Fabian Hirmann, BSc

Data Association of Real-Time Multi-Sensor Fusion for Automated Vehicles

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl-Ing. Dr.techn. Daniel Watzenig

Institute of Automation and Control
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Martin Horn

Graz, November 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Acknowledgements

First, I would like to thank my supervisor Dr.techn. Daniel Watzenig and my co-supervisor Dr.techn. Michael Stolz for their great guidance and support throughout this thesis. You introduced me to the fascinating research on automated vehicles and by this inspired me to do my own work in this field.

I would like to thank the whole group of ADAS/AD at AVL List GmbH for allowing this great collaboration. Special thanks go to my advisor Dr.techn. Thomas Mauthner for always having an open ear, even when he was short on time. All my colleagues make it a great place to work, however, I would particularly like to single out Daniel Hammer. Thank you Daniel for your patient support and for always giving me hope when I was struggling.

I would also like to thank my family for supporting me in everything I do. Special thanks go to my sister who helped me write this thesis by reviewing the entire work and giving me numerous linguistic suggestions. Finally, I would like to thank my friends for providing happy distractions to clear my mind from my studies.

Abstract

Automated vehicles need an accurate and stable perception of the environment. Sensor fusion can provide that by combining measurements from multiple types of sensors and positions. In addition to the high quality of the fusion, real-time performance is needed for automated vehicles.

Before fusing new measurements to the existing tracks, the correct measurement-to-track association must be found in a so-called data association step. There are various methods for the data association of general sensor fusion but there is a lack of research on their feasibility at the application of automated vehicles.

This work analyzes two different methods for the data association, global nearest neighbor (GNN) and joint probabilistic data association (JPDA). GNN is the current state of the art in sensor fusion for automated vehicles. It is a non-Bayesian method whereas JPDA is a Bayesian method with the lowest expected computational effort. JPDA has been tested in its original variant and with extensions for clustering and deduplication. To conduct testing in a structured way, a general framework for the evaluation of sensor fusion variants, and in particular the data association, has been developed. Five relevant scenarios have been created and executed in the simulation. The quality of the fusion has been determined using eleven metrics. Examples are the deviation from the real position, the number of additionally created objects, and the continuity of the assignments to the real objects.

The results show that JPDA in its original variant without extensions does not achieve real-time performance. In the performed tests, a single fusion of measurements has taken up to 5 s. However, JPDA with clustering and deduplication is capable of real time. Its maximum fusion time has been reported to be less than 0.1 s. This is similar to GNN which acts as a reference. Additionally, the fusion quality is better than GNN. For example, the value of a metric combining multiple types of errors has been on average 12 % better than that of GNN. Especially in high-cluttered environments the data association of a Bayesian method is superior to a non-Bayesian. By using the developed general framework for the evaluation of sensor fusion variants, further algorithms can be analyzed in the future as well.

Kurzfassung

Automatisierte Fahrzeuge benötigen eine genaue und stabile Erkennung der Umgebung. Dies kann mittels Fusion der Messungen verschiedenster Sensoren sichergestellt werden. Für automatisierte Fahrzeuge ist jedoch nicht nur die Qualität der Fusion wichtig, sondern auch die Echtzeitfähigkeit.

Vor der Fusion muss jedoch die korrekte Zuordnung der Messungen zu den bereits existierenden Schätzungen der Positionen gefunden werden. Es existieren bereits verschiedenste Lösungsvorschläge für das Zuordnungsproblem für allgemeine Problemstellungen, allerdings wird die spezifische Anwendung des automatisierten Fahrens in der Literatur wenig adressiert.

Diese Arbeit analysiert zwei verschiedene Methoden zur Lösung des Zuordnungsproblems, global nearest neighbor (GNN) und joint probabilistic data association (JPDA). GNN gehört zur Klasse der deterministischen Verfahren und spiegelt den Stand der Technik wider. JPDA zählt hingegen zu den probabilistischen Verfahren, welches im Vergleich zu anderen probabilistischen Verfahren den geringsten Rechenaufwand aufweist. JPDA wurde in der ursprünglichen Variante und mit Erweiterungen für Clusterbildung und Deduplizierung getestet. Für die Analyse wurde ein Framework zum Testen von Sensorfusionsvarianten entwickelt. Dafür wurden fünf relevante Szenarien erstellt und in Simulationsstudien evaluiert. Die Qualität der Fusion wurde mittels elf Metriken festgestellt. Beispiele dafür sind die Abweichung zur wahren Position, die Anzahl der zusätzlich erstellten Objekte und die Kontinuität der Zuweisungen zu den wahren Objekten.

Die Endergebnisse zeigen, dass JPDA in der originalen Variante nicht echtzeitfähig ist. Bei den durchgeführten Tests dauerte eine einzelne Fusion von Messungen bis zu 5 s. Bei JPDA mit Clusterbildung und Deduplizierung war die Fusionszeit stets unter 0,1 s. Diese Variante ist somit echtzeitfähig und auch ähnlich rechenintensiv wie GNN. Zusätzlich ist die Qualität der Fusion besser als jene von GNN. Beispielsweise war der Wert einer Metrik, die mehrere Fehlerarten kombiniert, im Durchschnitt um 12 % besser als jener von GNN. Besonders in Situationen mit vielen falschen Messungen übertrifft die Zuordnung mit probabilistischen Verfahren jene mit deterministischen. Das im Rahmen der Masterarbeit entwickelte modulare Framework ermöglicht die Evaluierung von unterschiedlichsten Sensorfusionsvarianten und -algorithmen.

Contents

List of Figures

1 Introduction

1.1 Motivation and Goals

Automated vehicles and self-driving cars will take safe transportation to the next level. Unlike human drivers, they do not get distracted, tired, or underestimate the criticality of a maneuver. They stay concentrated in every situation and only perform safe decisions. Additionally, automated vehicles have the potential to change the way of transportation. At present, the elderly, disabled individuals, or any other person without a driving license depend on another human driving for them. Automated vehicles can give them back their freedom. [1]–[4]

However, automated vehicles have still a long way to go. In general, the control of automated vehicles is divided into three stages, sense, plan, and act [5], [6]. First, at sense, the vehicle perceives the environment with various sensors. Next, at plan, the vehicle analyzes the situation based on the perceived environment and decides on a plan for the next step. Finally, at act, the vehicle executes the plan with basic actions like steering or accelerating.

This shows that the first step, sense, is crucial as it will influence all later decisions and actions. Automated vehicles perceive their environment with not only one but multiple sensors. There are also different types of sensors like lidar, radar, and camera. Each of them has their unique strengths and weaknesses. Using multiple sensors inherently needs a sensor fusion to combine their measurements. Additionally, sensor fusion creates tracks of the real objects to analyze not only the current position but also their movement in time. Before fusing a new measurement to a currently existing track, the correct association of the measurement to the track must be found. This is called data association. For automated vehicles, in addition to correct associations, real-time performance is also important.

This thesis aims to summarize methods for sensor fusion with a particular attention on the data association, present a structured way of comparing sen-

sor fusion variants in terms of real-time capability and fusion performance, and compare a chosen set of data association methods.

1.2 State of the Art and Related Work

There are various algorithms to perform data association at sensor fusion. Figure 1.1 gives an overview.

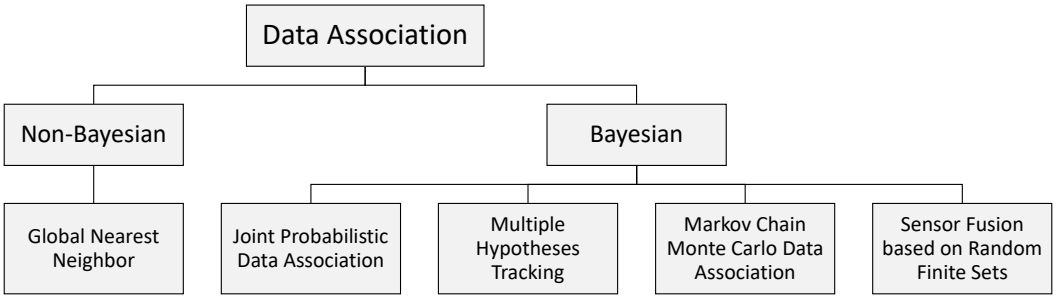

Figure 1.1: Overview of the algorithms for data association

The current state of the art at multi-sensor multi-track fusion for automated vehicles is global nearest neighbor (GNN) in combination with multiple Kalman filters. Examples from open source are [7]–[9]. Global nearest neighbor solves the general assignment problem [10]–[12] by minimizing the overall costs assigned to each possible measurement-to-track association. The Kalman filter [13]–[17] is then used for the actual fusion of the measurement to the track.

Other, widely used methods for general multi-sensor multi-track fusion [18] are joint probabilistic data association fusion (JPDAF) [19], multiple hypotheses tracking (MHT) [20], Markov chain Monte Carlo data association (MCMCDA) [21], and sensor fusion based on random finite sets (RFS) [22], [23]. Unlike sensor fusion with GNN, they solve the data association problem in a Bayesian way by evaluating the probabilities of the possible measurement-to-track associations.

JPDAF evaluates the association probabilities for the current measurements, while MHT evaluates the current measurements including all past measurements. Additionally, MHT works with a dynamic number of tracks instead of a static number as at JPDAF. This makes MHT generally accepted as the preferred data association method for general multi-track sensor fusion [20]. However, enumerating all possible associations can be vast and therefore real-time performance is unclear.
Unlike JPDAF and MHT, MCMCDA is a true approximation scheme for the optimal Bayesian filter. To still enable real-time performance, MCMDA uses Markov chain Monte Carlo sampling instead of enumerating all possible associations.
Sensor fusion based on RFS is a different approach. Instead of a separate step for data association, these sensor fusion algorithms estimate directly the multi-target state in their mathematical description. However, sensor fusion based on RFS is still an emerging field and some of its problems, like track continuity, are not fully solved yet.

There has not been done much research on using Bayesian data association methods at sensor fusion of automated vehicles. Sualeh and Kim [24] have used JPDAF for lidar-based multiple object detection and tracking at automated vehicles. Thomaidis, Spinoulas, Lytrivis, *et al.* [25] have evaluated MHT for automated vehicles. No known research projects have used sensor fusion based on RFS for automated vehicles. Additionally, none of the existing works have evaluated the real-time capability and compared their solution with the current state of the art, GNN.

There are many metrics usable for evaluating the fusion performance of multi-sensor multi-track fusion. The well-known OSPA metric was extended by Ristic, Vo, and Clark [26] to be able to evaluate multi-track fusion as well. Smith, Gatica-Perez, Odobez, *et al.* [27] have presented another set of metrics that uses traditional performance indicators like false positives or false negatives. The multiple object tracking accuracy (MOTA) score [28] is used at a benchmark for general multi-object tracking. It combines false positives, false negatives, and identification switches into one single value.

1.3 Contribution

This thesis has two main contributions.
First, a general framework for the testing of sensor fusion variants, and in particular their data association methods, is developed. This also includes that relevant scenarios are designed and executed in the simulation. The measurements for the sensor fusion are created by using a sensor setup consisting of one lidar, one radar, and two camera images. The perfect data from the simulation are propagated through an imperfect sensor model. The evaluation of the sensor fusion variants is split into two parts, the computing performance and the fusion performance. The computing performance evaluates the real-time capability, which is needed for sensor fusion at automated vehicles. The fusion performance evaluates the quality of the fusion itself. A relevant set of metrics are chosen to analyze and compare sensor fusion variants in a structured way.
The second contribution applies the developed general framework to compare a non-Bayesian data association, GNN, against Bayesian data associations, three variants of JPDAF. GNN is the current state of the art and used as a reference. Particular attention is on the real-time capability and the quality of data association in a high-cluttered environment.

By this, the following scientific questions are addressed and answered in this thesis:

1. How can the different data association methods at sensor fusion be tested?
   - Which metrics can quantify the quality of the data association?
   - What are relevant traffic scenarios?
2. How do alternative data association methods perform against the current state of the art, GNN?

1.4 Structural Overview of the Thesis

The next chapter gives a general introduction to sensor fusion. Particular attention is paid to the different methods for data association and their

strengths and weaknesses.
The third chapter describes the evaluation of different methods for data association in a general framework. In this chapter, first, the simulation environment is presented. Second, the sensor setup, which is used later at the fusion, is defined. Next, meaningful scenarios are designed. The scenarios are evaluated with the metrics defined in the next section. After that, the choice of the compared data association variants is explained. Last, the results of the evaluation are shown, followed by a discussion of them.
The last chapter concludes this work.

2 General Introduction to Sensor Fusion

In the following chapter a general introduction to sensor fusion is given. It introduces current methods and important aspects of sensor fusion that are then evaluated in the next chapter.

In general, sensor fusion algorithms contain the following steps which are processed in cyclic manner:

1. New measurements arrive
2. Current estimated tracks are predicted to the time of the new measurements
3. Measurement-to-track associations are found
4. Measurements are fused to the associated track
5. Possible new tracks are created, existing tracks are confirmed and old tracks are destructed

There are various characteristics to classify sensor fusion algorithms. First, there are the number of tracks. Based on the number of tracks, the sensor fusion algorithm can perform single-target tracking or multi-target tracking. Additionally, sensor fusion can be for a static number of tracks or a dynamic number of tracks. For a static number of tracks, track creation, confirmation, and destruction are not needed. Another characteristic of sensor fusion is how they process measurements [21]. Single-scan algorithms estimate the current state of tracks based on the previously estimated states and the current scan of measurements, while multi-scan algorithms estimate the current state of tracks based on the previously estimated states, multiple past scans, and the current scan of measurements. They may revisit past scans when processing a new scan and can even revise previous estimates based on new evidence found by newer measurements.

According to the shown steps of sensor fusion, the remainder of the chapter is structured as follows.

First, there is the data fusion itself. This eventually fuses new measurements to the current estimate. However, before the fusion, data association must be performed to associate the measurement to the correct track. Another key aspect of sensor fusion is the prediction of the tracks to the current time. Finally, track management takes care of creating new tracks for new objects, confirming existing tracks, and deleting tracks for no more present objects.

2.1 Data Fusion

Data fusion is the core building block of sensor fusion. Sensor data always contain noise and are inherently imperfect. Therefore, data fusion combines multiple sensor data to create a better estimate of reality.

For data fusion there are two widely used methods, the Kalman filter [13], [14] and the particle filter [29]. The Kalman filter assumes Gaussian noise and linear models but needs less computational effort, while the particle filter uses samples of the fused data (so-called particles) and calculates the mean out them. Therefore, particle filters can work with non-linear models and arbitrary probability distributions with non-specific shapes. However, particle filters have a higher computational effort. To overcome some limitations of the Kalman filter while keeping the computational effort low, there exist extensions of the Kalman filter to approximate non-linear models as well [14]–[17].

2.1.1 Kalman Filter

The Kalman filter [13], [14] consists of two parts, prediction and correction. At prediction the current estimate is predicted to the expected value at a later time based on the process model. Hence, the variance of the estimate grows. Section 2.3 gives more details of the prediction. Correction fuses the new sensor data to the current estimate according to the measurement model. This reduces the variance again.

The original Kalman filter assumes a linear process and measurement model, and Gaussian noise. Therefore, various extensions were developed to overcome these limitations. The extended Kalman filter (EKF) [14] linearizes the non-linear model by Taylor approximation. Another extension is the unscented Kalman filter (UKF) [15], [16] that uses the unscented transformation. This extension extracts special points, called sigma points, from the current estimate, propagates them directly through the non-linear model and then calculates the new estimate from the propagated sigma points. Similarly to the UKF, the cubature Kalman filter (CKF) [17] also extracts special points. Effectively the points are even the same as at the UKF with a specific parametrization. The CKF should have advantages over the UKF at numerical accuracy and filter stability.

A very detailed explanation of the Kalman filter, some extensions, and practical problems can be found in [14].

2.2 Data Association

Data association is a crucial point in sensor fusion. When fusing the wrong measurement to a track, the tracking suffers in performance. In the worst case a track can even get lost. A good data association is especially important for sensor data with high clutter or random dropouts.

Figure 2.1 shows a situation with two tracks, $\hat{z}^1$ and $\hat{z}^2$, and four measurements, z_1 to z_4. The ellipses around the tracks are the validation regions. The validation region is set up so that a measurement that falls into that region is with a high probability, called gate probability, originated from the track. Measurements outside the validation region are not considered for data association. This avoids searching for the correct measurement in the whole measurement space. The validation region depends on the expected error of the measurement. For low uncertainties of the track, it is a small region, for high uncertainties it is a large region. All the measurements in the validation region may have originated from the track, even though at most one is the true one. The others must be clutter. [19]

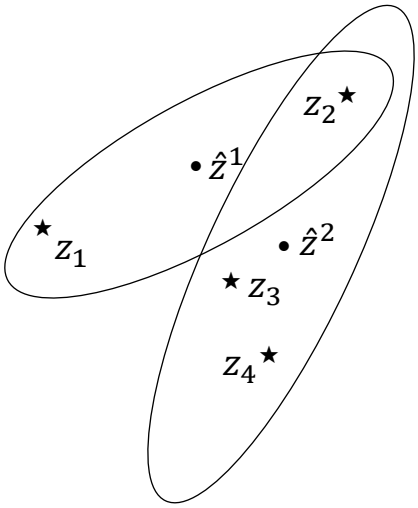

Figure 2.1: Two tracks, $\hat{z}^1$ and $\hat{z}^2$, and four measurements, z_1 to z_4. The ellipsis around the tracks are the validation regions. Source: Redrawn from [19]

At the situation of Figure 2.1, the following measurement origins, or association events, are possible: z_1 from track 1 or it is clutter, z_2 from either track 1 or track 2 or it is clutter, and z_3 and z_4 from track 2 or they are clutter. However, if z_2 originated from track 2, then it is likely that z_1 originated from track 1. This illustrates the interdependence of the data associations when validation regions intersect. In these cases, the joint association events should be considered.

Data association methods can be categorized if they use a Bayesian approach or not. Non-Bayesian methods usually demand less computing performance but can fail at high cluttered sensor data. In contrast, Bayesian methods need higher computing performance but can work well also at high-cluttered sensor data.

2.2.1 Non-Bayesian

Non-Bayesian data association methods use no probability of the associations. Instead, they assign in general arbitrary costs to the possible associations and calculate based on them the association. For automated vehicles,

the costs are, for example, the Euclidean distance between the detected objects from the sensor and the predicted track position from the sensor fusion. Other or additional costs like speed differences are also possible.

Global Nearest Neighbor

Global nearest neighbor (GNN) uses the costs for each data association possibility to calculate the globally lowest cost where each track gets assigned one measurement and vice versa. This is also known as the assignment problem [10].

Assume there are four tracks and four measurements, and they have the following (arbitrary) costs:

		track number			
		1	**2**	**3**	**4**
measurement number	**1**	3	2	8	1
	2	4	5	7	3
	3	2	3	4	5
	4	1	8	7	3

Table 2.1: Cost matrix of the example

This table is also called cost matrix. Based on the costs here the solution with the globally lowest costs is that track 1 gets assigned to measurement 4, track 2 to measurement 1, track 3 to measurement 3, and track 4 to measurement 2. This solution has the globally lowest cost which is summed up 10.

The Hungarian algorithm or also known as the Kuhn-Munkres algorithm is one widely used algorithm to solve the assignment problem [11]. It is strongly polynomial with the time complexity of $\mathcal{O}(n^4)$ [12]. However, this is still much lower than naive calculation which takes $\mathcal{O}(n!)$.

Often some assignments of measurements to tracks can be impossible because, for example, the Euclidean distance between them is far too big. The

threshold for the costs is called validation gate. For impossible assignments the cost gets an infinitely high number assigned to practically exclude it from the global nearest neighbor search.

For a full sensor fusion algorithm, GNN is often combined with Kalman filtering. For multi-target tracking, each track has a separate Kalman filter and works independently. GNN finds the best measurement-to-track association, and then with the Kalman filter the measurement is fused to the track.

High cluttered sensor data can be a problem for sensor fusion with GNN. Because clutter may be assigned the lowest cost, a wrong object might be used at data fusion. This creates higher error or can even lead to track loss.

2.2.2 Bayesian

For Bayesian data association methods, probabilities are taken into account. The methods of probabilistic data association (PDA) and joint probabilistic data association (JPDA) are similar [19]. PDA is the original method but can only work with a single track. JPDA builds up on PDA and adds capabilities for multiple tracks. Both assume a static number of tracks. Multiple hypothesis tracking (MHT) [20] provides a full framework for multi-track sensor fusion including track creation and destruction. Markov chain Monte Carlo data association (MCMCDA) [21] is in general similar to MHT but uses a sampling technique based on Markov chain Monte Carlo to avoid (possibly vast) enumeration of tracks done by MHT. Random finite set (RFS) [22] is an emerging field where sensor fusion can be done without doing an actual step for data association. Instead, it is directly included in the mathematical description of the sensor fusion.

Probabilistic Data Association

Probabilistic data association (PDA) [19] calculates the association probabilities to the track for each validated measurement at the current time. Additionally, the probability that no measurement is originated from the track is also modeled. This Bayesian information is then used in a data

fusion algorithm. Since the process and measurement model are assumed to be linear and the underlying probability density function is Gaussian, the data fusion algorithm is based on the Kalman filter. For non-linear models, the data fusion algorithm is based on the extended, unscented or cubature Kalman filter. The combination of PDA and data fusion is then called PDA fusion (PDAF).

PDAF assumes a single track that has already been initialized and calculates the new estimate from the current estimate and only the current measurement. Therefore, PDAF is a single-target tracking, single-scan sensor fusion algorithm. Compared to data fusion with GNN, there is not only one measurement that gets entirely fused to the track but a combination of them based on the association probabilities. This also includes the probability that no measurement is originated from the track. Additionally, the probability of clutter is specifically modeled.

For modeling the clutter, there are two variants. At the parametric PDA, the clutter is modeled with a Poisson distribution with the expected number of clutter measurements, while at the non-parametric PDA all numbers of clutter measurements are equally likely.

Because PDAF assumes a single track, it is not directly suitable for sensor fusion of automated vehicles. However, extensions like joint probabilistic data association fusion (JPDAF) exist which model multiple tracks.

Joint Probabilistic Data Association

Joint probabilistic data association (JPDA) [19] builds up on the base of PDA and extends it for multiple tracks. The number of tracks is static and they are still assumed to be already initialized. As at PDAF, together with data fusion based on Kalman filter or its extensions, it evolves to JPDA fusion (JPDAF). Each track can have a separate process and measurement model. For JPDAF the data fusion happens in a decoupled manner. For a coupled manner, where the assumption is that the states of the tracks are correlated, there exists the extension called JPDA coupled filter (JPDACF). All other characteristics from PDAF, like being a single-scan algorithm, stay the same.

Unlike PDAF, all joint association events are modeled. For a dense environment with many overlapping validation regions and measurements, the number of joint association events can grow very high. The number of joint association events is similar to the number of permutations which are $n!$. The evaluation of all possible joint association events is therefore the main influence on the computing performance of JPDAF.

Due to the vast enumeration of the joint association events and its impact on the computing performance, clustering is used to overcome this issue. An example that uses clustering is [24, p. 11]. By clustering, all the possible measurement-to-track associations are reduced to separate clusters where each of them is independent of each other. Instead of enumerating all joint association events, only the joint association events of each cluster must be considered.

Figure 2.2 shows a situation where clustering helps to reduce the number of joint association events. There are seven measurements z_0 to z_6 and six tracks $\hat{z}_0$ to $\hat{z}_5$. The ellipses around the tracks are the validation regions. Based on the possible measurement-to-track associations, three independent clusters can be made. This reduces the number of joint association events from 114 without clustering to 19 for cluster 1, 3 for cluster 2, and 2 for cluster 3. In total, there are only 24 joint association events for all clusters compared to 114 joint association events without clustering.

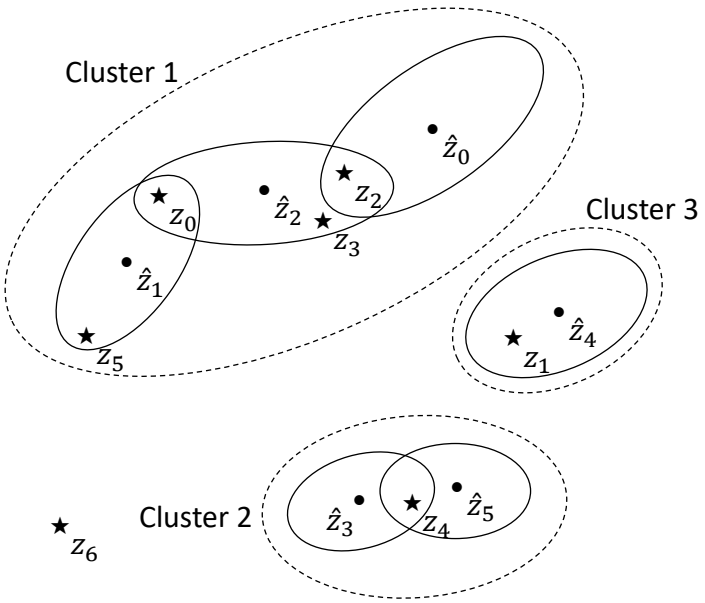

Figure 2.2: Situation with seven measurements z_0 to z_6 and six tracks $\hat{z}_0$ to $\hat{z}_5$

Multiple Hypothesis Tracking

Multiple hypothesis tracking (MHT) [20] provides a full framework for multi-target tracking sensor fusion. Rather than choosing the best data association hypothesis, as with GNN or combining the hypothesis as with JPDA, all hypotheses are propagated into the future in anticipation that subsequent data will resolve the uncertainty. This also means that past estimates can change as the best hypothesis changes. Therefore, it is a multi-scan algorithm. However, this feature also results in an exponentially increasing number of hypotheses which has even higher computational effort than JPDAF. To overcome this issue, pruning is done by discarding hypotheses with low probability.

As a full framework for multi-target sensor fusion, MHT works with a dynamic number of tracks by also modeling the probability of a new track. The tracks also do not need to be initialized.

Markov Chain Monte Carlo Data Association

Markov chain Monte Carlo data association (MCMCDA) [21] is a multi-target tracking method. Unlike MHT and JPDA, MCMCDA is a true approximation scheme for the optimal Bayesian filter. This means it converges to the Bayesian solution when run with unlimited resources. To still enable real-time performance, MCMCDA uses Markov chain Monte Carlo (MCMC) sampling instead of enumerating all possible associations. Similar to MHT, MCMCDA has a multi-scan version and includes detection failure, clutter, and track initialization and termination. Especially under extreme conditions, like a large number of targets in a dense environment, low detection probability, and much clutter, the authors state that MCMCDA shows good computational performance compared to MHT while keeping similar fusion results.

Random Finite Set

Algorithms for sensor fusion based on random finite set (RFS) theory [22] do not require an extra step for data association. In the RFS theory, sets are treated as having random cardinality of finite size, and the values within the sets are also random variables. Unlike all previously presented algorithms, they do not require explicit enumeration of the measurement-to-track association. Instead, they directly estimate the multi-target state. This approach covers more complex multi-target tracking problems under one framework without any ad hoc modifications [18, p. 7].

One computationally feasible algorithm based on RFS theory is the probability hypothesis density (PHD) filter [23]. The PHD filter is a single-scan algorithm. Mahler [23] derived equations for the PHD that account for multiple sensors, non-constant probability of detection, Poisson clutter, and appearance, spawning and disappearance of targets.

The original PHD filter does not work with tracks but just with multiple targets with no specific labeling [22, Sec. 2.1]. PHD filters can be combined with other multi-target trackers like MHT or JPDA to include tracking capability [30]–[32]. Another method to provide tracks with the PHD filter is commonly referred to as track labeling [22, Sec. 2.1]. Examples are [33]–[36].

Dunne and Kirubajaran [37] presented with the weight partitioned PHD filter another variant of the PHD labeling technique.

2.2.3 Overview

Table 2.2 gives an overview of the data association methods for multi-target tracking.

	Category	**Static or dynamic number of tracks**	**Single or multi scan**	**Computational effort**	**Remarks**
GNN	Non-Bayesian	Static	Single scan	Low	State of the art
JPDA	Bayesian	Static	Single scan	Medium	Bayesian method with lowest computational effort
MHT	Bayesian	Dynamic	Multi scan	High	Preferred method for general multi-target tracking but high computational effort
MCMC-DA	Bayesian	Dynamic	Multi scan	Medium to high	True approximation scheme for the optimal Bayesian filter
Sensor fusion based on RFS	Bayesian	Dynamic	Single scan	Medium to high	Novel method with little basic research

Table 2.2: Overview of the data association methods for multi-target tracking

2.3 Prediction

Prediction aims to change the track states in time as they would change in reality. This also helps to perform data association to the measurements since with a perfect prediction, the new measurement would exactly match the predicted one. In practice, there are still differences between the predicted track and the measurement because of measurement and prediction uncertainty, but they are less than without any prediction. However, prediction alone without fusing any new measurements inherently increases the uncertainty of the tracks.

2.3.1 Motion Model

To perform prediction, a motion model is used. The motion model should simulate the actual motion as well as possible while keeping the computational effort low [38, p. 13].

The motion model depends on the use case and properties of the tracks. No motion model is best for every situation. For example, vehicles can only move in certain directions while pedestrians can move in any direction. Therefore, different motion models were developed.

For vehicles, common motion models in increasing complexity are *constant velocity* (CV), *constant acceleration* (CA), *constant turn rate and velocity* (CTRV), and *constant turn rate and acceleration* (CTRA). [39]

There also exist hybrid filters that combine multiple motion models. One practical method for that is the interacting multiple model (IMM) estimator [40]. It automatically switches between multiple models according to a set of transition probabilities. An advantage of IMM is that it can be set up using Kalman filter, or any of its extensions like EKF, UKF or CKF. This IMM-(E/U/C)KF can then be used in any other sensor fusion algorithm like PDAF or JPDAF.

If using IMM, multiple motion models based on the behavior of the car can be used. Jo, Lee, Kim, *et al.* [41] proposed motion models based on the roadway geometry. These are *constant velocity lane keeping* (CVLK), *constant*

acceleration lane keeping (CALK), *constant velocity lane changing* (CVLC), and *constant acceleration lane changing* (CALC). Then, IMM chooses a lane-keeping or lane-changing motion model based on the current behavior of the car. However, this only works for constant roads without intersections, for example highways.

2.3.2 Ego-Motion Compensation

When the coordinate systems of the sensors are not fixed but move relative to the global coordinate system, this motion must also be compensated. Similar to the motion model, there are again different models to compensate this motion. The simplest models only take care of the change of positions by using the traveled distance based on their velocity. More advanced models also include the acceleration and turning.

2.4 Track Management

Track management ensures that new tracks are created, existing tracks are confirmed, and no more existing tracks are destructed. Examples for track management are in [25, p. 4] for MHT or in [24, Sec. 5.2] for JPDAF.

2.4.1 Track Creation and Confirmation

Measurements that cannot be associated with any existing track could be a newly detected object in the sensor view. Another possibility is that those measurements are clutter. Therefore, the combination of first track creation and later confirmation is used to eliminate the wrong tracks caused by clutter. Only confirmed tracks are added to the output of the sensor fusion algorithm.

An ad hoc solution for this is that at track creation any measurement, which is not assigned to an existing track, is created as a new but non-confirmed track. After a non-confirmed track gets associated with enough

measurements, it will get confirmed. This eliminates tracks caused by clutter because it is unlikely that there are many consecutive clutter measurements at the same position.

Another method is to specifically model the probability of existence and above a threshold, the track gets confirmed.

2.4.2 Track Destruction

Track destruction ensures that old tracks, which are no more in the sensor view, get deleted. This also reduces the computational effort for the other parts of sensor fusion. For tracks that must be predicted in the future but no measurement can be associated, the uncertainty grows to the whole measurement space. Then, in practice no statement of the track position can be made. However, tracks should also not get deleted too fast. It could be that the track is only shortly out of the sensor view, currently blocked by another object, or there is a short dropout of the sensor.

A solution for track destruction is to delete a track after there cannot be assigned any measurement for a certain time. Another method is to use again the probability of existence and delete the track if it is below a certain threshold.

Usually confirmed and non-confirmed tracks are also treated differently. Non-confirmed tracks can be deleted faster, as it was likely only clutter, while confirmed tracks should stay longer.

3 Evaluation

This chapter describes the evaluation of different methods for data association of real-time multi-sensor sensor fusion in a general framework.

First, the simulation environment, including the testing workflow, is presented to provide a comparable environment. Second, the sensor setup that is then used for fusion is defined. Next, meaningful scenarios are presented which challenge the different sensor fusion variants. Various metrics compare sensor fusion in a structured way. The next section presents the choice of the compared data association methods, GNN and three variants of JPDA. Last, the results for each data association method are shown, followed by a discussion of the results.

3.1 Simulation Environment and Testing Workflow

To provide a comparable environment with ground truth, simulation is used. The simulation is done by using VIRES Virtual Test Drive [42]. The scenarios are created with the included scenario editor in the OpenDRIVE® standard [43].

The sensor data is stored and processed in the OSI-format. The open simulation interface (OSI) [44] is a generic interface for the environment perception of automated driving functions in virtual scenarios. OSI ensures modularity, integrability, and interchangeability of the individual components from the environment simulation over the sensor model and logical model to the function itself. OSI is widely used and open source. In this framework it provides interchangeability of the parts and a standard format to easily test other sensor fusion algorithms as well.

The full testing workflow is presented in Figure 3.1. Simulation with the defined sensor setup and scenarios is performed to create sensor data and

ground truth data in the OSI-format. The different sensor fusion variants use the sensor data in the OSI-format as input and provide the fused data again as OSI sensor data. The outputs of the sensor fusion variants are then evaluated against the ground truth.

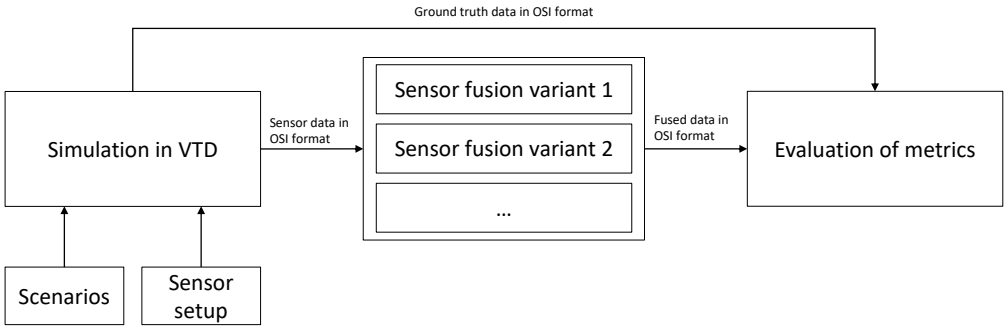

Figure 3.1: Testing workflow

3.2 Sensor Setup

Before evaluation, a sensor setup must be defined. Figure 3.2 shows the sensor setup. This setup mainly focuses on the front view for highway scenarios. It is based on a rooftop box solution from AVL List GmbH called Dynamic Ground Truth (DGT) medium lite. It combines the strengths of lidar, camera and radar.

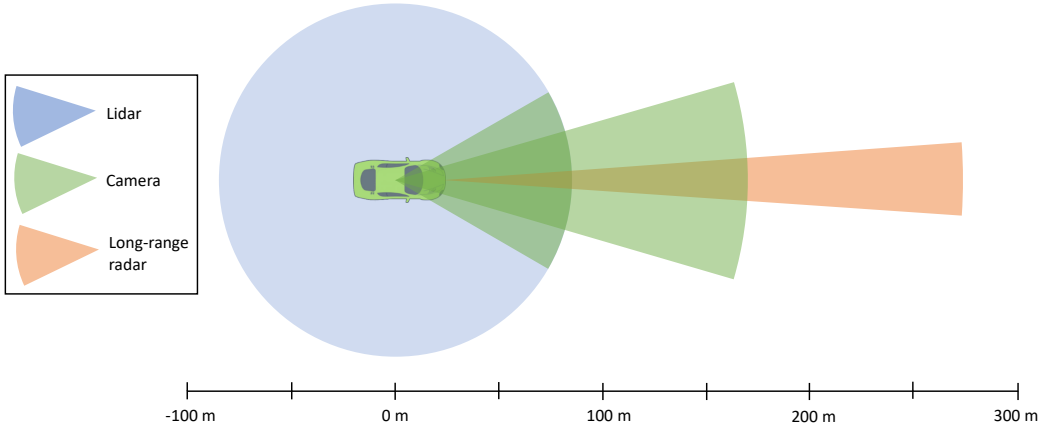

Figure 3.2: Sensor setup. The field-of-views are scaled.

Table 3.1 shows the detailed field of view of each sensor.

	Camera 1	**Camera 2**	**Lidar**	**Long range radar**
Range	85 m	170 m	85 m	250 m
Angle of view	$\pm 30°$	$\pm 16.6°$	360°	$\pm 4°$

Table 3.1: Field of view of the sensor setup

At this simulation, each sensor measures the same values which are the position, speed, acceleration, orientation, and turn rate.

Ground truth data is provided by using a virtual sensor in the origin of the vehicle with a field of view of 360° and a range of 250 m.

3.2.1 Sensor Model

A sensor model is used to create imperfect sensor data from a perfectly simulated environment. The sensor model adds error on the position, speed, acceleration, orientation, and turn rate. Additionally, random clutter is added to the sensor data. The number of clutter objects is distributed according to the Poisson distribution. The positions of the clutter objects are uniformly distributed inside the field of view of the sensor.

For this evaluation the error on the measured values is set low as the data fusion itself is not the main aspect of this work. Instead, more focus is on the clutter because this will challenge the data association part.

3.3 Scenarios

The sensor fusion algorithms are tested in different scenarios.

Ulbrich, Menzel, Reschka, *et al.* define a scenario as follows:

> A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions & events as well as goals & values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time [45, p. 5].

The used term *scene* has the following definition:

> A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities [45, p. 2].

Examples for dynamic elements are the dynamic objects' states and attributes, for the scenery the lane network, stationary objects, or the environment conditions, and for the self-representation the skills and abilities like the field of view or occlusions [45, p. 2].

The scenarios are so chosen that they challenge the data association part. All of them are derived from real world. The urban scenario challenges the computing performance in a dense environment. The other highway scenarios cover situations where an object is occluded and then (re-)appears.

3.3.1 Urban

The urban scenario challenges the computing performance. First, this scenario shows a short following of a two-lane road with oncoming traffic. Additionally, parked vehicles are on the side of the road. Then, the ego

vehicle comes to a congestion caused by a red traffic light and stops. After the traffic light switches to green, the other vehicles move again and the ego vehicle takes a right turn to a highway entrance ramp. Then the simulation stops. Figures 3.3 to 3.8 show snapshots of the simulation of the urban scenario.

This scenario covers oncoming traffic, standing vehicles, and a dense environment with many vehicles at the congestion. This will challenge the computing performance and checks if real-time sensor fusion can also be achieved with many vehicles.

Figure 3.3: Scenario urban: Begin of scenario: The white car is the ego vehicle and drives constantly with around 50 km/h on a one-lane road with oncoming traffic. After a few seconds, the road broadens to two lanes and the ego vehicle drives on the right lane. Additionally, there are stationary vehicles parking on the right side.

Figure 3.4: Scenario urban: Mid of scenario: The ego vehicle drives still with around 50 km/h on the right lane. After around 20 s, the ego vehicle comes to a congestion caused by a red traffic light and has to decelerate.

Figure 3.5: Scenario urban: Begin of congestion: The ego vehicle nearly stopped because of the congestion caused by the previously red traffic light. There are many vehicles close to the ego vehicle.

Figure 3.6: Scenario urban: End of congestion: Before a full stop, the ego vehicle can accelerate again because the traffic light has switched to green and the leading vehicles are moving again. At the intersection, the ego vehicle will turn right to a highway entrance.

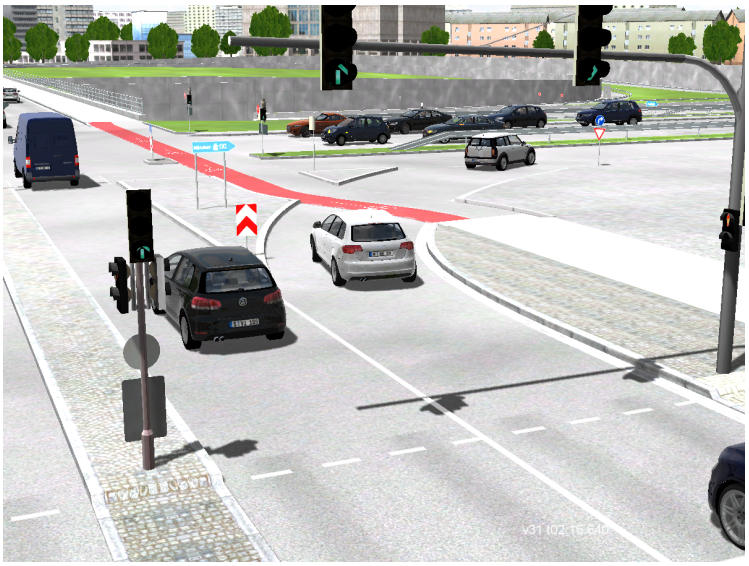

Figure 3.7: Scenario urban: Begin of highway entrance: The ego vehicle turns right to a highway entrance. Many vehicles are waiting on the right side of the intersection coming from the highway. They are visible for the first time and must be detected from the sensor fusion algorithms. Additionally, the ego vehicle has a lot of movement when turning right which must be taken into account by the ego-motion compensation. Since the compensation can never be perfect, this inherent error between the expected and measured position of the other vehicles further challenges the data association.

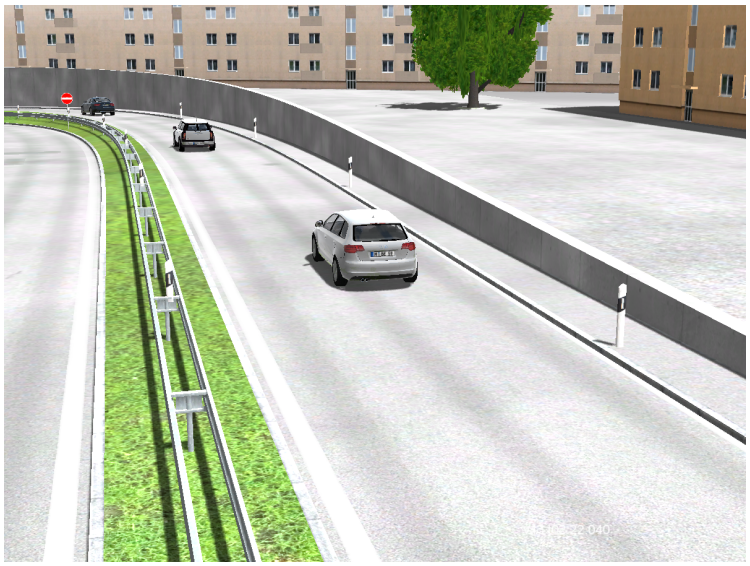

Figure 3.8: Scenario urban: End of scenario: After a few seconds on the highway entrance, the scenario ends. This snapshot is the last included in the urban scenario.

3.3.2 Highway overtaking

This scenario was found as a critical scenario for automated vehicles by Hansen [46, p. 153]. It is further described in Figure 3.9. In this scenario, the sensor fusion algorithm should detect fast the blue car so that the ego vehicle can react properly.

In detail the used scenario is as follows. The blue ego vehicle follows the green car with a constant distance of 30 m and a vehicle speed of 140 km/h. 150 m in front of the green car, the red car drives much slower with 60 km/h. At a distance of 75 m to the red car, the green car changes the lane to the left. The ego vehicle sees the red car only after it. As a result, it switches lanes later with a shorter distance of 65 m to the red car.

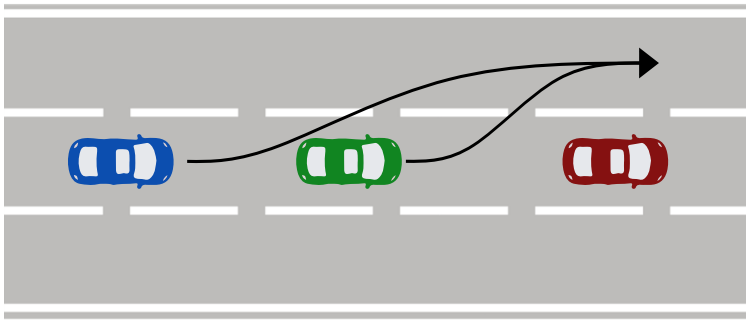

Figure 3.9: Scenario highway overtaking: In front of the ego vehicle (blue) another car (green) is driving with the same speed. That car then overtakes the red car, which is much slower. The ego vehicle reacts fast and also overtakes the red car after the green car. Source: Redrawn from [46]

3.3.3 Highway cut-through between

This scenario covers a situation where one car is shortly occluded by another car. While occluded, the other car drives at a constant speed and a constant distance. Therefore, this scenario should be the easiest for the data association algorithms. In real-world, this scenario happens when a faster car wants to switch lanes so that it can exit the highway.

In detail the used scenario is as follows. The blue ego vehicle follows the red car with a constant distance of 40 m and a vehicle speed of 130 km/h. The orange car drives on the rightmost lane 25 m in front of the red car with a lower speed of 125 km/h. The green car drives faster with 140 km/h and starts 20 m behind the blue car. 15 m behind the red car, the green car changes both lanes to the right and simultaneously lowers its speed to 125 km/h.

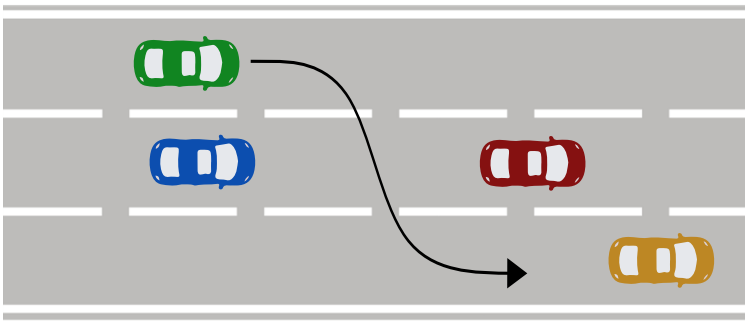

Figure 3.10: Scenario highway cut-through between: The blue vehicle is the ego vehicle. The green car cuts through both lanes between the ego vehicle and the red car in front. This maneuver shortly occludes the red car.

3.3.4 Highway cut-through front 1

Similar to the scenario before, this scenario covers again a situation where one car is shortly occluded. The difference is that while occluded, the car switches the lane and has a different speed. Therefore, after reappearing, the car has a different position and speed. In real-world, this scenario happens in normal situations when one car overtakes other cars.

In detail the used scenario is as follows. The blue ego vehicle follows the red car with a constant distance of 30 m and again a vehicle speed of 130 km/h. The orange car drives on the rightmost lane 20 m in front of the red car with a lower speed of 125 km/h. The green car drives faster with 140 km/h and starts 20 m behind the blue car. When the green car is 20 m in front of the red car, the green car changes both lanes to the right.

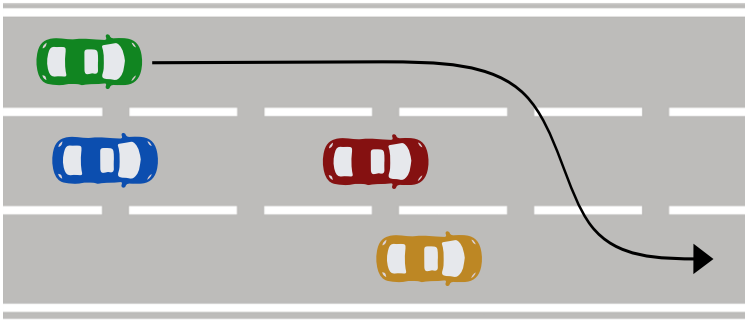

Figure 3.11: Scenario highway cut-through front 1: The blue vehicle is the ego vehicle. The green car is faster and overtakes all cars. After that it cuts through to the rightmost lane. This maneuver occludes then the green car behind the red car. After switching the lanes, the green car is again visible.

3.3.5 Highway cut-through front 2

This scenario is similar to the scenario before. The difference is that this time the overtaking car brakes again. In real-world, this happens when the overtaking car wants to exit the highway after overtaking and has to decelerate for that.

In detail this scenario is the same as the previous scenario with an additional step at the end. After all other maneuvers, the green car decelerates to 100 km/h. Therefore, the orange car must also decelerate to 100 km/h. The red and the blue car then overtake both other cars.

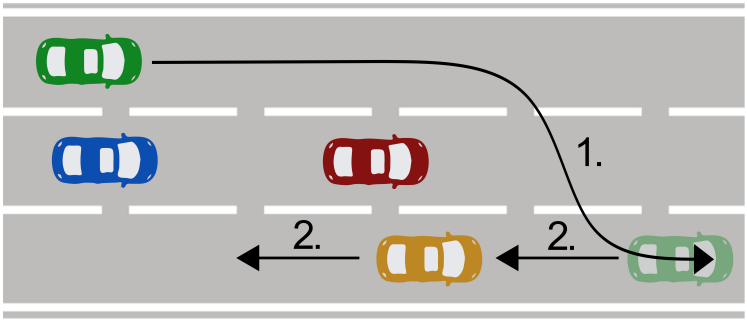

Figure 3.12: Scenario highway cut-through front 2: The blue vehicle is the ego vehicle. This scenario is similar to the scenario before. The difference is that after the green car was cutting through, it decelerates which then also causes that the orange car has to decelerate. At the end, the green and orange car is overtaken again by the other cars.

3.4 Evaluation Criteria

The evaluation of the sensor fusion algorithms is split into two parts. One part evaluates the computing performance to assess the real-time capability. The other part evaluates the performance of the fusion itself.

3.4.1 Computing Performance Metrics

To evaluate the computing performance, various statistics about computing time and its connected statistics are gathered. Particular attention is on the number of input objects to the fusion. For Bayesian data association algorithms (Sec. 2.2.2), the number of hypotheses that must be evaluated is another important metric.

The statistics are illustrated with the fusion time and connected metrics at each time step. A boxplot of the fusion times illustrates the area where most fusion time is.

3.4.2 Fusion Performance Metrics

The fusion performance metrics evaluate the quality of the fused results and collect them into comparable metrics.

One used metric is the OSPA metric for tracks. Ristic, Vo, and Clark [26] extended the well-known OSPA metric to evaluate the performance of multi-target tracking algorithms. In addition to the already covered cardinality and localization error, the track label error is considered to capture the data association performance. The OSPA metric for tracks gives an insight into the fusion performance at each timestamp with an overall error, but also split into the cardinality error, localization error, and label error. In this way, special situations can be analyzed and a more detailed view of the strengths and weaknesses of the algorithms can be given.

Smith, Gatica-Perez, Odobez, *et al.* [27] present another set of metrics. They are split into two parts, configuration error and identification error.
The configuration error solely evaluates the correct number of objects and their positions without taking care of a consistent label. It evaluates the number of false positives (FP), false negatives (FN), multiple trackers per ground truth object (MT), and multiple ground truth objects per tracker (MO). The latter two indicate if there are multiple (overlapping) tracks that follow one ground truth object and if the bounding box from one tracker covers multiple ground truth objects, respectively. The configuration distance (CD) measures the difference between the number of tracks and ground truth objects normalized by $\max(N_{GT}^t, 1)$ where N_{GT}^t is the number of ground truth objects at time t. If there are more tracks than ground truth objects, it is positive. If vice versa, it is negative. All of those metrics are evaluated at each timestamp. For an overall metric of the whole scenario, the false positives, false negatives, multiple trackers per ground truth object, and multiple ground truth objects per tracker are normalized by $\max(N_{GT}^t, 1)$ and averaged over the whole time. The configuration distance is time-averaged with its absolute values.
The identification error measures the consistent tracking of a ground truth object. Two metrics are measured every timestamp, falsely identified tracker (FIT) and falsely identified object (FIO). For a perfect result, exactly one tracker follows one ground truth object, and vice versa exactly one ground

truth object is followed by one tracker. For falsely identified tracker (FIT), the number of wrong trackers following the ground truth object is measured, and vice versa for falsely identified object (FIO). Figure 3.13 illustrates the different identification errors. Additionally, the purity is measured to evaluate the degree of consistency to which a track correctly follows a ground truth object and vice versa. The tracker purity (TP) measures the ratio of time that a track follows its assigned ground truth object. The object purity (OP) measures the ratio of time that a ground truth object follows its assigned track. Both metrics are evaluated for each track and ground truth object, respectively. For an overall metric of the whole scenario, the falsely identified tracker and falsely identified objects are normalized by $\max(N_{GT}^t, 1)$ and averaged over the whole time. The tracker purity and object purity is averaged over all tracks and ground truth objects, respectively.

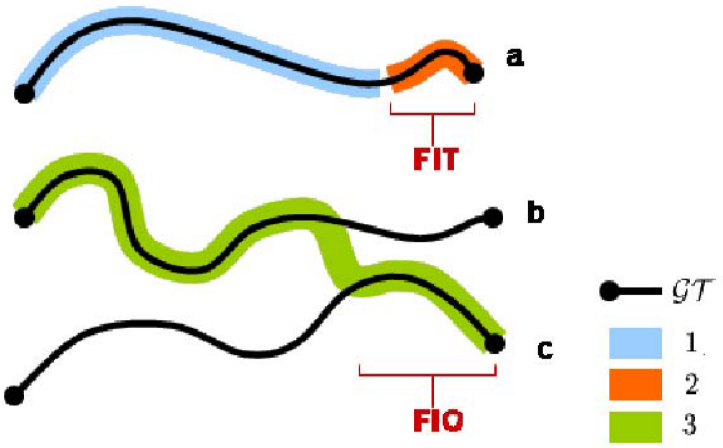

Figure 3.13: Types of identification errors: Three ground truth objects are tracked by three estimates. A falsely identified tracker error (FIT) occurs when object a is tracked by a second estimate. A falsely identified object error (FIO) occurs when an estimate swaps ground truth objects. Source: [27]

The last metric is the multiple object tracking accuracy (MOTA) score [28]. The MOTA is given in Equation 3.1. It combines at each timestamp the number of false negatives (FN_t), false positives (FP_t), and identification

switches (IDSW_t) normalized by the number of ground truth objects at each timestamp (GT_t).

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDSW}_t)}{\sum_t \mathrm{GT}_t} \tag{3.1}$$

Ground-Truth-to-Track assignment

Before evaluation of the metrics, the according ground truth track to the fused track, and vice versa, must be found. Ristic, Vo, and Clark [26, p. 4] suggest a rather simple algorithm, which is used in this work. It assigns costs to each possible assignment and solves them by using any assignment algorithms (for example the Hungarian algorithm, which was presented before in Section 2.2.1). Equation 3.2 shows the costs of assigning a ground truth track s to a fused track t. The ground truth track with label s is denoted as the set of positions $(\boldsymbol{x}^s_{k^s_b}, \ldots, \boldsymbol{x}^s_{k^s_e})$ where k^s_b and k^s_e denote the beginning and the end of the track, respectively. Similarly, the fused track with label t is denoted as $(\boldsymbol{y}^t_{k^t_b}, \ldots, \boldsymbol{y}^t_{k^t_e})$. $e^l_k = 1$ if track $l \in \{s, t\}$ exists at time k. Otherwise e^l_k is zero. $||\cdot||$ denotes the euclidean norm.

$$c(s,t) = \begin{cases} \frac{\sum_k e^s_k e^t_k ||\boldsymbol{x}^s_k - \boldsymbol{y}^t_k||}{\exp\left\{\sum_k e^s_k e^t_k\right\}}, & \text{if } \sum_k e^s_k e^t_k > 0 \\ \infty, & \text{otherwise} \end{cases} \tag{3.2}$$

These costs include the euclidean distance but favor longer duration. Figure 3.14 shows an example with two ground truth tracks and three fused tracks.

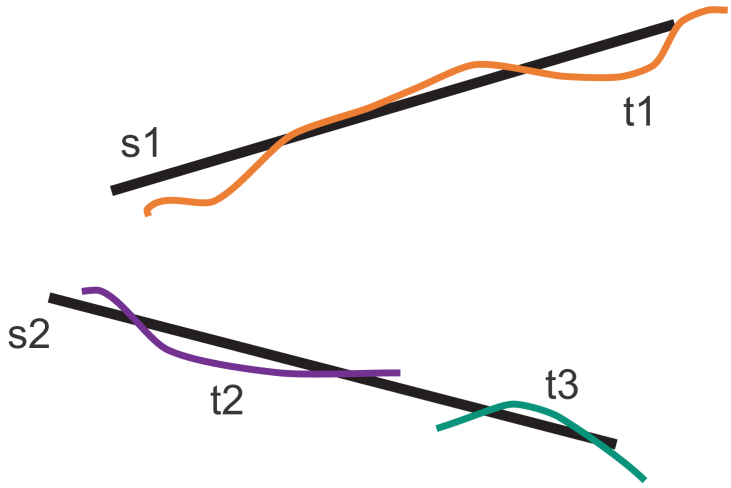

Figure 3.14: Example for ground-truth-to-track assignment with two ground truth tracks, s_1 and s_2 (thick black lines), and three fused tracks, t_1 (orange), t_2 (purple) and t_3 (green). After the global assignment, the ground truth track s_1 is assigned to the fused track t_1 and s_2 is assigned to t_2, respectively. The fused track t_3 has no assigned ground truth track. Source: [26]

3.5 Compared Fusion Methods

Various methods for data association were presented in Section 2.2.

GNN in combination with Kalman filters is a simple, non-Bayesian method that emerged to be the standard for multi-sensor fusion for automated vehicles [7]–[9]. Reasons for that are the low complexity with good computing performance while still having good fusion results. In general, this combination works with a static number of tracks that are already initialized.

The other methods are Bayesian methods that can have advantages in some situations. Especially for a high-cluttered environment, a Bayesian method should have advantages over non-Bayesian methods.

PDAF and its multi-track extension JPDAF are Bayesian, single-scan methods with still low to medium complexity. The number of tracks is in general

static and the tracks are assumed to be initialized. The computing performance strongly depends on the application and specific scenario. PDAF and its extensions are already long-existing sensor fusion methods that are widely used at many different tracking applications, like radar target tracking [19, pp. 15-17]. However, there is also some research in the field of automated vehicles [24].

MHT is a full framework for sensor fusion. It is a Bayesian method that extends to multiple scans and models a dynamic number of tracks including track creation and destruction. Because of this, the complexity is very high. Only approximations of the original MHT are used in practice and can reach real-time performance. MHT is generally accepted to be the preferred data association method for general multi-target tracking applications [20]. An approximation of MHT was already used at automated vehicles [25].

MCMCDA is similar to MHT and shares most properties. However, due to its sampling method, the computing performance should be lower than MHT while keeping similar fusion performance results.

Sensor fusion based on RFS is a new, very different method. Instead of an extra step for data association, they directly estimate the multi-target state. This approach covers more complex multi-target tracking problems under one framework without any ad hoc modifications. However, many aspects of sensor fusion based on RFS, like computational feasibility, practical applications, and consistent track labeling, are still open.

Based on that analysis, GNN and JPDAF are chosen for further evaluation. GNN as the state of the art is used as a reference for comparison. JPDAF is chosen because it is a Bayesian method with still low to medium complexity. Real-time performance could still be possible. Additionally, research on a comparison of GNN and JPDAF, in particular on the real-time capability, is missing. Comparing those two methods also evaluates the possible advantages of a Bayesian method against a non-Bayesian method. JPDAF itself is evaluated in three variants, in the standard variant without clustering and deduplication, with clustering but without deduplication, and with clustering and deduplication. Variants with clustering should provide much better computing performance while keeping the fusion results similarly good. At JPDAF it could happen that two or more tracks will get associated with the same measured object. If this happens multiple times, both tracks effectively

merge into one. At deduplication this effect is prevented by checking if there are overlapping tracks and then removing the younger existing track.

3.5.1 Shared Implementation Details of GNN and JPDAF

The two chosen methods for evaluation, GNN and JPDAF, share some properties. Whenever possible, both use the very same parameters to provide good comparability.

Both use after the data association the Kalman filter for fusing the data. For the motion model, constant turn rate and acceleration (CTRA) (see Section 2.3.1) is chosen. Because CTRA is a non-linear model, the unscented Kalman filter is used for fusion.

The measurement model is a simple, linear model as the measurements are OSI-sensor-data messages which directly contain the measurements of the estimated states. The estimated states of the objects are the position in the longitudinal and lateral direction, the yaw angle, the absolute velocity, the absolute acceleration, and the absolute turn rate. The measured states are the position in the longitudinal and lateral direction, and the yaw angle.

However, for track initialization, all the estimated states are initialized with their measured values to simplify that. As both data association variants do not contain any track initialization by itself, and they use the very same initialization, this additional information for estimation at initialization has still no relevance on the comparison of both variants.

Track confirmation is used to prevent tracks based on clutter measurements. After a track is associated with any measurement for more than five times in its first 0.5 s, it will get confirmed. A confirmed track is deleted again if it is not associated with any measurement for more than 2.0 s. Figure 3.15 illustrates the states of a track.

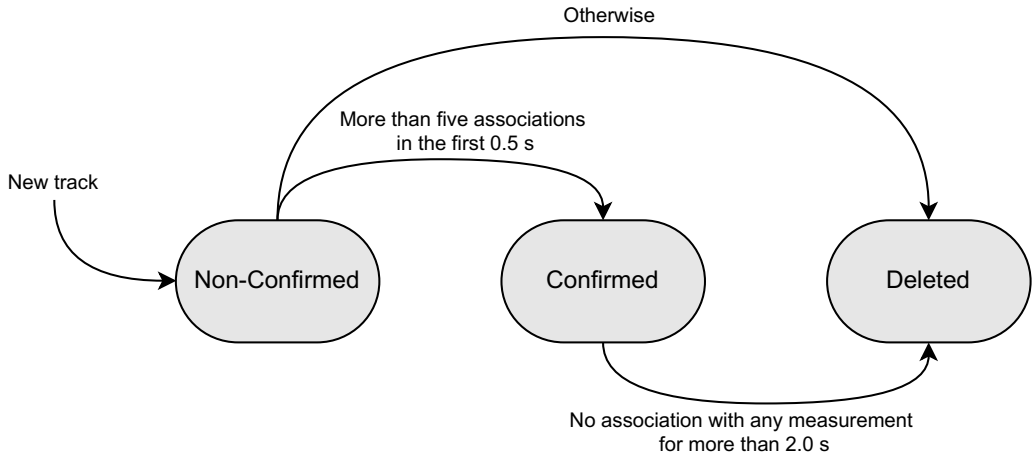

Figure 3.15: Track state diagram: When a new track is created, it is non-confirmed. A track is confirmed if there are more than five associations in the first 0.5 s. If not, it will get directly deleted. A confirmed track is deleted if there is no association with any measurement for more than 2.0 s.

For ego-motion compensation, a more advanced method as presented in Section 2.3.2 is chosen. Not only is the speed compensated, but also the acceleration and turn rate.

Both methods are developed in C++ which allows high computing performance to ensure real-time capability. Additionally, C++ is also the main programming language in the automotive domain.

For testing purposes, both variants create a lot of debug outputs that can cost much performance.

3.5.2 Specific Implementation Details of GNN

GNN uses the Hungarian algorithm to solve the assignment problem. The costs are simply the Euclidean distance between the estimated and measured positions. The validation gate to already exclude unlikely measurements from the assignment problem is 3 m.

3.5.3 Specific Implementation Details of JPDA

Unlike the standard JPDA algorithm, the association matrix here is not built by using the full measured state. Instead, only the position is used. GNN also only uses the position and using the yaw angle can have non-linear effects when the angle changes from $-\pi$ to π.

The non-parametric version of JPDA is used to model the clutter. All numbers of clutter measurements are assumed to be equally likely.

The clustering extension to the standard JPDA algorithm is implemented as described in Section 2.2.2.

Deduplication is implemented by checking if the tracks overlap by using the Euclidean distance of the positions. If they overlap, the older track survives and the younger track dies out.

For better computing performance, all computations, which can run in parallel, are in different threads. Additionally, instead of calculating the same values multiple times, they are precomputed and cached to save computing time.

3.6 Results

The results are split into two parts, the computing performance and the fusion performance.

3.6.1 Computing performance

The computing performance evaluates if the algorithms are suitable for automated vehicles with real-time constraints. For this, solely the urban scenario (Section 3.3.1) is used. This scenario contains a dense environment with many vehicles and will challenge the data association part of the sensor fusion algorithms. The other scenarios only contain a few vehicles that take low computational effort.

The computing performance is measured by a single run on a high-performance laptop (Intel Core i7-8750H, 32 GB RAM).

Figure 3.16 shows a comparison of the fusion time and the number of objects at each timestamp. The fusion time is the time it takes to fuse the measurements of all sensors at that timestamp. The number of input objects is the sum of all the measured objects at that timestamp. If an object is measured from multiple sensors, it is counted multiple times as these are also multiple measurements that will be fused. The number of objects before fusion is the number of validated tracks that are currently estimated before the fusion of new measurements of that timestamp.

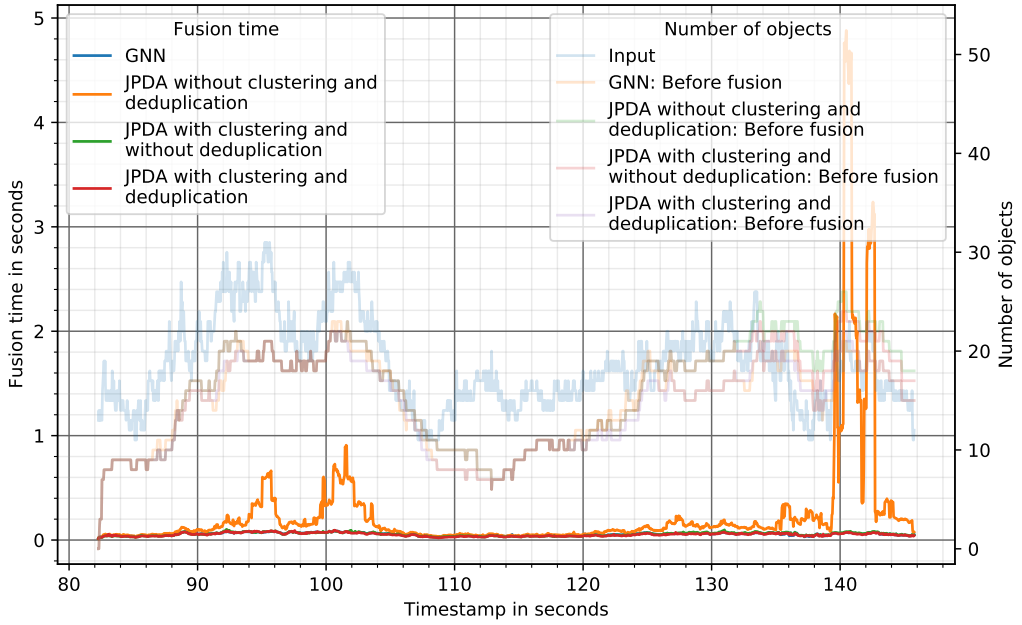

Figure 3.16: Comparison of fusion methods in terms of fusion time at each timestamp. The number of objects is added as an extra information in lighter lines.

The fusion time of JPDA without clustering and deduplication is significantly higher than of the other variants. The other fusion algorithms always stay below a fusion time of 0.11 s.

Figure 3.17 gives a more detailed view of the fusion time of JPDA without clustering and deduplication. The fusion time and the number of objects are also split into the measurements of the different sensors.

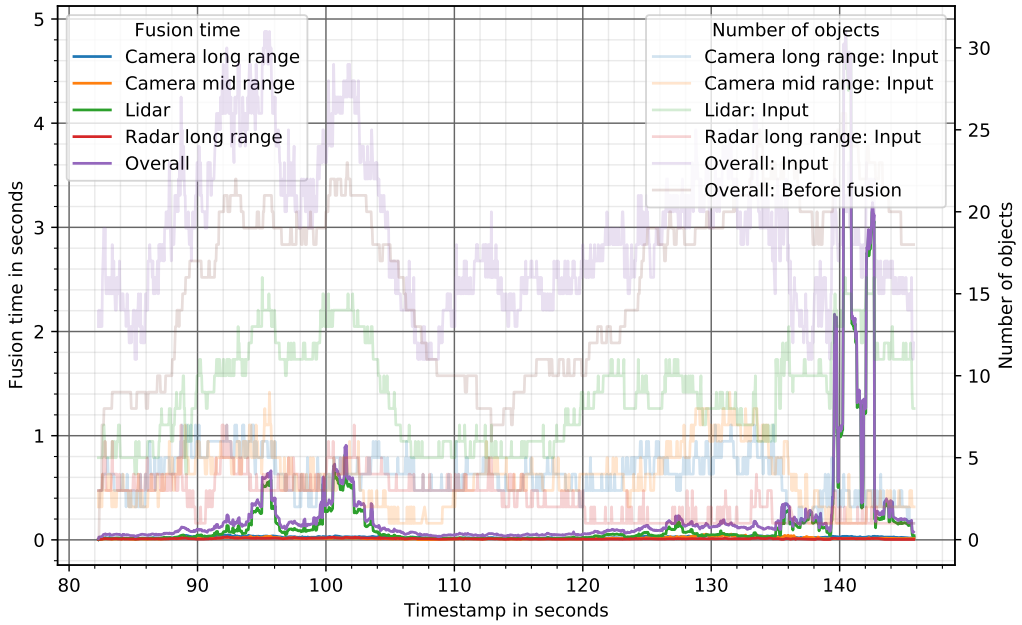

Figure 3.17: JPDA without clustering and deduplication: Fusion time at each timestamp. The number of objects is added as an extra information in lighter lines.

At the time of the higher fusion time, there are many objects to fuse. The first two peaks are at the area with much oncoming traffic and some parked vehicles (Figure 3.4). The much higher peak at the end is between the end of the congestion (Figure 3.6) and the begin of the highway entrance (Figure 3.7). Both share that there are many close vehicles for fusion, which is also indicated by the number of input objects in lighter lines in the graph. At the higher peak at the end, additionally, many new objects from the right side of the intersection are visible for the first time. However, the number of input objects alone cannot fully explain the much higher fusion time at JPDA without clustering.

For JPDA the number of joint association event hypotheses has a signifi-

cant impact on the computing performance. Figure 3.18 shows again the fusion time of JPDA without clustering and deduplication, but this time the number of hypotheses is added. Again, the fusion time and the number of hypotheses are split into the measurements of the different sensors.

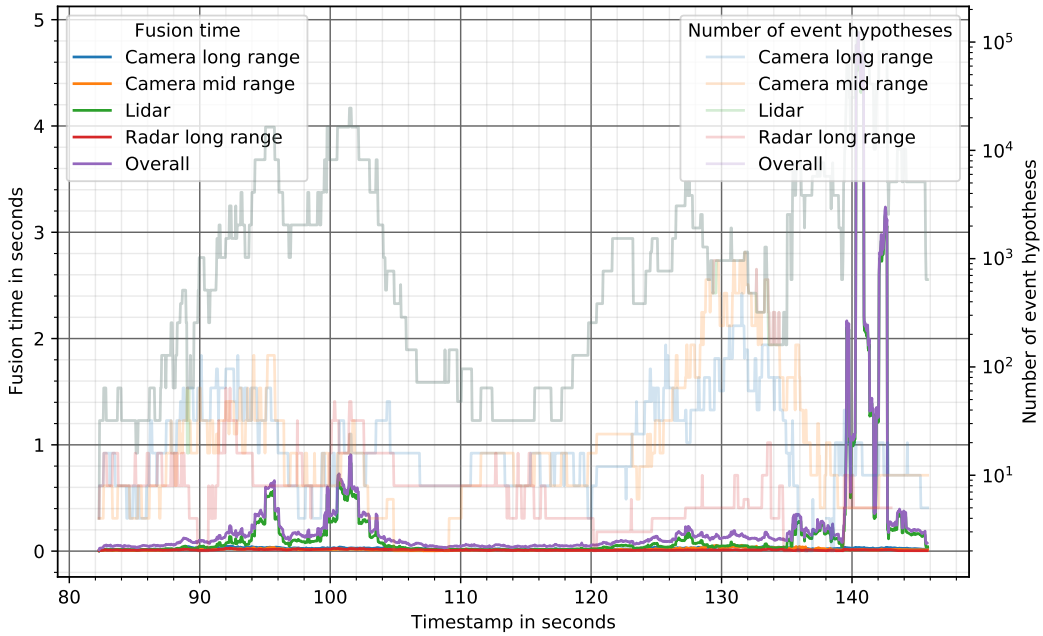

Figure 3.18: JPDA without clustering and deduplication: Fusion time at each timestamp. The number of hypotheses in logarithmic scaling is added as an extra information in lighter lines.

The fusion time and the number of event hypotheses are directly correlated. At the time of the higher fusion time, the number of event hypotheses is significantly higher than at lower fusion times. The highest fusion time with 5 s for one single timestamp is when there are more than 120 000 event hypotheses. By looking at the fusion times of the different sensors, it can be observed that the lidar sensor needs most fusion time. This is because the lidar with its 360° field-of-view detects many objects. Therefore many event hypotheses are created.

Figure 3.19 shows the same information for JPDA with clustering but still without deduplication. The number of event hypotheses is the maximum

number in one single cluster and this time in linear and not in logarithmic scaling.

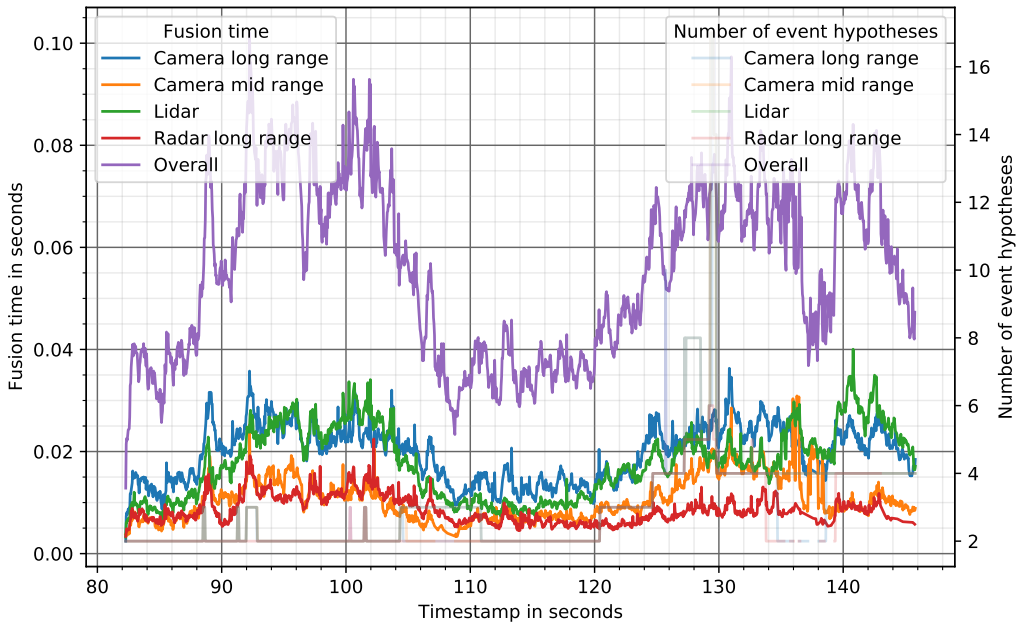

Figure 3.19: JPDA with clustering and without deduplication: Fusion time at each timestamp. The maximum number of hypotheses in one single cluster in linear scaling is added as an extra information in lighter lines.

The fusion time is significantly lower at the critical parts from before. By clustering, the number of event hypotheses always stays in an acceptable range. The maximum number of event hypotheses in one single cluster is 17. The maximum sum of event hypotheses of all clusters is 33. Without clustering, the maximum number of event hypotheses is 122 880. With clustering at the very same time, the measurements are split into 15 clusters with one time 4 hypotheses, one time 3 hypotheses, and 13 times 2 hypotheses. The sum of the hypotheses is highly reduced to 33.

Based on the fusion time at each timestamp, various statistics can be created. These statistics are summarized with boxplots and compared in Figure 3.20. Figure 3.21 zooms into the more relevant area.

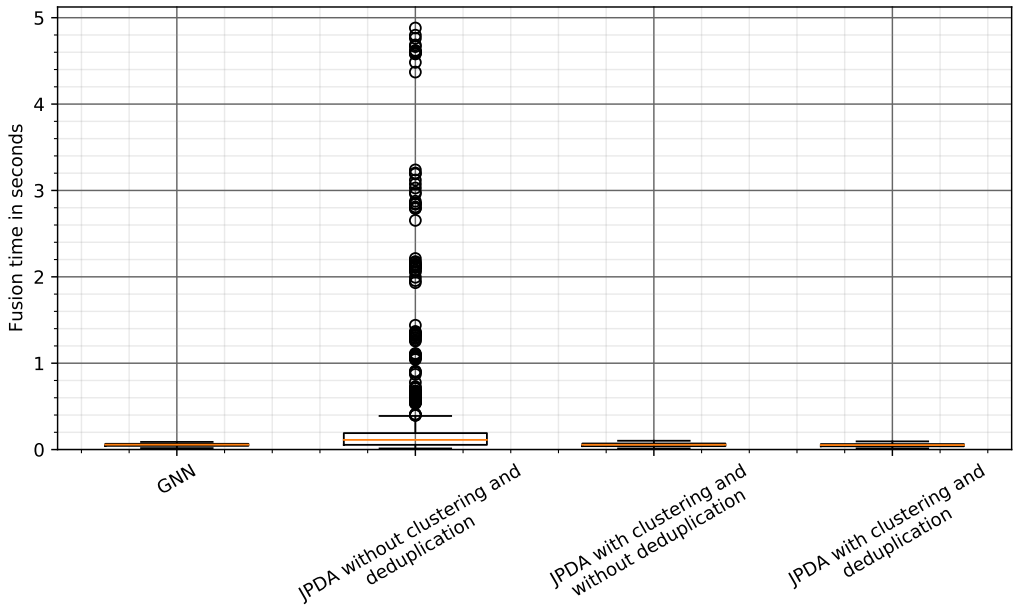

Figure 3.20: Comparison of fusion methods with boxplots of the fusion time

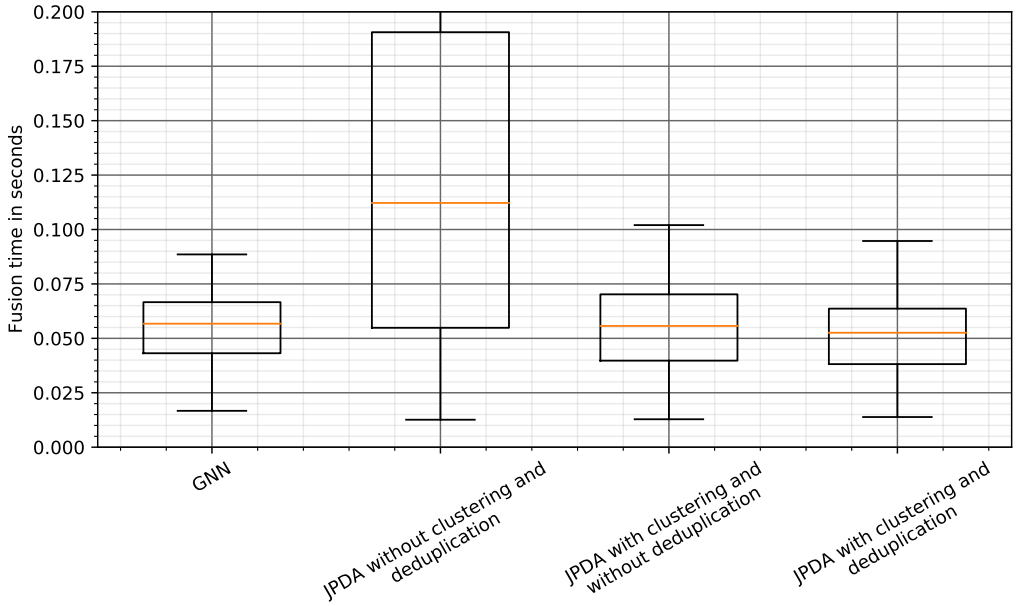

Figure 3.21: Comparison of fusion methods with boxplots of the fusion time (zoom)

The computing performance of JPDA without clustering and deduplication is far worse than the other ones. The other fusion variants have a similar median fusion time. At the worst case, JPDA with clustering but without deduplication has a fusion time slightly above 0.1 s. The other two variants stay always below. Therefore, real-time performance, with, for example 10 Hz, would be possible.

The performance of JPDA with deduplication is slightly better than without deduplication. Removing the duplicate tracks results in fewer tracks to fuse. This saves computing time.

3.6.2 Fusion Performance

The fusion performance is evaluated for each scenario except the urban scenario (for the scenarios see Section 3.3) with the metrics described in Section 3.4.2.

Highway overtaking

Figure 3.22 shows the overall OSPA metric. Figure 3.23 shows the OSPA metric split into localization, cardinality, and labeling error.

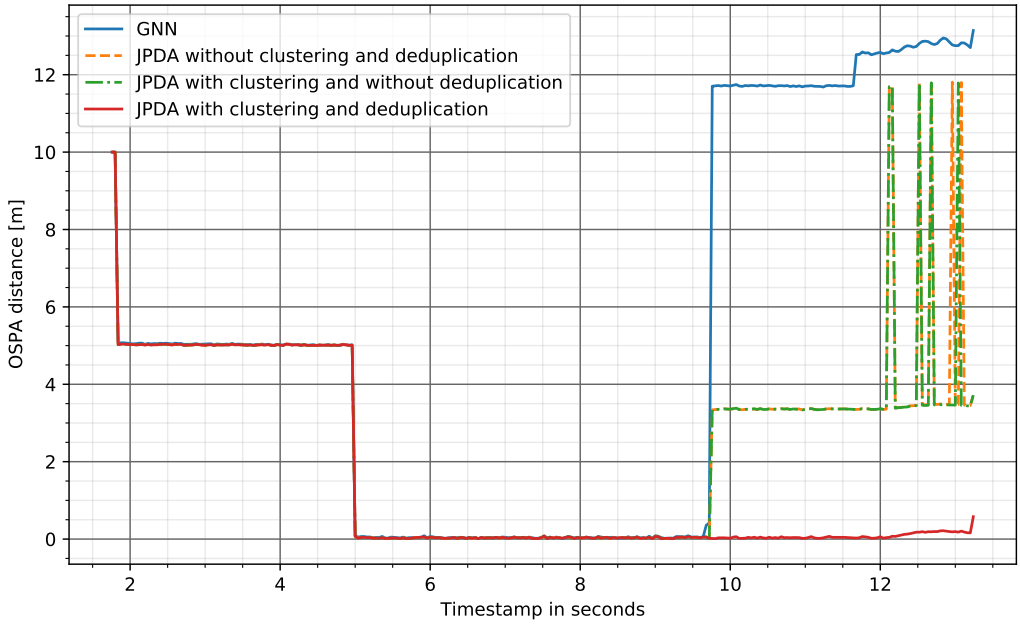

Figure 3.22: Scenario highway overtaking: Comparison of overall OSPA distance

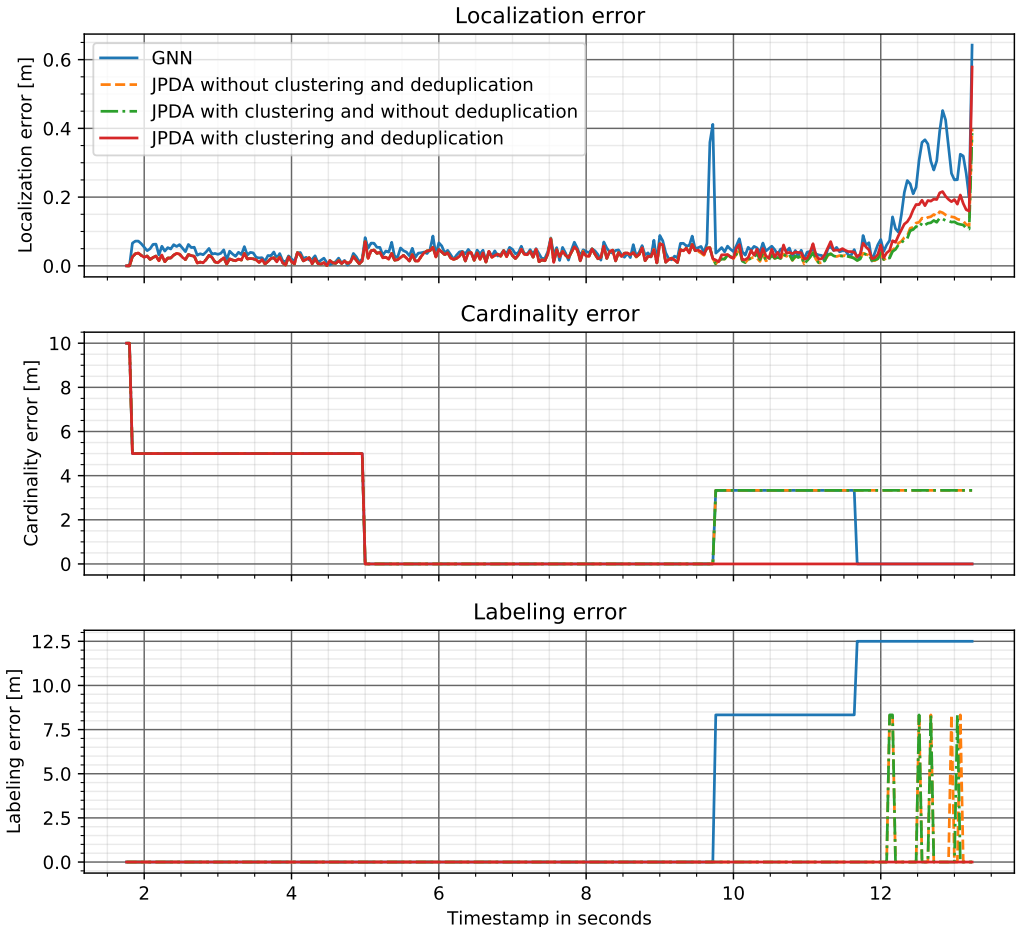

Figure 3.23: Scenario highway overtaking: Comparison of OSPA distance split into localization, cardinality, and labeling error

Figure 3.24 shows the different error types at each timestamp.

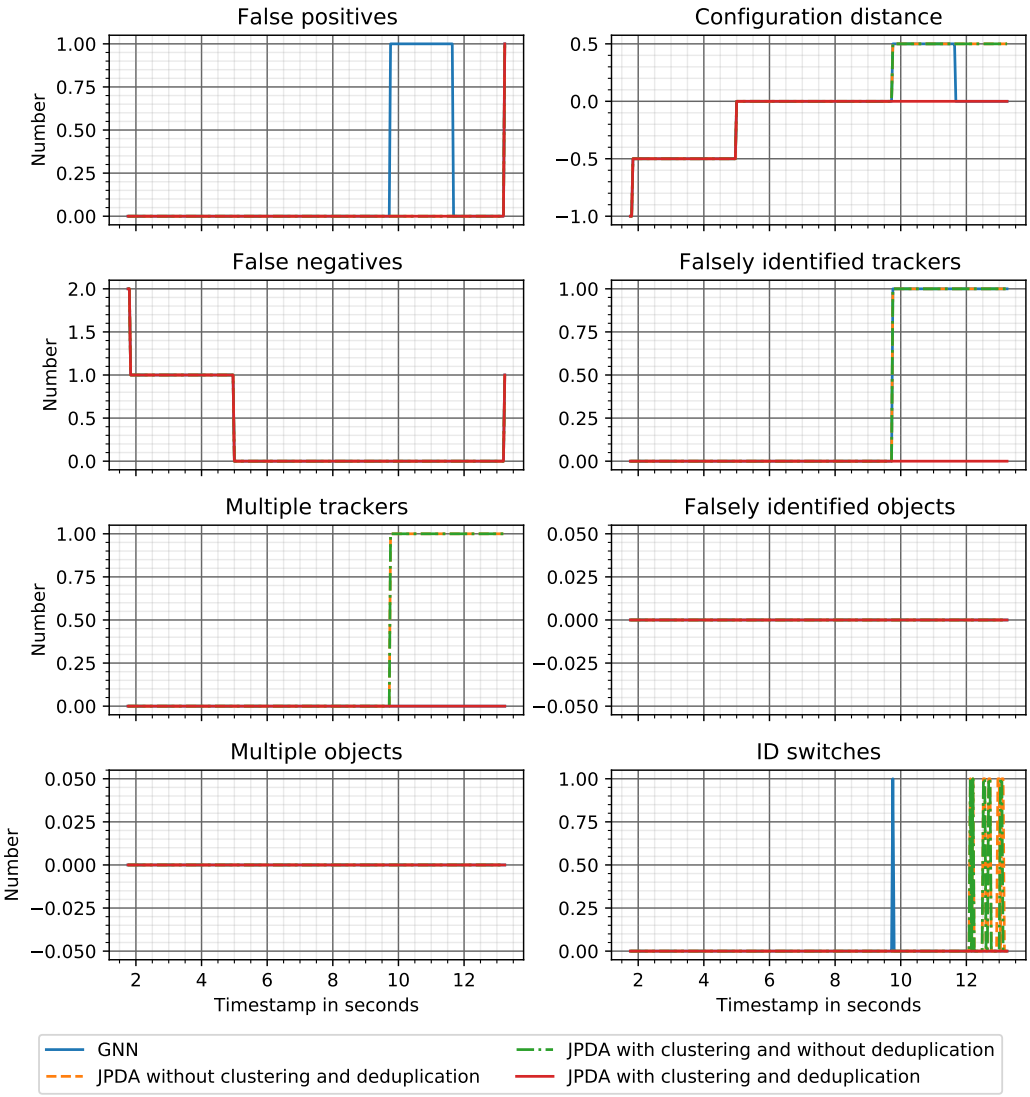

Figure 3.24: Scenario highway overtaking: Different error types at each timestamp

Time-Based Analysis At the beginning of the scenario, the OSPA metric for cardinality, the false negatives, and the configuration distance show that there is one missing track. This is the second car in front. However, this is no real error as the second car can only be detected after the first car

changes its lane. After the first car had changed its lane, the second car was detected correctly from every sensor fusion variant. At around 10 s there is an additional track created from GNN and JPDA variants without deduplication. This is caused by a clutter object. Track validation did not help in that case because the clutter object was very close to the actual track. After the track was initially created (non-confirmed), measurements from the actual track got associated with the new track and it got confirmed.
For GNN, the new track caused by clutter became the track following the actual object. This introduces the error at the OSPA labeling and the falsely identified tracker. Later at around 12 s, the original track following the object is destructed. The OSPA labeling error increases again because the number of tracks, which is used as normalization, is reduced.
For JPDA variants without deduplication is for the same clutter object also a track created and confirmed. The newly created track does not die out this time because there is always a low probability that the measurement can be assigned to that track. Eventually, the newly created track caused by clutter and the original track both follow the actual object (error at multiple trackers). At the very end, there are many ID switches and an often changing labeling error observable. The track caused by clutter and the original track overlap now completely, and the track closer to the ground truth often changes.
JPDA with clustering and deduplication never has any confirmed track caused by clutter. Before confirming the same track as at JPDA without deduplication, it is removed because it overlaps with the longer existing original track.
The OSPA localization error is similar for all JPDA variants. At GNN, the localization error at the cut-through is worse because the clutter object is wrongly fused to the track. At the end, the error increases because the number of tracks, which is again used as normalization, is reduced.

Overall metrics Figure 3.25 shows a comparison of the overall metrics calculated from the entire dataset including the MOTA score.

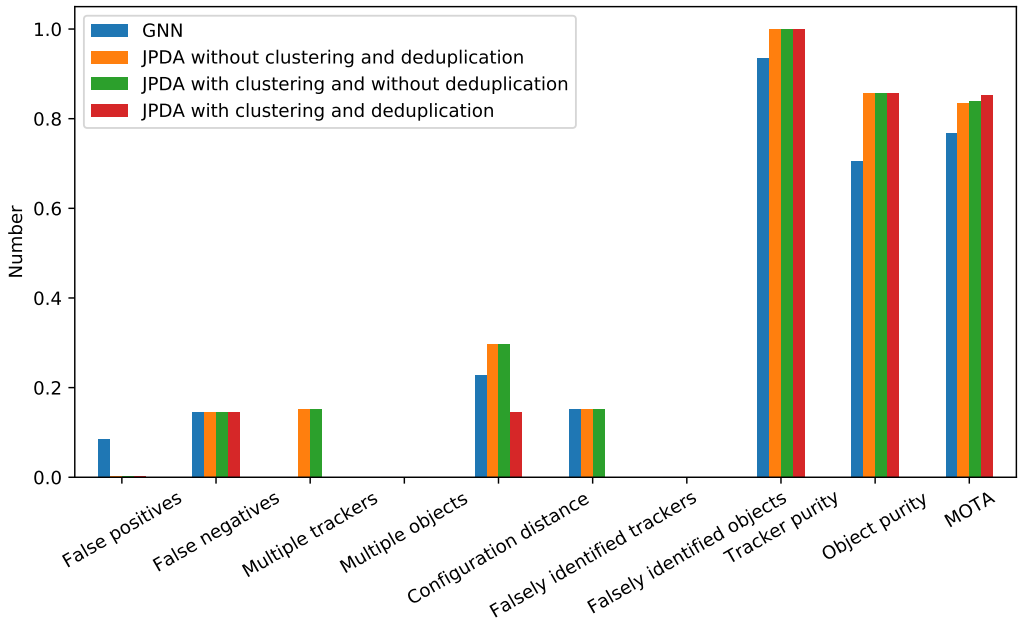

Figure 3.25: Scenario highway overtaking: Comparison of the overall metrics including the MOTA score. For the three rightmost metrics, the higher value is better. For the other, the lower is better.

The overall metrics reflect the findings from the previous paragraph. Additionally, the tracker purity and the object purity show that at GNN the track caused by clutter eventually follows the ground truth, and the original track dies out.
JPDA with clustering and deduplication is best in every metric.

Highway cut-through between

Figure 3.26 shows the overall OSPA metric. Figure 3.27 shows the OSPA metric split into localization, cardinality, and labeling error.

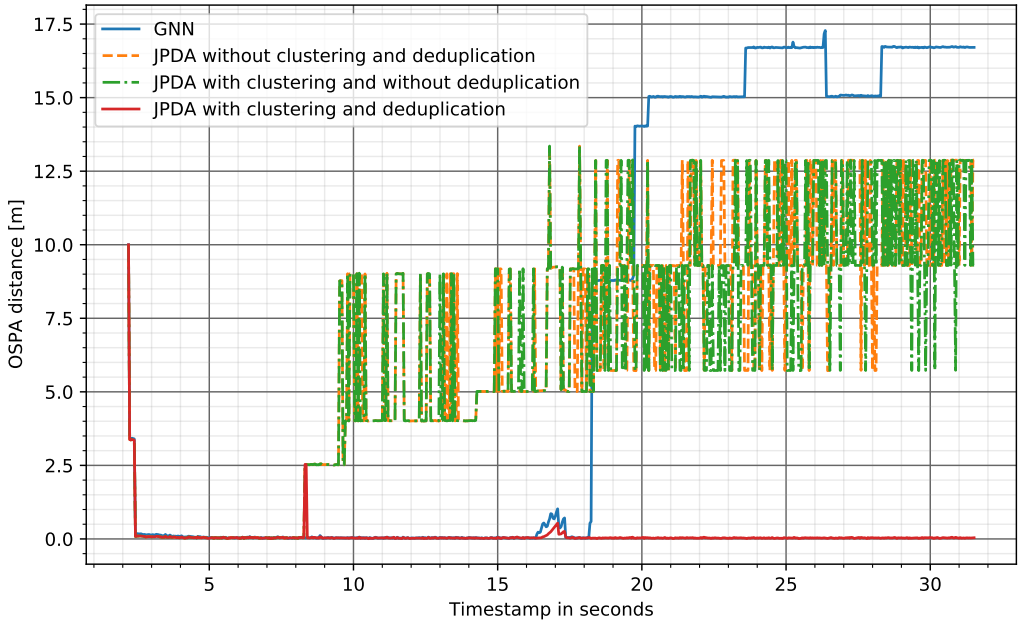

Figure 3.26: Scenario highway cut-through between: Comparison of overall OSPA distance

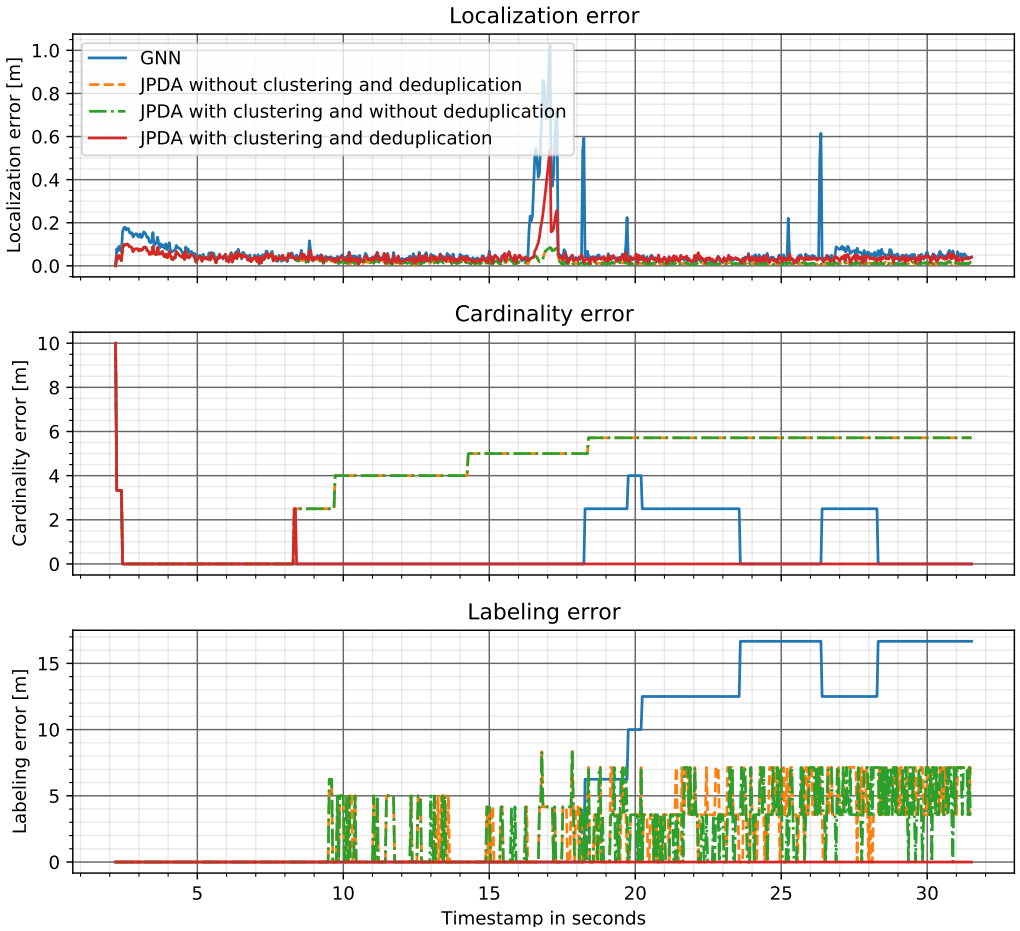

Figure 3.27: Scenario highway cut-through between: Comparison of OSPA distance split into localization, cardinality, and labeling error

Figure 3.28 shows the different error types at each timestamp.

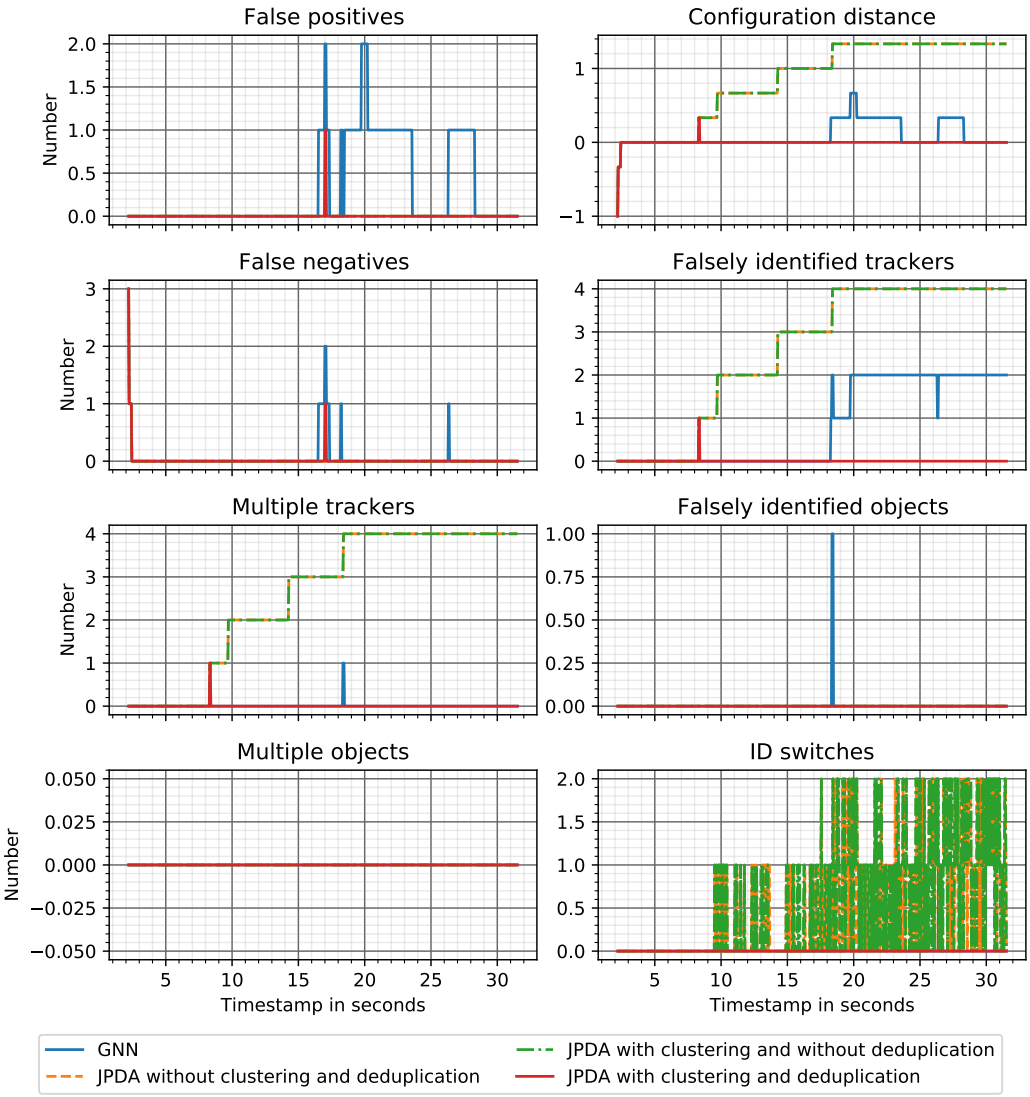

Figure 3.28: Scenario highway cut-through between: Different error types at each timestamp

Time-Based Analysis At the beginning, all objects are correctly tracked. At around 8 s, a clutter object causes that the JPDA variants create an additional track (error at OSPA cardinality error, multiple trackers, and configuration distance). JPDA with clustering and deduplication deletes the new track

after a very short time. For JPDA variants without deduplication the same behavior happens a few times between 8 s and 18 s.
The cut-through happens at around 17 s. The OSPA localization error increases for GNN, and JPDA with clustering and deduplication because the red car in front is not visible while the green car is cutting through (see Figure 3.10). For JPDA variants without deduplication, the OSPA localization error is not so much increased because the localization error is normalized by the number of tracked objects, which are higher than for the other variants. After the cutting-through, the JPDA variants again follow the correct actual object. On the contrary, at GNN the original track of the red car gets lost while the green car is cutting through. Then, when the red car is reappearing again, a new track is created (OSPA labeling error, falsely identified objects, and falsely identified trackers increases). The original track is getting lost because at GNN a clutter object is fully fused to the original track while the green car is cutting through. At JPDA, variants without deduplication cause again many ID switches because multiple objects nearly overlap and the closest one to the ground truth often changes.
There are again some points at GNN where the OSPA localization error is worse because a clutter object is fully fused to the track.

Overall Metrics Figure 3.29 shows a comparison of the overall metrics calculated from the entire dataset including the MOTA score.

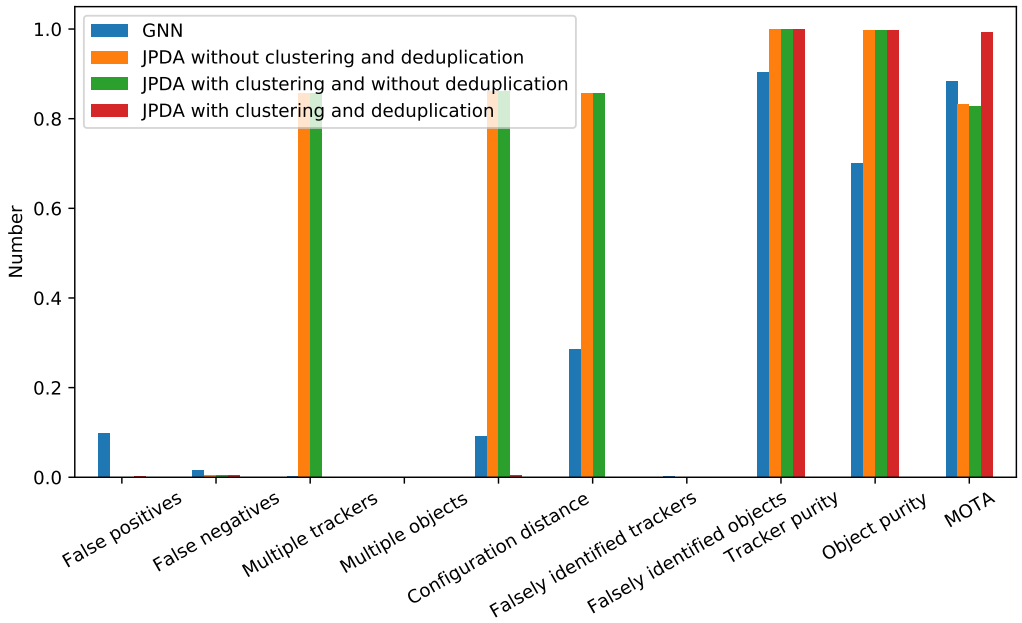

Figure 3.29: Scenario highway cut-through between: Comparison of the overall metrics including the MOTA score. For the three rightmost metrics, the higher value is better. For the other, the lower is better.

The overall metrics show that JPDA with clustering and deduplication again outperforms the other variants. JPDA with or without clustering but without deduplication shows the same performance. GNN performs between the other variants.

Highway cut-through front 1

Figure 3.30 shows the overall OSPA metric. Figure 3.31 shows the OSPA metric split into localization, cardinality, and labeling error.

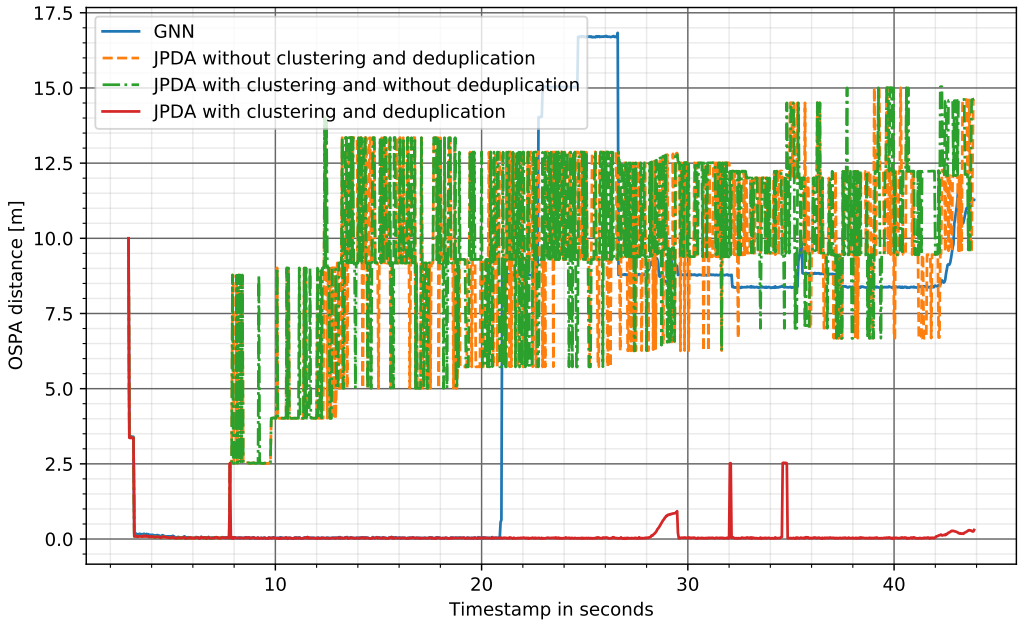

Figure 3.30: Scenario highway cut-through front 1: Comparison of overall OSPA distance

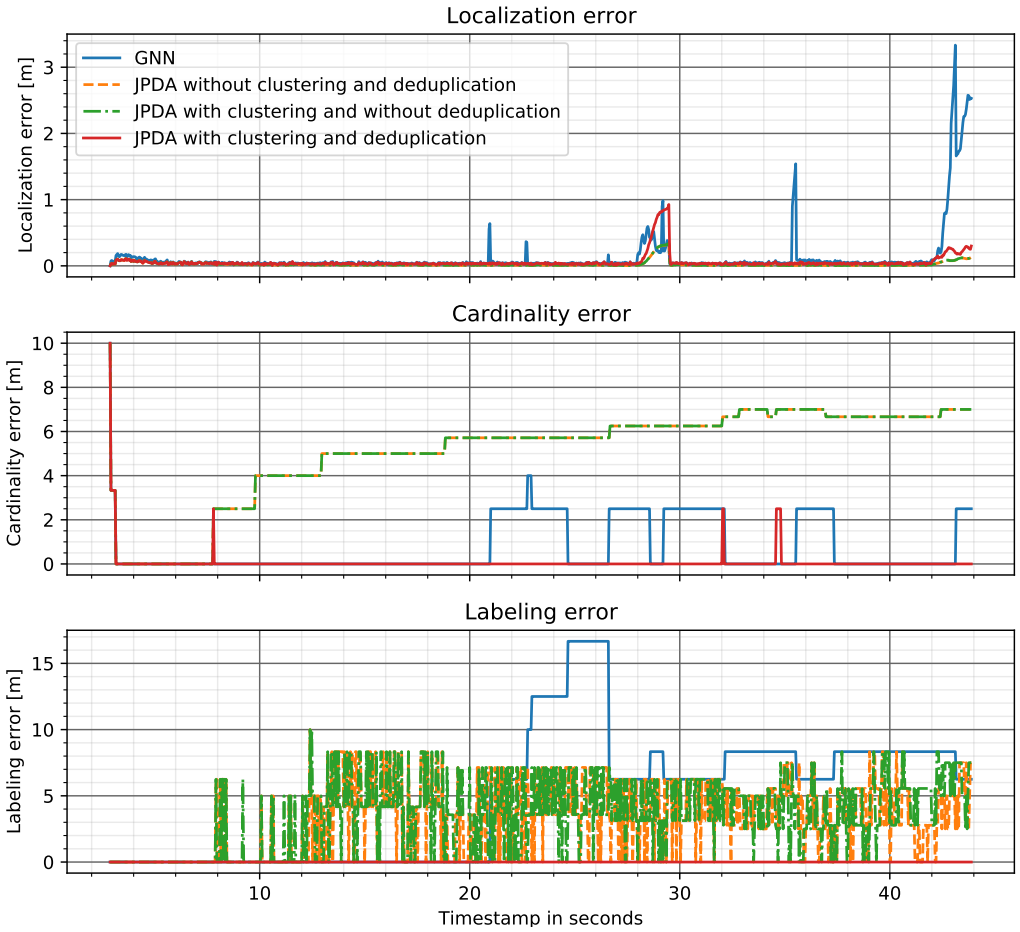

Figure 3.31: Scenario highway cut-through front 1: Comparison of OSPA distance split into localization, cardinality, and labeling error

Figure 3.32 shows the different error types at each timestamp.

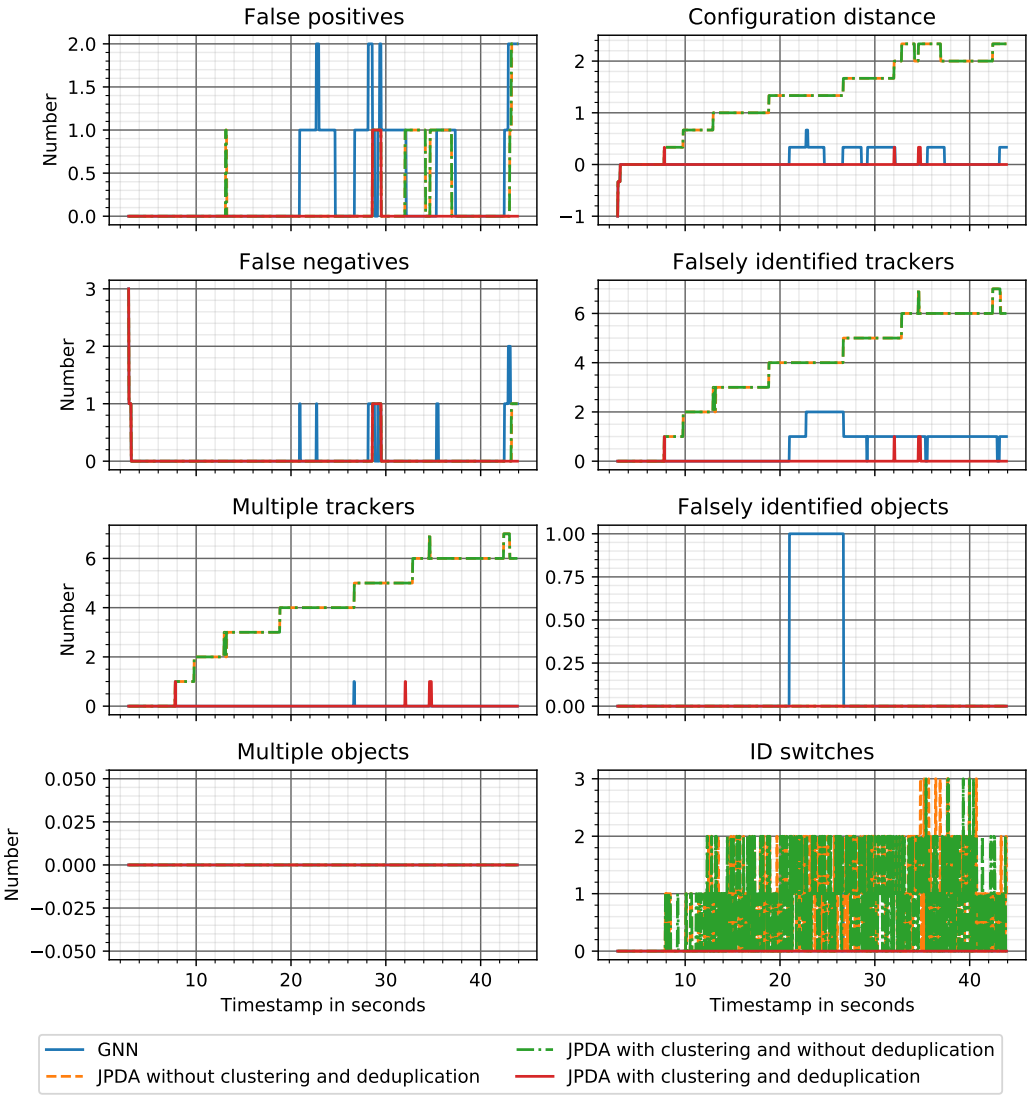

Figure 3.32: Scenario highway cut-through front 1: Different error types at each timestamp

Time-Based Analysis The behavior is similar to the previous scenario. At the beginning, many tracks caused by clutter are again created from the JPDA variants without deduplication. Then happens the cut-trough between 28 s and 30 s. This increases the OSPA localization error. Again, GNN fuses

a clutter object to the original track which causes the track of the green car to get lost, and a new track is created for the reappearing green car.

Overall Metrics Figure 3.33 shows a comparison of the overall metrics calculated from the entire dataset including the MOTA score.

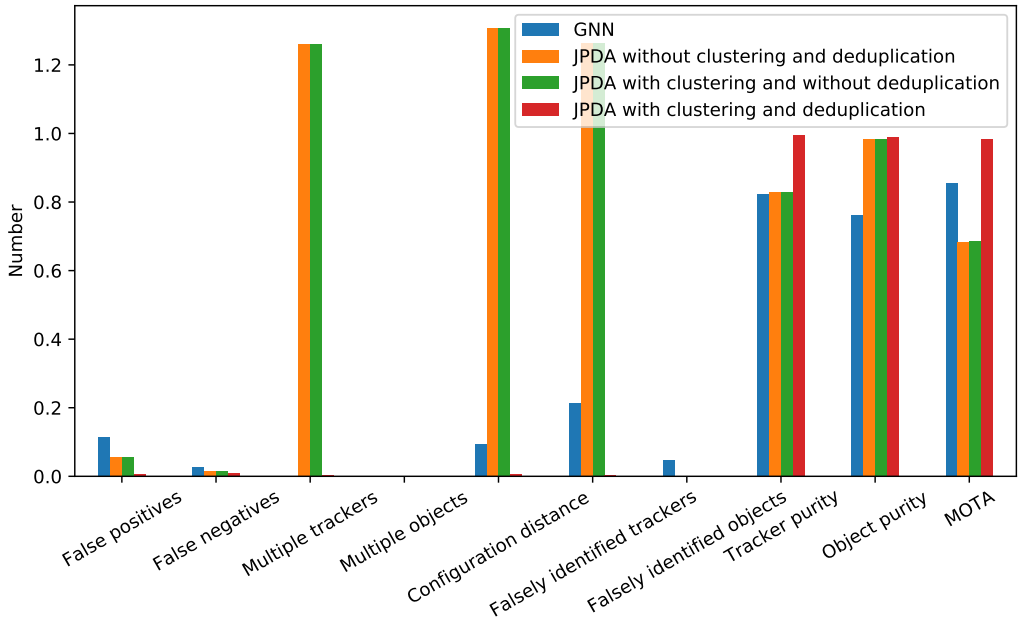

Figure 3.33: Scenario highway cut-through front 1: Comparison of the overall metrics including the MOTA score. For the three rightmost metrics, the higher value is better. For the other, the lower is better.

The results are similar as before. JPDA with clustering and deduplication outperforms the other variants, while GNN is mostly second best. JPDA with or without clustering but without deduplication shows the same performance.

Highway cut-through front 2

Figure 3.34 shows the overall OSPA metric. Figure 3.35 shows the OSPA metric split into localization, cardinality, and labeling error.

Figure 3.34: Scenario highway cut-through front 2: Comparison of overall OSPA distance

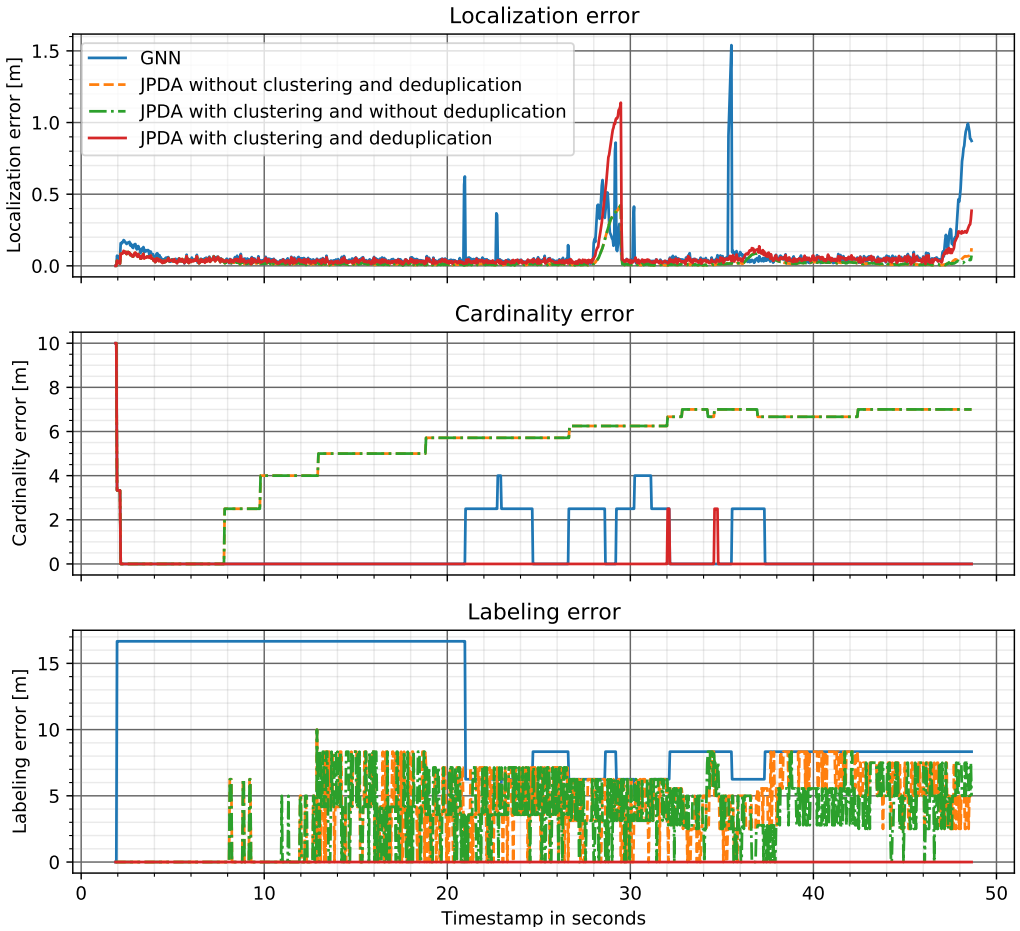

Figure 3.35: Scenario highway cut-through front 2: Comparison of OSPA distance split into localization, cardinality, and labeling error

Figure 3.36 shows the different error types at each timestamp.

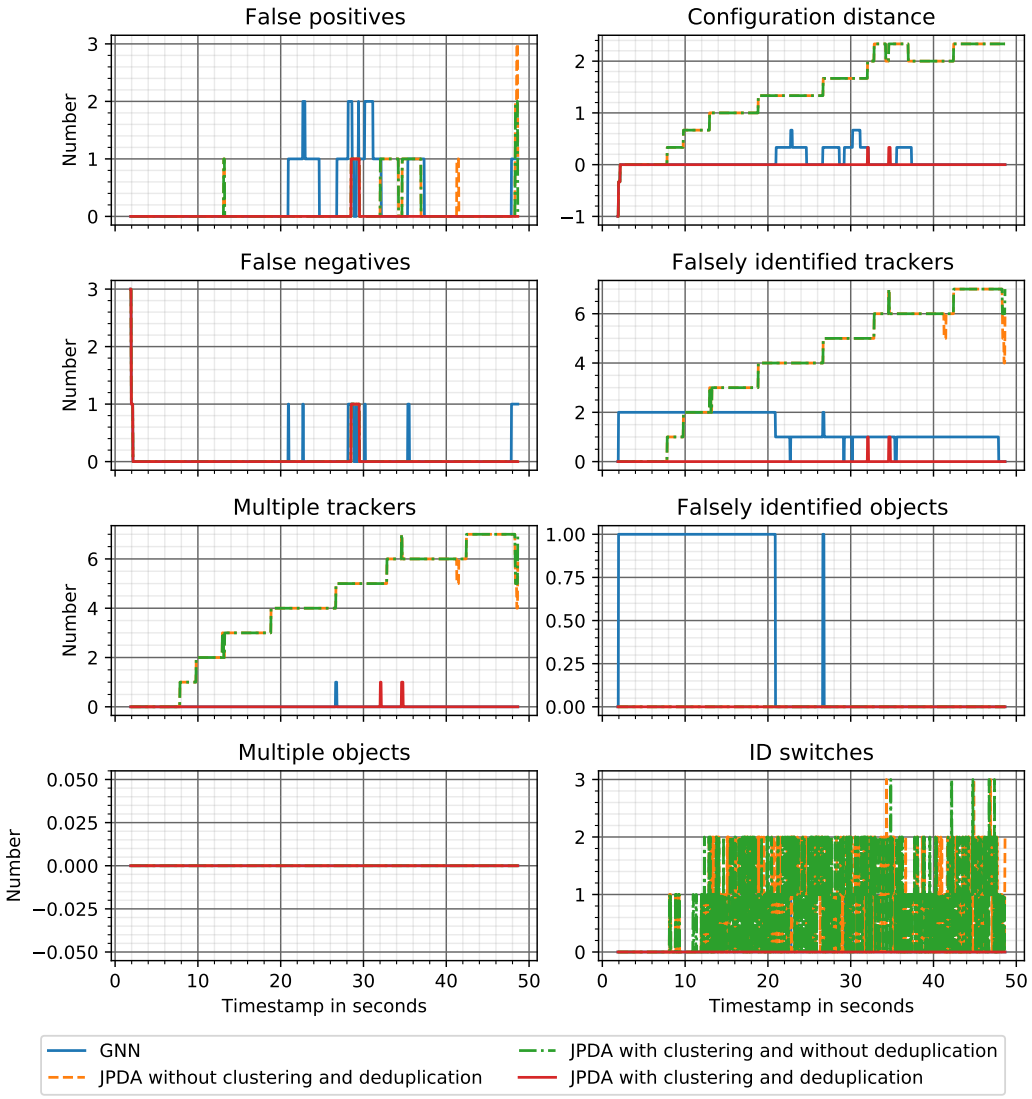

Figure 3.36: Scenario highway cut-through front 2: Different error types at each timestamp

Time-Based Analysis The scenario is the same as highway cut-through front 1 with an additional maneuver at the end. The results are also very similar although at first they look different. At GNN, there is, from the very beginning, an OSPA labeling error and a falsely-identified-object error.

It is the very same track as before but at this longer scenario the other track, which is later created by the reappearing green car, is assigned to the ground truth because the ground-truth-to-track assignment for evaluation prefers the longer existing track (see Section 3.4.2). Therefore, the track at the beginning is identified as an error and then later, when the other track is created, the error gets reduced. For the rest, the fusion variants show the same behavior as in the previous scenario.

Overall Metrics Figure 3.37 shows a comparison of the overall metrics calculated from the entire dataset including the MOTA score.

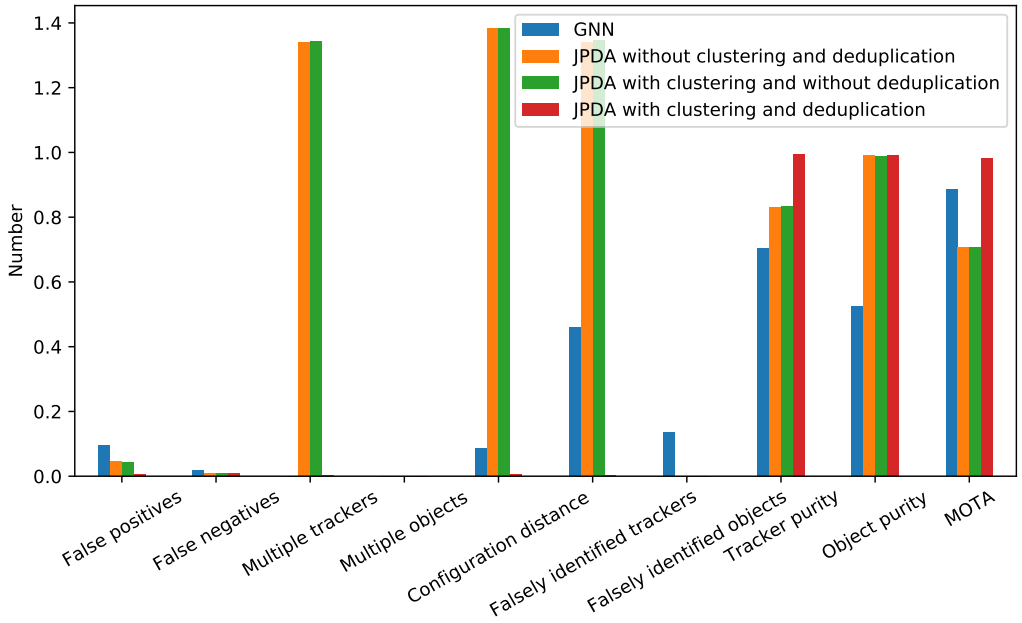

Figure 3.37: Scenario highway cut-through front 2: Comparison of the overall metrics including the MOTA score. For the three rightmost metrics, the higher value is better. For the other, the lower is better.

The results are similar to the previous scenario. A difference is at GNN where the tracker purity and object purity decreased because the falsely identified trackers and falsely identified objects increased.

3.7 Discussion

The results about the computing performance showed that JPDA variants with clustering have similar performance to GNN. Without clustering, the enumeration of event hypotheses is vast. This makes real-time performance in a dense environment with many detected vehicles impossible. The dependence on the detected vehicles showed that the sensor setup with the different properties of each sensor strongly influences the computing performance. For example, a lidar sensor mounted on the rooftop has a field of view of 360° and can therefore detect many vehicles. The fusion time of the measurements from the lidar sensor was always the highest.

In terms of fusion performance, JPDA with clustering and deduplication exceeds all other variants. There were 4 different scenarios tested and all of them were evaluated with 10 different overall metrics. GNN was best at 9 of the overall 40 tests, JPDA without clustering and deduplication at 18, JPDA with clustering but without deduplication at 17, and JPDA with clustering and deduplication at 34. The sum of the best tests is more than the overall 40 tests because if multiple variants have the same best test result, it is counted multiple times. At the MOTA score, which combines false positives, false negatives, and ID switches into one metric, JPDA with clustering and deduplication was always the best fusion variant. This is also reflected in the average MOTA score of all scenarios. For GNN this was 0.85, for JPDA without clustering and deduplication 0.76, for JPDA with clustering but without deduplication 0.77, and for JPDA with clustering and deduplication 0.95. The results also show that clustering does not harm the fusion performance. The results were mostly even the same. However, JPDA variants without deduplication are not usable. There are too many duplicate tracks created that completely overlap.

Summarized, JPDA with clustering and deduplication is a feasible alternative to the current state of the art which is GNN. While the computing performance is similar, the fusion performance, especially in situations with many clutter objects, outperforms GNN.

4 Conclusion

To compare different data association methods in a structured way, this thesis has presented a general framework for the testing of sensor fusion. The framework has been applied to compare a non-Bayesian data association method, global nearest neighbor (GNN), which has acted as a reference, against Bayesian data association methods, three variants of joint probabilistic data association (JPDA). By this, the performance of alternative data association methods against the current state of the art has been evaluated.

The general framework consists of several parts. First, a simulation is used to provide a comparable environment. Second, a sensor setup, which is the input for the fusion, has been defined. Five relevant scenarios have been designed, which are used to evaluate the performance. One scenario includes many close vehicles in an urban environment to test the computing performance. The other four scenarios contain situations where a vehicle must be detected quickly or a vehicle is shortly occluded by another vehicle. These scenarios have been designed to evaluate the fusion performance. For example, after a vehicle is visible again, the algorithms should associate the new measurement to the already existing track and not create a new track. One single track should follow the vehicle. To challenge the data association, a high number of clutter objects are modeled. The evaluation is split into two parts, the computing performance and the fusion performance. The computing performance evaluates the real-time capability based on the median and maximum time needed to fuse the measurements at one timestamp. The fusion performance evaluates the quality of the data association and fusion. The used metrics are the OSPA metric for tracks, the MOTA score, and several standardized metrics based on traditional performance indicators like false positives and false negatives.

Within this general framework, the selected data association methods, GNN and three variants of JPDA, have been evaluated. The research has shown that the original variant of JPDA has no comparable computing and fusion performance to GNN. JPDA evaluates all possible measurement-to-track association. This can be vast in a dense environment with many close vehicles.

At areas with much traffic, the fusion time of a single measurement has been nearly 5 s. By clustering, this problem has been circumvented. Instead of enumerating all possible measurement-to-track associations, independent clusters are created. This has reduced the maximum fusion time to less than 0.1 s, which equals the reference, GNN. Analysis of the fusion performance has shown that the original variant of JPDA likely creates duplicate tracks in a high-cluttered environment. By simply removing the duplicate tracks based on the Euclidean distance between them, this issue has been resolved. To summarize, the results have shown that JPDA without clustering has unacceptable computing performance. With clustering, the fusion time has been similar to GNN and real-time performance is possible. Additionally, the original variant of JPDA likely creates duplicate tracks, although only one single track should follow one ground truth object. The resulting variant, JPDA with clustering and deduplication, has shown comparable computing performance and better fusion performance than GNN. For example, the MOTA score, which combines false positives, false negatives, and ID switches in a single metric, has been on average 12 % better than that of GNN.

However, real-world applications of JPDA at sensor fusion for automated vehicles are still missing. Further research might reveal additional benefits or drawbacks.

Additionally, analysis of other data association variants within the developed general framework could provide additional knowledge on choosing the right data association method for sensor fusion of automated vehicles. Multiple hypothesis tracking and the emerging field of sensor fusion based on random finite sets are interesting candidates for further testing.

In summary, this thesis has developed a general framework for the testing of sensor fusion and applied it to GNN and three variants of JPDA. The results have shown that JPDA with clustering and deduplication is a feasible alternative to the current state of the art, GNN. Real-time performance can be guaranteed and especially in high-cluttered environments the data association quality is superior to GNN.

Bibliography

[1] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. Santa Monica, CA: RAND Corporation, 2016. DOI: 10.7249/RR443-2.

[2] W. Gruel and J. M. Stanford, "Assessing the long-term effects of autonomous vehicles: A speculative approach," *Transportation Research Procedia*, vol. 13, pp. 18–29, 2016, Towards future innovative transport: visions, trends and methods 43rd European Transport Conference Selected Proceedings, ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2016.05.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352146516300035.

[3] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies," *Journal of Modern Transportation*, vol. 24, no. 4, pp. 284–303, Dec. 2016, ISSN: 2196-0577. DOI: 10.1007/s40534-016-0117-3. [Online]. Available: https://doi.org/10.1007/s40534-016-0117-3.

[4] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," 2015.

[5] S. Behere and M. Törngren, "A functional architecture for autonomous driving," in *Proceedings of the First International Workshop on Automotive Software Architecture*, ser. WASA '15, Montréal, QC, Canada: Association for Computing Machinery, 2015, pp. 3–10, ISBN: 9781450334440. DOI: 10.1145/2752489.2752491. [Online]. Available: https://doi.org/10.1145/2752489.2752491.

[6] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer, 2008, ISBN: 978-3-540-23957-4. DOI: 10.1007/978-3-540-30301-5. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5.

[7] The Autoware Foundation. (2020). "Autoware.AI," [Online]. Available: https://www.autoware.ai.

[8] The Autoware Foundation. (2020). "Autoware.Auto," [Online]. Available: https://www.autoware.auto.

[9] Baidu. (2020). "Apollo," [Online]. Available: https://apollo.auto.

[10] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems. Revised reprint.* English. SIAM - Society of Industrial and Applied Mathematics, 2012, 393 Seiten, ISBN: 978-1-611972-22-1.

[11] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. DOI: 10.1002/nav.3800020109. eprint: https://www.onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109. [Online]. Available: https://www.onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[12] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of The Society for Industrial and Applied Mathematics*, vol. 10, pp. 196–210, 1957.

[13] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. [Online]. Available: https://doi.org/10.1115/1.3662552.

[14] M. Grewal and A. Andrews, "Kalman filtering: Theory and practice using MATLAB," *New York: John Wiley and Sons*, vol. 14, Jan. 2001. DOI: 10.1002/9780470377819.

[15] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000. DOI: 10.1109/9.847726.

[16] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[17] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009. DOI: 10.1109/TAC.2009.2019800.

[18] B.-N. Vo, M. Mallick, Y. bar-shalom, S. Coraluppi, R. III, R. Mahler, and B.-T. Vo, "Multitarget tracking," *Wiley Encyclopedia*, pp. 1–25, Sep. 2015. DOI: 10.1002/047134608X.W8275.

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

[20] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[21] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009.

[22] D. Dunne, "Random finite set methods for multitarget tracking," Ph.D. dissertation, McMaster University, Canada, 2013. [Online]. Available: http://hdl.handle.net/11375/12941.

[23] R. P. S. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[24] M. Sualeh and G.-W. Kim, "Dynamic multi-LiDAR based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, 2019, ISSN: 1424-8220. DOI: 10.3390/s19061474. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1474.

[25] G. Thomaidis, L. Spinoulas, P. Lytrivis, M. Ahrholdt, G. Grubb, and A. Amditis, "Multiple hypothesis tracking for automated vehicle perception," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 1122–1127.

[26] B. Ristic, B. Vo, and D. Clark, "Performance evaluation of multi-target tracking using the OSPA metric," in *2010 13th International Conference on Information Fusion*, 2010, pp. 1–7.

[27] K. Smith, D. Gatica-Perez, J. Odobez, and Sileye Ba, "Evaluating multi-object tracking," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, 2005, pp. 36–36.

[28] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, *MOT16: A benchmark for multi-object tracking*, 2016. arXiv: 1603.00831 [cs.CV].

[29] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993. DOI: 10.1049/ip-f-2.1993.0015.

[30] L. Lin, Y. Bar-Shalom, and T. Kirubarajan, "Data association combined with the probability hypothesis density filter for multitarget tracking," in *Signal and Data Processing of Small Targets 2004*, O. E. Drummond, Ed., ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 5428, Aug. 2004, pp. 464–475. DOI: 10.1117/12.542218.

[31] K. Panta, B. Vo, and S. Singh, "Novel data association schemes for the probability hypothesis density filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 2, pp. 556–570, 2007.

[32] Y. Wang, Z. Jing, and S. Hu, "Data association for PHD filter based on MHT," Aug. 2008, pp. 1–8.

[33] D. E. Clark and J. Bell, "Data association for the PHD filter," in *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2005, pp. 217–222.

[34] D. Clark and J. Bell, "Multi-target state estimation and track continuity for the particle PHD filter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 43, pp. 1441–1453, Nov. 2007. DOI: 10.1109/TAES.2007.4441750.

[35] K. Panta, Ba-Ngu-Vo, and D. E. Clark, "An efficient track management scheme for the Gaussian-mixture probability hypothesis density tracker," in *2006 Fourth International Conference on Intelligent Sensing and Information Processing*, 2006, pp. 230–235.

[36] T. Wood, D. Clark, and B. Ristic, "Efficient resampling and basic track continuity for the SMC-PHD filter," *Proceedings of Cognitive Systems with Interactive Sensors, Crawley, UK*, 2010.

[37] D. Dunne and T. Kirubajaran, "Weight partitioned probability hypothesis density filters," *Fusion 2011 - 14th International Conference on Information Fusion*, Jan. 2011.

[38] Z. Wang, Y. Wu, and Q. Niu, "Multi-sensor fusion in automated driving: A survey," *IEEE Access*, vol. 8, pp. 2847–2868, 2020.

[39] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–6.

[40] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, 1998.

[41] K. Jo, M. Lee, J. Kim, and M. Sunwoo, "Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 460–476, 2017.

[42] VIRES. (2020). "VIRES Virtual Test Drive," [Online]. Available: `https://vires.mscsoftware.com`.

[43] VIRES. (2020). "OpenDRIVE," [Online]. Available: `http://www.opendrive.org`.

[44] T. Hanke, N. Hirsenkorn, C. van Driesten, P. Garcia Ramos, M. Schiementz, S. Schneider, and E. Biebl, *Open Simulation Interface: A generic interface for the environment perception of automated driving functions in virtual scenarios.* 2017. [Online]. Available: `https://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/` (visited on 09/26/2020).

[45] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 982–988. DOI: `10.1109/ITSC.2015.164`.

[46] T. Hansen, "Kombinierte Längs- und Quertrajektorienplanung für automatisierte Fahrstreifenwechsel," Ph.D. dissertation, Technische Universität, Darmstadt, 2018. [Online]. Available: `http://tuprints.ulb.tu-darmstadt.de/8081/`.