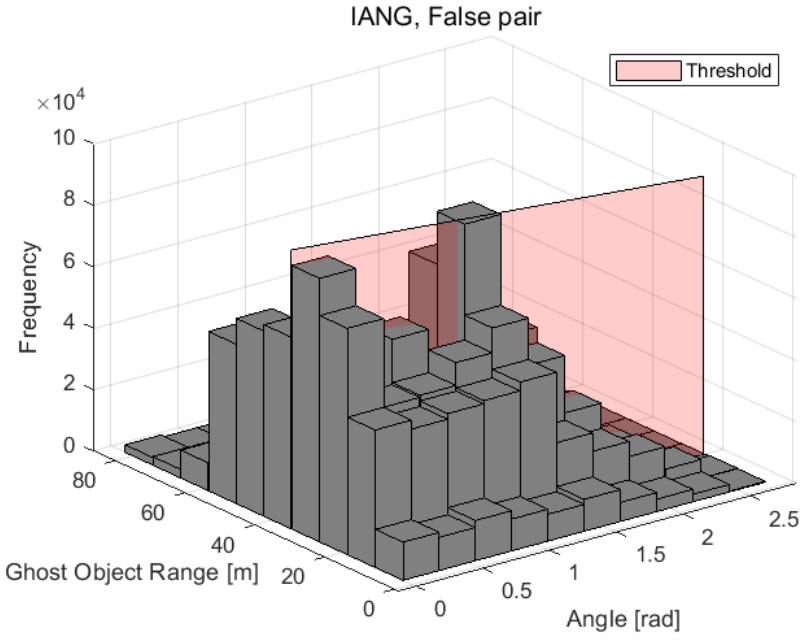


Figure A.39: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria IANG, false pairs.

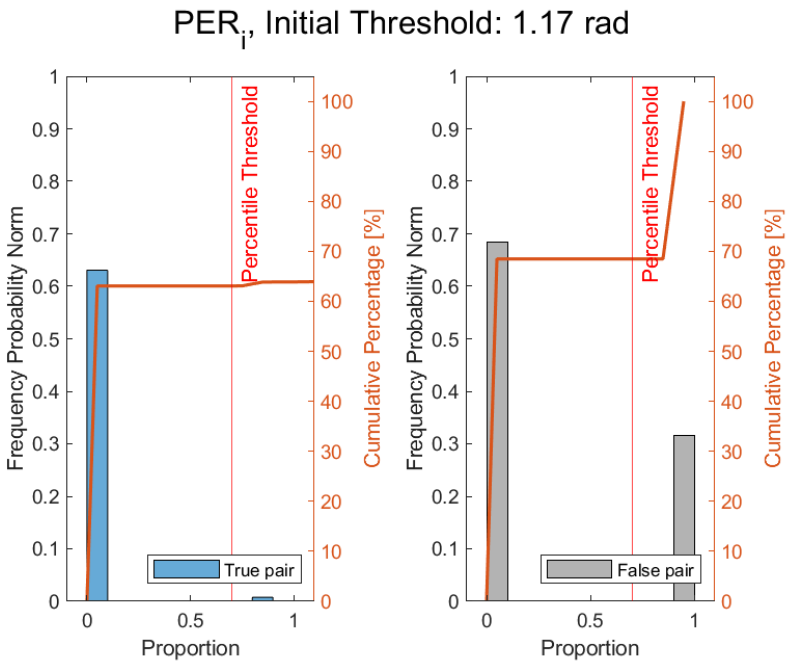
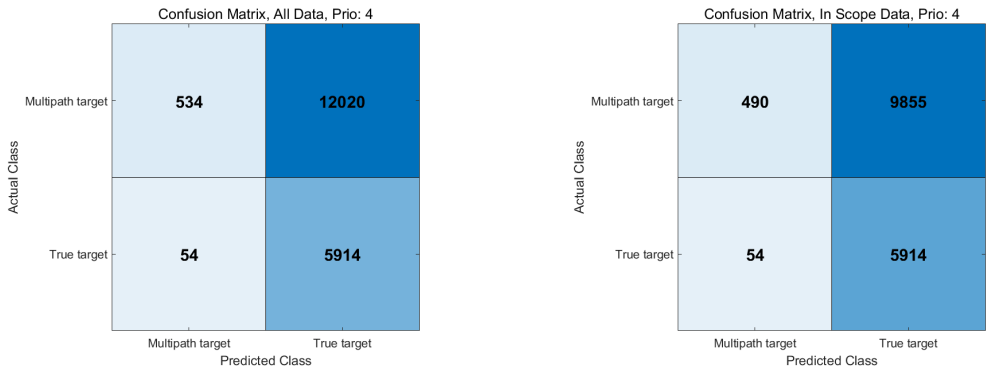


Figure A.40: Statistics and tuned threshold for in scope data from scenario set 2, all priority levels. Criteria PER_i.

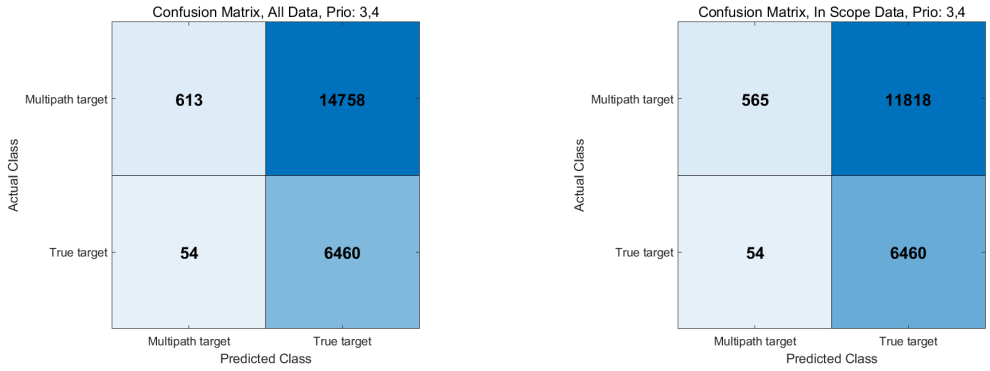
A.3.4 Baseline Scenario Set 2 Results



(a) Performance over all data

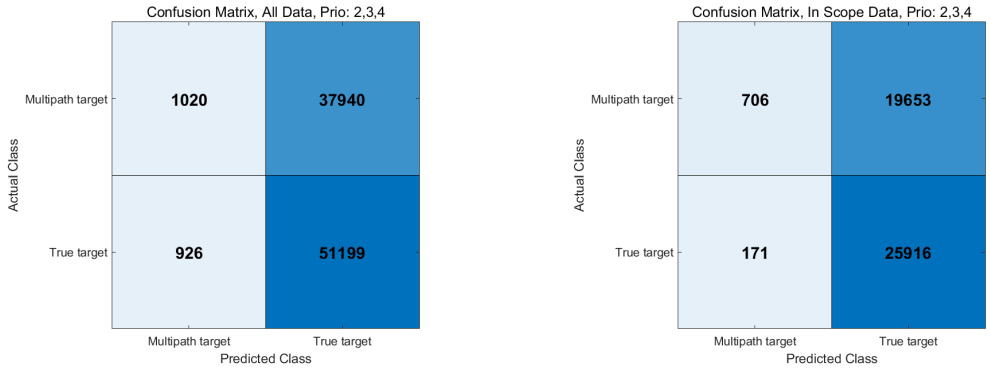
(b) Performance over in scope data

Figure A.41: Performance of state of the art algorithm on scenario set 2, priority level 4.



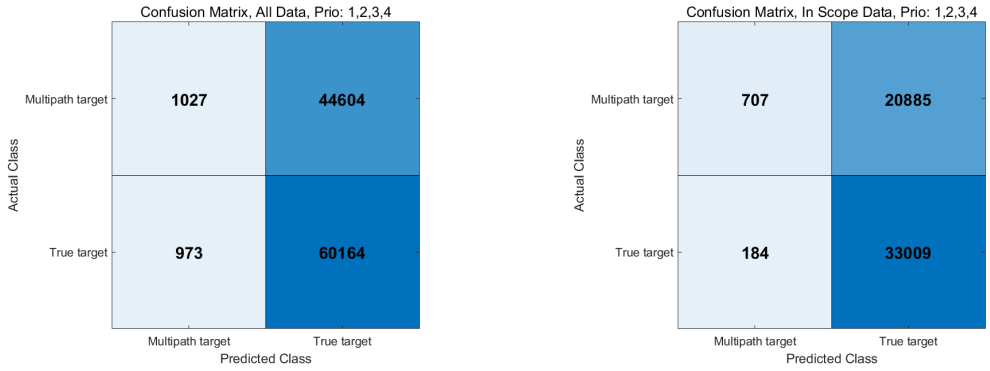
(a) Performance over all data (b) Performance over in scope data

Figure A.42: Performance of state of the art algorithm on scenario set 2, priority levels 3 through 4.



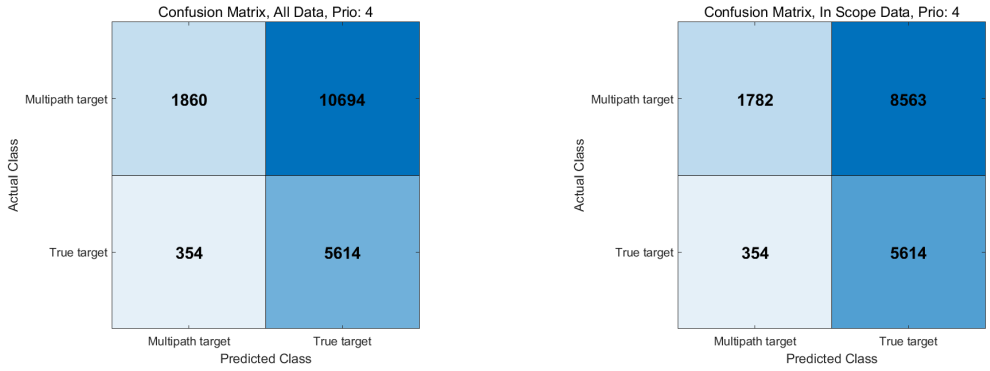
(a) Performance over all data (b) Performance over in scope data

Figure A.43: Performance of state of the art algorithm on scenario set 2, priority levels 2 through 4.



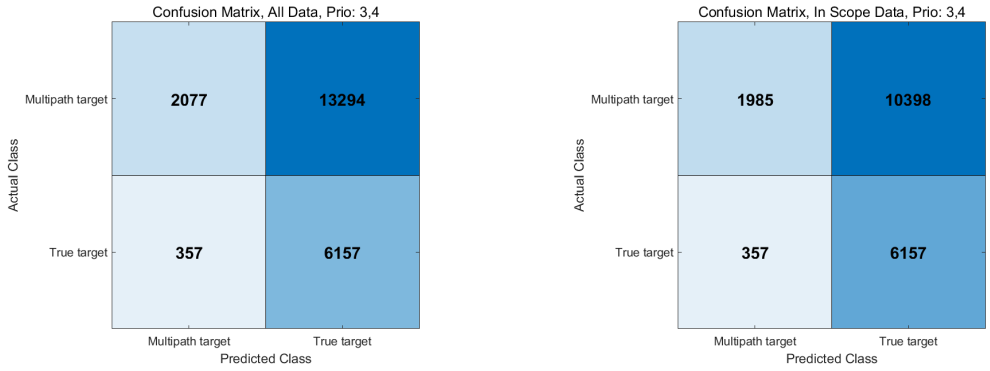
(a) Performance over all data (b) Performance over in scope data

Figure A.44: Performance of state of the art algorithm on scenario set 2, priority levels 1 through 4.



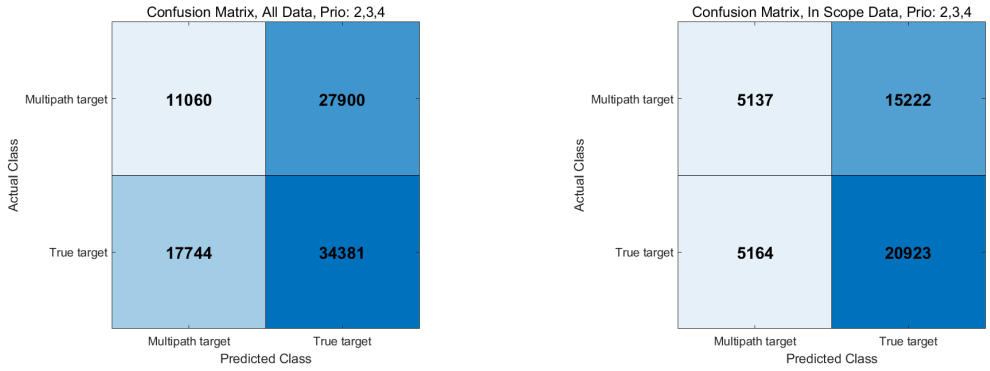
(a) Performance over all data (b) Performance over in scope data

Figure A.45: Performance of state of the art algorithm on scenario set 2 using tuned parameters, priority level 4.



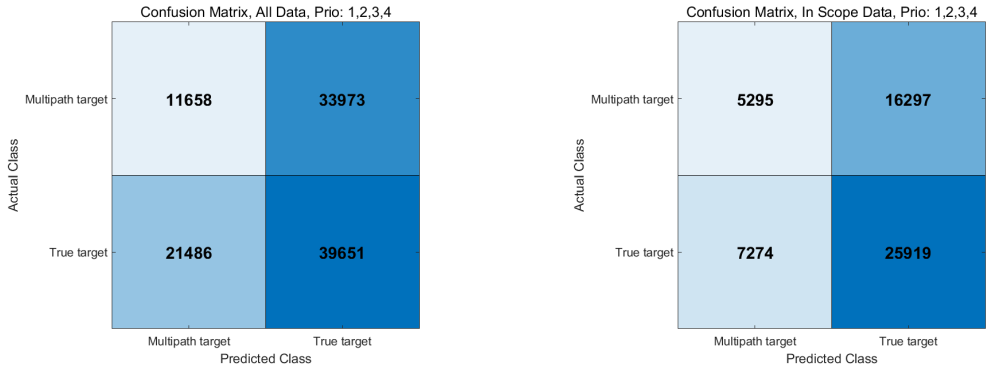
(a) Performance over all data (b) Performance over in scope data

Figure A.46: Performance of state of the art algorithm on scenario set 2 using tuned parameters, priority levels 3 through 4.



(a) Performance over all data (b) Performance over in scope data

Figure A.47: Performance of state of the art algorithm on scenario set 2 using tuned parameters, priority levels 2 through 4.



(a) Performance over all data

(b) Performance over in scope data

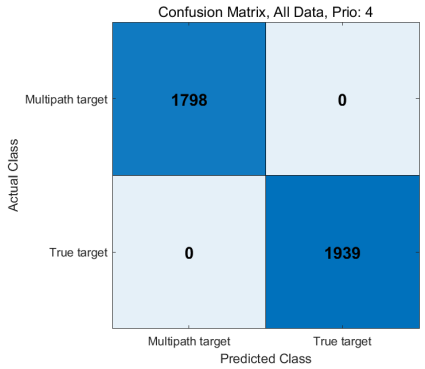
Figure A.48: Performance of state of the art algorithm on scenario set 2 using tuned parameters, priority levels 1 through 4.

A.4 PRED-RAG Algorithm

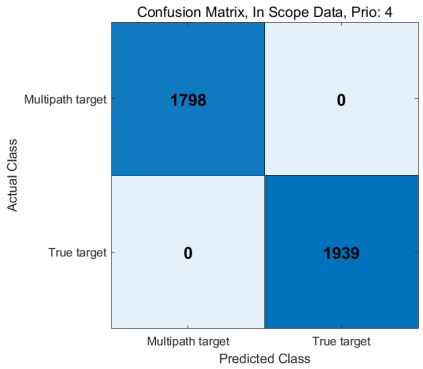
Table A.5: Range-rate distribution lambda values and probability thresholds.

Category	λ^t	λ^f	**Threshold**
Type 1 SSS	3.358	1.238	0.731
Type 1 SSM	0.578	0.096	0.858
Type 1 SMS	4.134	0.189	0.953
Type 1 SMM	4.534	0.077	0.863
Type 1 MSS	0.441	0.068	0.265
Type 1 MSM	0.566	0.406	0.012
Type 1 MMS	1.225	0.108	0.826
Type 1 MMM	1.395	0.720	0.498
Type 2 SSS	1.158	7.171	0.778
Type 2 SSM	0.202	0.074	0.732
Type 2 SMS	0.561	0.057	0.855
Type 2 SMM	0.333	0.062	0.785
Type 2 MSS	0.162	0.063	0.273
Type 2 MSM	0.412	0.342	0.035
Type 2 MMS	0.181	0.049	0.556
Type 2 MMM	1.477	0.654	0.694

A.4.1 PRED-RAG algorithm Set 1 Results

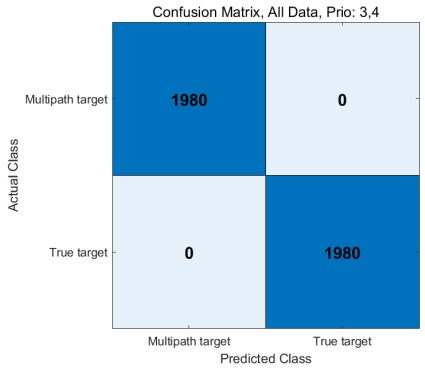


(a) Performance over all data

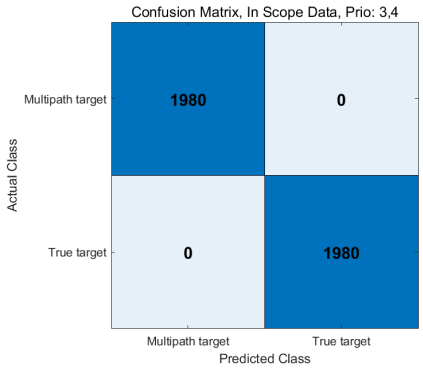


(b) Performance over in scope data

Figure A.49: Performance of PRED-RAG algorithm on scenario set 1 priority level 4.

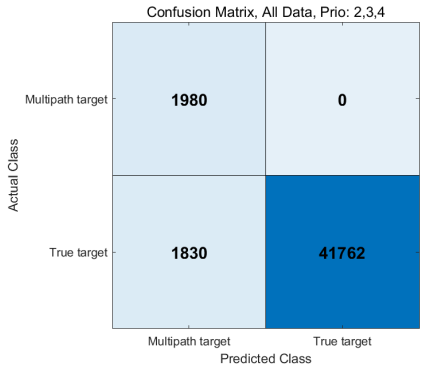


(a) Performance over all data

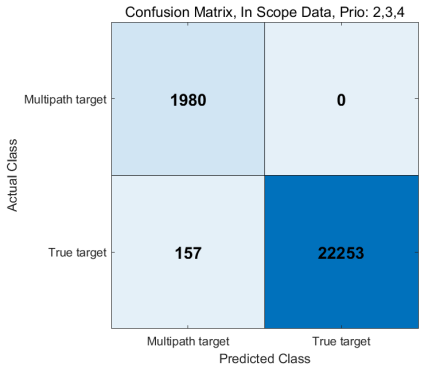


(b) Performance over in scope data

Figure A.50: Performance of PRED-RAG algorithm on scenario set 1 priority level 3 and 4.

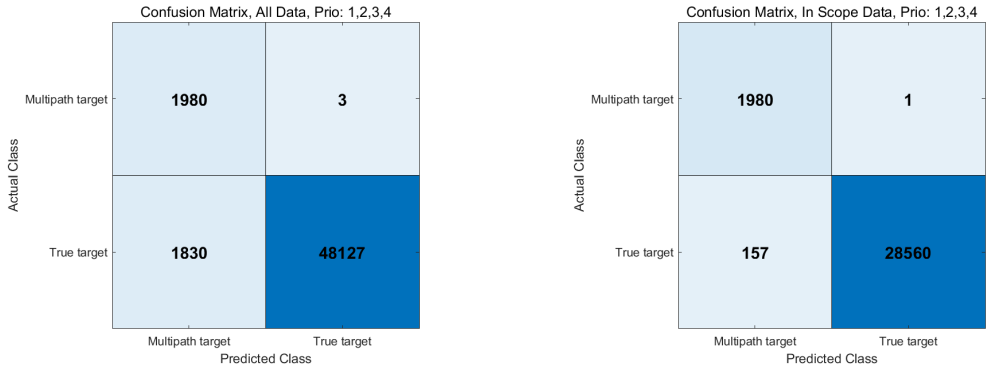


(a) Performance over all data



(b) Performance over in scope data

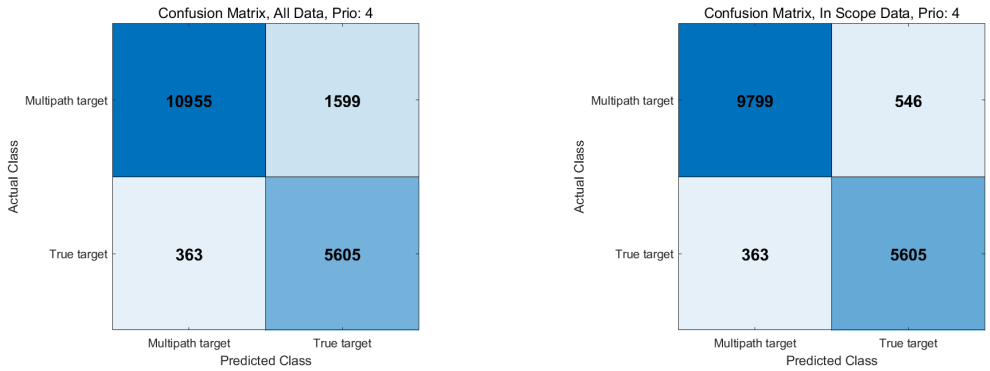
Figure A.51: Performance of PRED-RAG algorithm on scenario set 1 priority level 2, 3 and 4.



(a) Performance over all data (b) Performance over in scope data

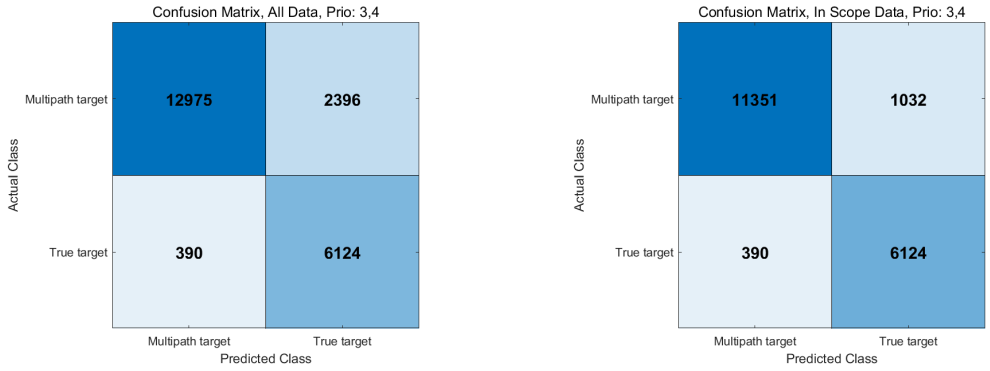
Figure A.52: Performance of PRED-RAG algorithm on scenario set 1 for all priority levels.

A.4.2 PRED-RAG algorithm Set 2 Results



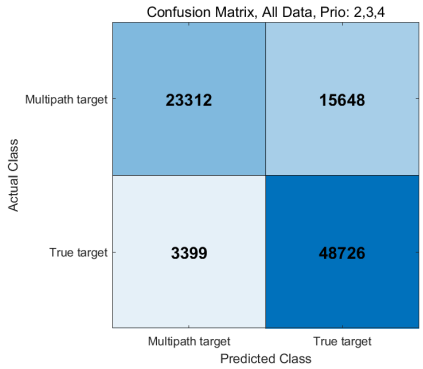
(a) Performance over all data (b) Performance over in scope data

Figure A.53: Performance of PRED-RAG algorithm on scenario set 2 priority level 4.

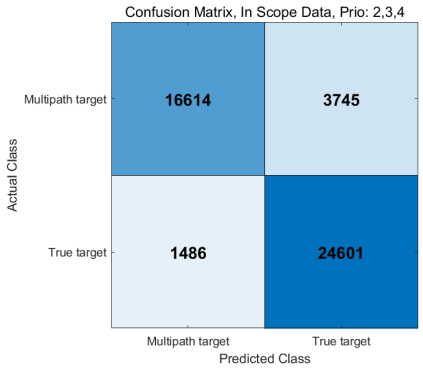


(a) Performance over all data (b) Performance over in scope data

Figure A.54: Performance of PRED-RAG algorithm on scenario set 2 priority level 3 and 4.

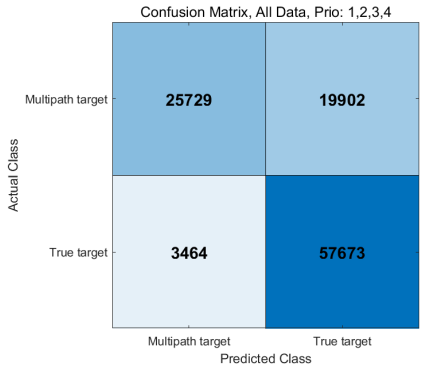


(a) Performance over all data

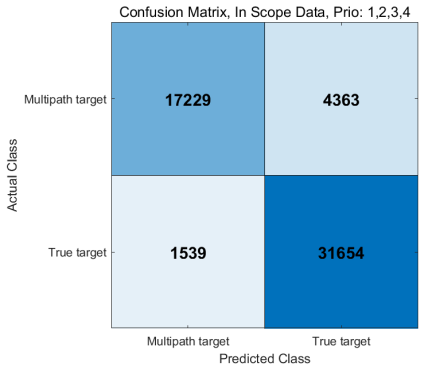


(b) Performance over in scope data

Figure A.55: Performance of PRED-RAG algorithm on scenario set 2 priority level 2, 3 and 4.



(a) Performance over all data



(b) Performance over in scope data

Figure A.56: Performance of PRED-RAG algorithm on scenario set 2 for all priority levels.