



HAL
open science

Cooperative Perception Integrity for Intelligent Vehicles

Antoine Lima

► **To cite this version:**

Antoine Lima. Cooperative Perception Integrity for Intelligent Vehicles. Robotics [cs.RO]. UTC - Université de Technologie de Compiègne rue du Dr Schweitzer. Université de Technologie de Compiègne (UTC), Compiègne, FRA., 2023. English. NNT : . tel-04671509v1

HAL Id: tel-04671509

<https://hal.science/tel-04671509v1>

Submitted on 1 Jul 2024 (v1), last revised 15 Aug 2024 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Cooperative Perception Integrity for Intelligent Vehicles

Antoine Lima

PhD Thesis Prepared at the Heudiasyc Laboratory, UMR
UTC/CNRS 7253

Defended on the Third of May, 2023

Specialty: Automatics & Robotics

Committee:

Reviewers	Romuald Aufrère	Univ. Clermont Auvergne
	Rémi Boutteau	Univ. de Rouen Normandie
Examiners	Franck Davoine	Univ. de Technologie de Compiègne
	Joelle Hage	Univ. de Technologie de Compiègne
	Fawzi Nashashibi	Inria Paris Rocquencourt
	Clément Zinoune	Renault
Supervisors	Philippe Bonnifait	Univ. de Technologie de Compiègne
	Véronique Cherfaoui	Univ. de Technologie de Compiègne



Contents

Acknowledgments	4
Abstract	5
1 General Introduction	11
1.1 Intelligent Vehicles	11
1.1.1 Levels of Automation	11
1.1.2 Autonomous Navigation Stack	12
1.1.3 Environment Perception	12
1.1.4 Cooperative Perception	13
1.2 Integrity	15
1.2.1 General Definition	15
1.2.2 Localization Integrity	16
1.2.3 Perception Integrity	17
1.3 Objectives	18
1.4 Manuscript Organization	19
2 Methods and Tools for Decentralized Data Fusion	21
2.1 Introduction	21
2.2 Symbolic Information and Belief Functions	22
2.2.1 Representation	22
2.2.2 Combination of Mass Functions	26
2.2.2.1 Conjunctive Combination	26
2.2.2.2 Disjunctive Combination	27
2.2.2.3 Cautious Combination	28
2.2.2.4 Partially Overlapping Fusion	28
2.2.3 Discounting	28
2.2.4 Conclusion	29
2.3 Metric Representation and State Filtering	29
2.3.1 Random State Vectors	29
2.3.2 State Filtering	32
2.3.2.1 Kalman Filtering	32
2.3.2.2 Extensions to the Kalman Filter	35
2.3.2.3 Informational Filtering	36
2.3.3 Covariance Intersection Filtering	36
2.3.4 Split Covariance Intersection Filtering	39
2.4 Analysis of Covariance Intersection Filters	40

2.4.1	CI Filtering Comparison with Kalman Filtering	41
2.4.2	Convergence Issues with Similarly Shaped Observation Co- variances	43
2.4.3	Slow Convergence with Partial Measurement	43
2.4.4	SCI Comparison with a Kalman-CI Combination	46
2.4.5	Tuning SCIF Evolution and Observation Models	47
2.5	Conclusion	48
3	Sensor Processing and Tracking	51
3.1	Introduction	51
3.2	Objects and Free Space Detection	51
3.2.1	Sensor Pre-Processing	52
3.2.2	Model Based Object Detection	55
3.2.3	Deep Learning Based Object Detection	56
3.2.4	Free Space	58
3.3	Multi-Object Tracking	60
3.3.1	Data Association for Object Traking	60
3.3.2	Track Management	62
3.4	Perception Evaluation	64
3.4.1	Perception Ground Truth	64
3.4.2	Evaluation Metrics	65
3.5	Description of the Perception System Used in this Work	69
3.5.1	Experimental Setup	69
3.5.1.1	Hardware	69
3.5.1.2	Software	73
3.5.2	Cars and Traffic Signs Detection using LiDAR	74
3.5.2.1	Sensor Tracking	76
3.5.2.2	Track Management	77
3.6	Evaluation	78
3.6.1	Evaluation of Sign Detection	78
3.6.2	Evaluation of Car Detection	80
3.7	Conclusion	82
4	Cooperative Perception in a Trustworthy Network	83
4.1	Introduction	83
4.2	Review of Cooperative Perception	84
4.2.1	Communication for Intelligent Transportation Systems	84
4.2.1.1	Medium	84
4.2.1.2	Messages	84
4.2.1.3	Contents of Cooperative Perception Messages	86
4.2.1.4	Security in Vehicular Networks	87
4.2.2	Cooperative Track-To-Track Fusion	87
4.2.2.1	Cooperative Fusion Architectures	87
4.2.2.2	Cooperative State Filtering	88
4.2.2.3	Spatial Alignment	89
4.2.2.4	Temporal Alignment	92
4.2.2.5	Out-Of-Sequence Observations	92

4.2.3	Evaluation Methods for Cooperative Perception	94
4.3	Fusion of Multiple Points of View	95
4.3.1	Generic Fusion Architecture	96
4.3.2	Managing the Detectability of Multiple Sources	96
4.3.2.1	Definition of Detectability	97
4.3.2.2	Computation of Detectability	98
4.3.2.3	Fusion of Detectability Grids	99
4.3.2.4	Object Detectability	103
4.3.3	Similarity between Objects	103
4.3.4	Estimating the Existence of Tracked Objects	106
4.4	Evaluation of Cooperative Perception	108
4.4.1	Evaluation Methodology	108
4.4.1.1	Global and Local Evaluations	108
4.4.1.2	Datasets with Ground-Truth	108
4.4.1.3	Evaluation Metrics	109
4.4.2	Study of the Added-Value of Cooperative Perception	114
4.4.3	Study of the Contribution of Detectability for Cooperative Perception	118
4.5	Conclusion	119
5	Estimation of Trust in Cooperative Peers	121
5.1	Introduction	121
5.2	Review of Trust in Intelligent Vehicles	121
5.2.1	Misbehavior Detection	122
5.2.2	Aggregation	123
5.3	Trust Estimation and Use for Data Fusion	124
5.3.1	Fusion Architecture with Trust Management	124
5.3.2	Evidential Estimation of Trust	124
5.3.3	Coherency	125
5.3.3.1	Object Detectability	126
5.3.3.2	Attribute Coherency	126
5.3.3.3	Spatial Coherency	127
5.3.4	Consistency	128
5.3.5	Confirmation	129
5.3.5.1	Object Similarity	130
5.3.5.2	Object Dissimilarity	130
5.3.5.3	Object-Free-Space Inconsistency	131
5.3.5.4	Free-Space Similarity	131
5.3.6	Summary of Trust Parameters	133
5.3.7	Trust-Aware Tracking	134
5.4	Experimental Evaluation	135
5.4.1	Added Value of Trust in Nominal Cases	136
5.4.2	Trust Estimation and Perception Performance in Case of Faults	138
5.4.3	Impact of Trust Parameters	141
5.5	Conclusion	145

6	General Conclusion	147
6.1	Conclusion	147
6.2	Contributions	148
6.3	Perspectives	149
	List of Figures	151
	List of Tables	158
A	Developments	163
A.1	Datasets	163
A.2	Perception	163
A.3	Tracking	164
A.4	Display	165
A.5	Special Processing	165
B	Cooperative Datasets	167
B.1	Roundabout	167
B.2	Intersection	168
B.3	Overtaking	168
C	Cooperative Ground-Truth	169
D	Study of the Combination Rule Used in Trust Estimation	171
E	Study of Consensus Building in Trust Estimation	173
	Bibliography	175

Acknowledgments

As an opening to this manuscript, I would like to thank all those who made it possible.

First and foremost, I would like to thank Philippe Bonnifait and Véronique Cherfaoui for supervising this thesis. Their guidance and the long and late but fruitful discussion we had helped me grow and complete this massive undertaking.

Secondly, I would like to thank Romuald Aufrère and Rémi Boutteau for reviewing this manuscript. The time they put into making sure this work was worth reading and their remarks greatly improved it. Many thanks also goes to Franck Davoine, Joelle Hage, Fawzi Nashashibi and Clément Zinoune for examining this work and for the discussion we had. I am especially thankful to Joelle and Fawzi that followed this thesis through the years. They helped me gain confidence in my work and focus my ideas.

Thirdly, I would like to thanks my Heudiasyc colleagues and friends: Baptiste, Rémy, Stéphane, Maxime&Maxime, Lyes, Anthony, Corentin, Jean-Benoist, Luca and Romain. Talking about science, work and other stuff helped me keep my head above the water. Also thanks to my ex-roommates Tobias and Adrien for their energy and making sure I was not only working.

Finally, my deepest thanks goes to Roxane, my parents and my brothers for emotionally supporting me for all these years, especially those last four.

Abstract

In order to navigate safely and comfortably, intelligent vehicles require highly reliable perception of their environment. Since on-board sensors are necessarily limited in range, and because their field of view can be obscured, an emerging solution is cooperative perception: vehicles share their perception with other vehicles via wireless communication.

Intelligent vehicles can thus communicate complex information over long distances. They see further and more completely than their sensors could ever allow. However, information from external sources must be treated with caution, as misleading information can lead to a dangerous situation. The sources of degradation of this information's "integrity" in the cooperative system must therefore be kept to a minimum. In this thesis, we study these sources and propose suitable methods for managing them and avoiding their propagation. Our work focuses in particular on the fusion of tracked objects, the representation of areas covered by perception systems and the management of trust attributable to other communicating agents.

In order to avoid underestimating the uncertainty linked to the state of perceived objects, we are studying data fusion filters capable of handling the information loops induced by exchanges. Our results on simulated data show that a split covariance intersection filter is a suitable method for this problem. Coupled with the parameter-tuning methodology we propose, this method also appears to outperform more conventional methods.

Next, we introduce a formalism for representing the areas covered by each sensor and the areas seen as free, in order to better merge the detected objects. This is the concept of evidential detectability grids, based on the theory of belief functions. These detectability grids make it possible to merge several points of view to obtain a global representation of the environment, while explicitly managing uncertainties.

Finally, we propose a method for each vehicle to elaborate a trust index on the other cooperative agents. It is based on an evidential tree combining several pieces of evidence, such as the consistency and concordance of the information received. The confidence index is then used to ensure that each vehicle reliably combines locally perceived information with that transmitted by other vehicles.

The performance of the global cooperative perception method is evaluated on real data obtained using three experimental vehicles equipped with omnidirectional

LiDAR sensors. The corresponding data sets are made available to the scientific community.

Résumé

Afin de naviguer de manière sûre et confortable, les véhicules intelligents nécessitent une perception très fiable de leur environnement d'évolution. Les capteurs embarqués étant nécessairement limités en portée et leur champ de vue pouvant faire l'objet d'occultations, une solution émergente est la perception coopérative : les véhicules partagent leur perception avec les autres véhicules par des moyens de communication sans-fil.

Les véhicules intelligents peuvent ainsi communiquer des informations complexes à travers de longues distances. Ils voient plus loin et de manière plus complète que ce que leurs capteurs leur permettent. Cependant, les informations provenant d'une source extérieure doivent être considérées avec prudence car une perception trompeuse peut entraîner une situation dangereuse. Il convient donc de limiter au maximum les sources de dégradation de l'intégrité de l'information dans le système coopératif. Dans cette thèse, nous étudions ces sources et nous proposons des méthodes adaptées pour les gérer et éviter leur propagation. Nos travaux se concentrent en particulier sur la fusion d'objets pistés, la représentation des zones couvertes par les systèmes de perception et la gestion de la confiance imputable aux autres agents communicants.

Afin d'éviter de sous-estimer l'incertitude liée à l'état des objets perçus, nous étudions des filtres de fusion de données capables de gérer les boucles d'information, induites par les échanges. Nos résultats sur données simulées montrent qu'un filtre à intersection de covariance partitionnée est une méthode adaptée à ce problème. Couplée à la méthodologie de réglage des paramètres que nous proposons, cette méthode peut également être plus performante que d'autres plus classiques.

Ensuite, nous présentons un formalisme permettant de représenter les zones couvertes par chaque capteur et les zones vues comme libres afin de mieux fusionner les objets détectés. C'est le concept de grilles de détectabilité évidentielles, basé sur la théorie des fonctions de croyance. Ces grilles de détectabilité permettent de fusionner plusieurs points de vue pour obtenir une représentation globale de l'environnement tout en gérant explicitement les incertitudes.

Finalement, nous proposons une méthode pour que chaque véhicule élabore un indice de confiance sur les autres agents coopératifs. Elle se base sur un arbre évidentiel combinant plusieurs éléments de preuve comme la cohérence et la concordance des informations reçues. L'indice de confiance est ensuite utilisé pour que chaque véhicule combine de façon fiable les informations perçues localement avec celles transmises par les autres véhicules.

Les performances de la méthode globale de perception coopérative sont évaluées sur des données réelles obtenues à l'aide de trois véhicules expérimentaux équipés de capteurs LiDAR omnidirectionnels. Les jeux de données correspondants sont rendus publics à la communauté scientifique.

Chapter 1

General Introduction

Contents

1.1	Intelligent Vehicles	11
1.2	Integrity	15
1.3	Objectives	18
1.4	Manuscript Organization	19

1.1 Intelligent Vehicles







Transportation and mobility have become major concern in modern societies. Humans are becoming increasingly more reliant on being able to move rapidly on a daily basis. At the same time, transportation is the cause of many accidents and casualties every year. As road vehicles occupy a large portion of both mobility and casualty aspects, intelligent road vehicles are seen as a good way to reduce the number of accidents. In addition, delegating navigation tasks to computers might simplify the use of private and public transportation, in particular to individuals with limited mobility. These elements motivated the development of on-board intelligence technologies, now allowing vehicles to be automated and potentially be autonomous in the future.

1.1.1 Levels of Automation

The autonomy of intelligent vehicles depends on their level of automation, which is classically defined by the Society of American Engineer (SAE) J3016 standard (SAE-J3016 2021) summarized in Table 1.1. The first group, Advanced Driver-Assistance Systems (ADASs), aggregates lower levels of automation that help human drivers in mundane driving tasks such as lane-keeping or Adaptive Cruise Control (ACC). ADAS are already available to consumers with for example *Tesla's Autopilot* that is authorized in the United States of America (USA). The second group is autonomous vehicles for levels in which road monitoring and navigation are progressively deferred to autonomous systems and drivers are less and less required. Several applications are being developed as of writing this

manuscript, such as *Zoox* or *Navia* autonomous shuttles that can drive on predetermined paths. In Europe, the *Mercedes Model S* is currently L3-certified (level 3) and offers autonomous driving on German highways during traffic jams up to 60 km/h.

Table 1.1: 6 Levels of Automation as defined by the SAE.

					
0	1	2	3	4	5
No	Driver	ADAS	Conditional	High	Full
Automation	Assistance		Automation	Automation	Automation

1.1.2 Autonomous Navigation Stack

The software architecture of autonomous vehicles is composed of several modules that are responsible for various navigation tasks, like localization or obstacle avoidance. Figures 1.1 and 1.2 give two examples of architectures used at Ulm University and Renault respectively. They provide an overview of the required modules and their interactions with each other. Starting on the left with raw sensors, modules progressively transform data to high level information and finally to low level control. In between is the *brain* of the vehicle that analyzes and understands its environment before deciding what to do and how.

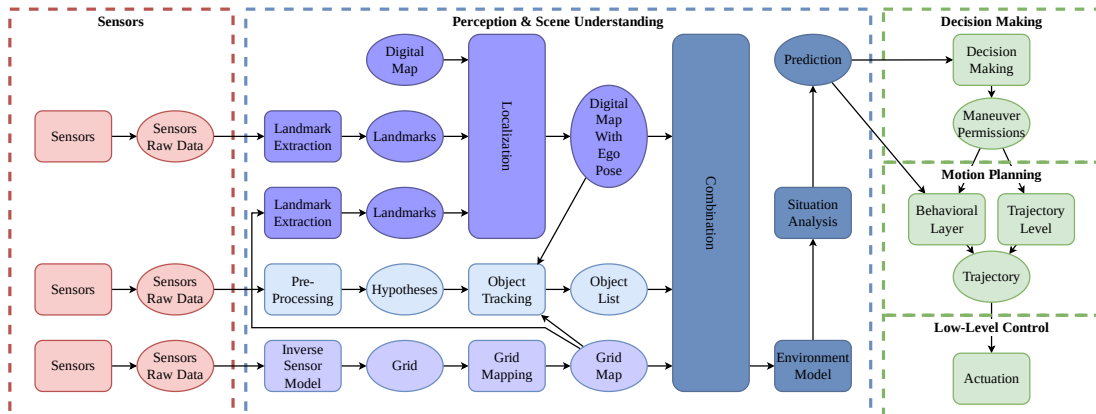


Figure 1.1: Architecture used at Ulm University, adapted from (Taş et al. 2016). Modules are regrouped in three main categories: input (sensors), processing (perception, scene understanding) and output (navigation, control). In the perception modules, note that objects and space are both estimated.

1.1.3 Environment Perception

The first part of the brain is the focus of this manuscript. Perception is a term encompassing everything that turns exteroceptive sensor data into information

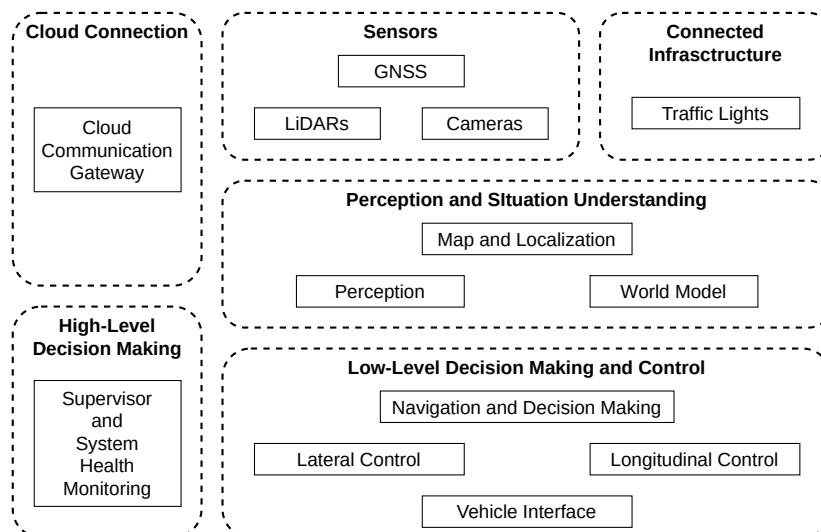


Figure 1.2: Architecture used at Renault during the Tornado project, adapted from (Milanés et al. 2022). The same main categories are present, but note the presence of communication with connected infrastructure and cloud connection.

that can be interpreted by situation understanding and motion planning modules. These modules decide where and how to move for the next few seconds in order to get closer to the goal while remaining safe. As such, the best information a perception system could give to them is the state of every point in space around the vehicle for the next few seconds. However, current technology and algorithms cannot provide such levels of information in both space and time. The next best solution is thus to split the problem in two and provide a *dual* information, as illustrated in Figure 1.3. Perception information is split in sparse but predictable objects and dense but instantaneous mappings. This separation stems from the realization that the reciprocal of object presence is not necessarily their absence but also that they have not been seen. That is why areas explicitly measured as free are mapped and constitute the second aspect of perception. How these two sides of the same coin are represented and processed will be discussed later.

1.1.4 Cooperative Perception

By using wireless communication, vehicles can exchange information and cooperate with each others. This way, vehicles can cooperatively perceive their environment, by exchanging objects and free space. This is called *Cooperative Perception* (CP), sometimes also called *collective perception* in the literature. Thanks to the diversity of points of view, their estimation of the driving environment can be improved and hidden zones can be reduced. Hidden areas or areas beyond the range of the sensors are indeed important because as the vehicle cannot obtain information at these locations its autonomous navigation capabilities, especially at high speed, are reduced. As communication can have large ranges and can pass through small obstructions, the perception of every vehicles can virtually be extended further and behind obstacles.

Cooperation can be of several kinds, summarized in Figure 1.4. *Centralized* ap-

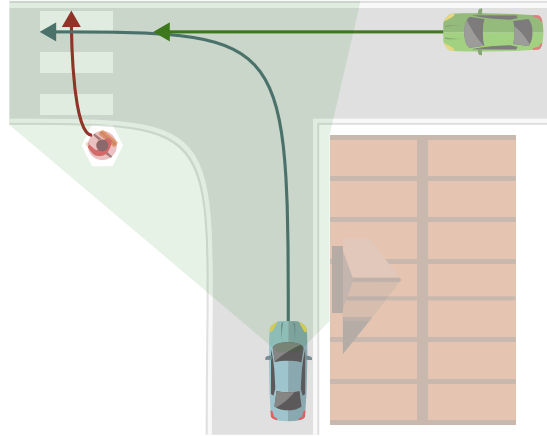


Figure 1.3: Scene composed of two vehicles and a pedestrian. Perceived objects and space considered free by the blue vehicle at the bottom are depicted in red and green respectively. This example also illustrates that there are always hidden areas in a field of perception.

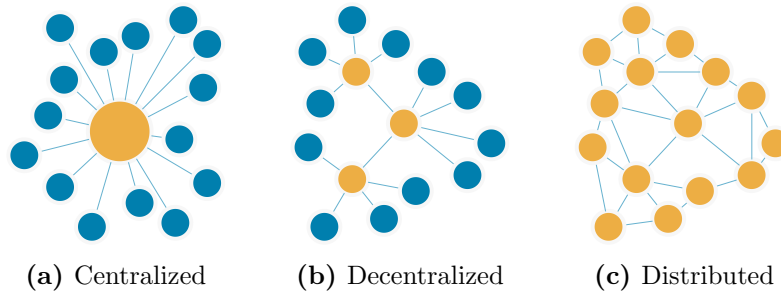


Figure 1.4: Common structures for communication and computation between multiple peers. Small blue nodes are sources of information. Orange nodes are where computations are realized. Small orange nodes are both information sources and computers.

approaches have a server at their center of operation. The central server gathers all available information, processes it then dispatches the result. This solution is the simplest to implement and can give optimal results but does not scale well. As the number of participants increases, exponentially more and more computations are required at a single point, which can become unfeasible and be less reliable as there is a single point of failure. In addition, centralized approaches require a constant connection to the internet, which is not yet possible everywhere nor necessarily desirable for privacy and environmental concerns. *Distributed* approaches propose to solve most of these issues by distributing the computation between peers. However, this comes at the expense of more complex protocols, which can be a significant issue for vehicular networks, that are already sufficiently complex due to their high dynamics.

To prevent these issues, *decentralized* approaches can be used instead. In this case, each participant is the center of its own partial world. Computation results can be shared with each other but no guarantee can be made about its redundancy between peers. Indeed, because each peer has a different point of view,

its knowledge of the environment might differ from peer to peer. This is not a problem though, as vehicles are supposed to be independently responsible for their own navigation tasks, and thus environment perception. A decentralized approach is thus more adapted to CP as it allows for situations where vehicles are not communicating or not cooperating.

Another common distinction depends on the type of task being realized. When all peers work together towards a common goal, the task is said *collective*. When all peers work independently but still share a common goal, the task is said *collaborative*. Finally, if peers work independently on their own goal but help each other, the task is said *cooperative*. Once again, because vehicles are supposed to be independently responsible for their own navigation, *cooperation* is better fitted to the problem at hand: vehicles can help each other but should be able to be independent.

CP can be seen as a way to improve the *quality* of perception or each peer. However, defining what quality means requires the introduction of another concept: *integrity*.

1.2 Integrity

1.2.1 General Definition

Integrity has an intuitive meaning in everyday life that is based on moral and ethical values such as honesty, incorruptibility or wholeness. According to (N. Zhu et al. 2018), the term integrity originates from the aeronautics field where it is used to manage the trust that can be placed in a navigation solution. The notion of integrity has since evolved towards more general concepts. For example, (Boritz 2005) defines information integrity as a subset of information quality, where the information is relevant, usable and reliable.

The idea has been progressively extended to the following set of characteristics summarized by (Balakrishnan 2020) in the context of intelligent vehicle localization:

- *Correctness*: Correspondence of the information with the physical reality up to a known degree of accuracy. For example, the information should be **accurate**, **consistent** in time and **unaltered** (whether intentionally or not);
- *Availability*: Information is available when its **extent** and its **quality** are sufficient to systems using it;
- *Completeness*: The available information faithfully represents the **whole** situation without missing a part of it;
- *Validity*: The available information faithfully represents the conditions, rules or relationships of the real world. For this, the information acquisition process thus has to be **understandable** and **verifiable**.

A good information in this sense is thus not the most accurate, but rather the most faithful to real world or in other words, that which will mislead other systems the least. Information is thus of high integrity as long as it is not misleading and sufficiently qualitative for the task to perform.

Additionally, integrity is transitive: if every sub-system is provided with non-misleading information and maintains that property throughout, then the whole system is assumed to be safe. For example, (Reid et al. 2019) details the interactions between modules in intelligent vehicles.

In the field of robotics and autonomous systems, integrity is particularly important as significant injuries and casualties can be caused by misunderstanding. The most critical modules in these are the localization and environment perception modules as a situation misunderstanding can result in hazardous behaviors. As such, let us present how integrity is defined for these modules.

1.2.2 Localization Integrity

The concept of integrity has historically been introduced to intelligent vehicles through the localization aspect (N. Zhu et al. 2018). Because localization refers to vehicles that always exist, it is by definition valid and complete, leaving only correctness to be monitored and availability to be evaluated. The worst situation possible for a localization system is to provide an erroneous but very confident pose to downstream systems. As such, its integrity mainly refers to correctness.

Localization integrity can be performed in real-time by computing protection levels and comparing them to Alert Limits (ALs) to declare the localization system available or not. In practice, the validation of a system that computes protection levels is done in post-processing with a reference ground truth by checking if its error is correctly bounded at all times.

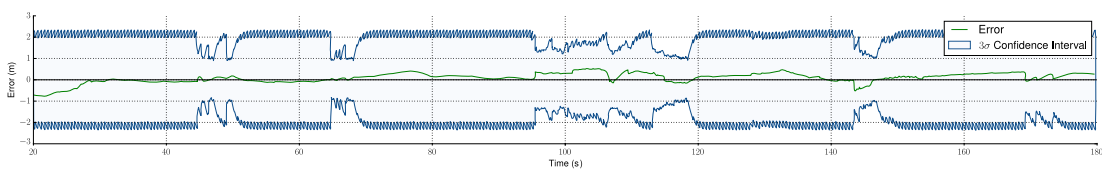


Figure 1.5: Lateral position error and bounds of a localization system through time. Taken from (Lima, Welte, et al. 2020).

Figure 1.5 illustrates such a problem, where a localization error and its bounds are plotted on the same graph. If the error is above the associated bound for more than a given percentage of the time, called Target Integrity Risk (TIR), the system does not respect the integrity requirement. The specification of the TIR is still a research topic, although some studies indicate that it should be between 10^{-4} and 10^{-7} per hour of critical operation (Reid et al. 2019).

Stanford diagrams are tools often used in aeronautics to evaluate integrity. They plot the real error against an estimated upper bound (i.e. protection levels), and define several zones to differentiate when the system is unavailable or not.

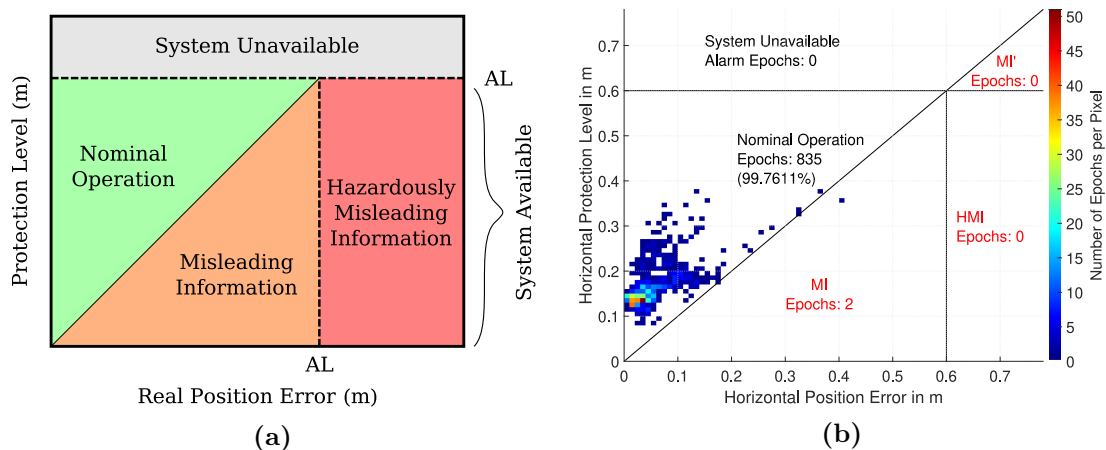


Figure 1.6: Simplified Stanford diagram and example of usage for evaluating a localization system taken from (Gottschalg et al. 2020). Dotted lines represent the Alert Limit.

As illustrated in Figure 1.6, when the system is available, three areas are differentiated: nominal, misleading or hazardously misleading. The protection level is linked to the TIR, which can in that case be interpreted as a bound for the risk of hazardously misleading information. The AL depends on the navigation context and mission. For example, on a wide motorway, localization does not have to be as accurate as in narrow city streets. Guaranteed scenarios and navigation conditions (i.e: traffic, meteorological) can be described in what is called an Operational Design Domain (ODD). The integrity analysis thus amounts to evaluating the system under all situations it is designed for and making sure that localization errors remains bounded.

1.2.3 Perception Integrity

As introduced in Section 1.1.3, the perception system is located before situation understanding and decision making. As such, perception has to characterize space that is free and detect road users in areas important for the ongoing navigation. Following a top-down approach to characterize perception integrity, problematic cases for perception systems are defined as:

1. Missing objects and considering space is free when it is not. This can lead the navigation system to unknowingly go towards other road users and colliding with them;
2. Estimating the position and velocities of objects with too much error. This can lead the decision system to incorrectly predict the trajectory of other road users which can lead to collisions and accidents;
3. Detecting objects that do not exist or considering space occupied when it is not. While less risky than the two previous points, it can lead the vehicle to either stop unnecessarily or issue false alarms to the driver.

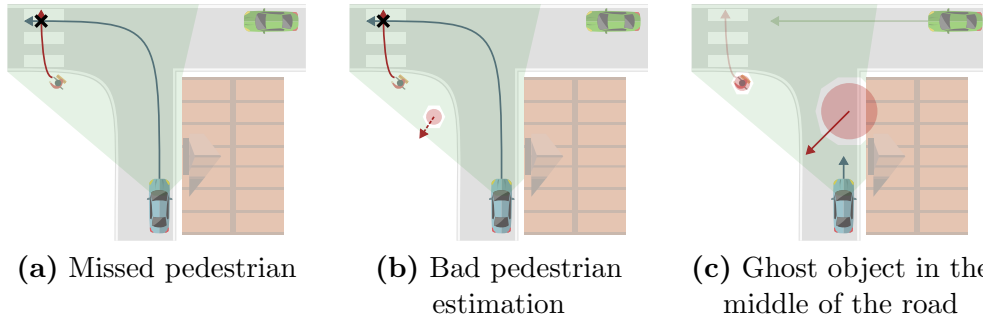
Example 1.1:

Figure 1.7: Misleading perception cases in the same situation as Figure 1.3. The road, real objects and their motion are displayed in the background. Localization error and estimated motion in red. The space perceived as free is in green.

Consider an intersection with two vehicles and a pedestrian as in Figure 1.7. From the point of view of the bottom blue vehicle, the green vehicle is hidden and the red pedestrian is about to cross. In Figure 1.7a, the green vehicle is not detected though space beneath it is supposedly free. In Figure 1.7b, the pose and motion of red pedestrian is badly estimated. In Figure 1.7c, a red object is detected though nothing is here. The danger of these situations resides in the fact that they will mislead later modules to think that respectively nothing is on the right, a pedestrian is not trying to cross and that an emergency braking is required.

Example 1.1 illustrates that a non-misleading perception system must be able to correctly assess where it can see whether space is free or not, while correctly estimating the position and velocity of perceived objects. Similarly to localization, a good perception system depends on the driving situation and task to be performed. For example, if the blue vehicle were to turn right, missing the green vehicle would be less dangerous than turning left, or at least would be less comfortable to the user.

Finally, although it is secondary to correctness, availability of the perception system also has to be considered. Hidden areas, such as the *building shadow* in Figure 1.7, are part of the nominal operation of perception but they reduce the availability of autonomous navigation. This can lead the navigation system to be overly cautious or even stop the vehicle. This is where perception sharing can be most useful.

1.3 Objectives

In this work, we will study the problem of cooperatively perceiving the environment in the context of intelligent vehicles evolving on open roads. We place ourselves in the scope of decentralized cooperative systems for previously mentioned reasons with the goal of managing the integrity of the information provided by

the perception modules. For this, we must ensure that all processing carried out in the perception module (i.e. detection and data fusion) maintains the integrity of perceptual information.

As such, the problem of fusing on-board and cooperative information will be central to this manuscript. It will lead us to study the detection of on-board exteroceptive information and the fusion of multiple points of view to extend the range of perception. In other words, the objective is to propose a cooperative perception system capable of fusing data coming from on-board sensors and from neighboring vehicles. In order to guarantee the data fusion quality, our approach will focus on estimating the uncertainties associated with perceived and received data as well as their coherency. Doing so, we will question the reliability of peers and derive evaluation methodologies adapted to cooperative systems.

Another objective of this work is to experimentally evaluate our proposals and implementations under realistic operating conditions on roads.

1.4 Manuscript Organization

This manuscript is split in four main chapters that respectively study four aspects of cooperative perception.

In Chapter 2, we review how symbolic and metric information can be represented and fused in cooperative systems. The concepts of belief functions and state filtering are introduced and a study is conducted on the resiliency of several filters to communication exchange loops when doing cooperation. We also propose an interpretation to the parameters Split Covariance Intersection Filter (SCIF) alongside a methodology to tune them.

In Chapter 3, we address how sensor data can be processed to detect objects and free space. Our experimental platform is described and a study is conducted on the detection capabilities of our experimental system based on evaluation metrics that will be used throughout this work.

In Chapter 4, we review cooperative perception methods and propose an architecture to fuse perception sources (either on-board sensor or other peers) when they only partially share fields of view. To this end, we introduce the novel concept of *evidential detectability*, a dense representation of the environment that combines free space and field of view. Several studies based on real datasets recorded for the occasion are conducted to illustrate the effectiveness of our method at improving object and free space detection.

In Chapter 5, we review the problem of malicious or faithfully erroneous peers and how they can be ignored. We then propose to manage the *trust*, a quantity estimated over time to represent how much information from a given external source can be trusted. Several experimental studies are conducted to evaluate trust estimation and its impact on cooperative object detection.

Finally, in Chapter 6, a general conclusion to this manuscript is drawn and future works are proposed.

Chapter 2

Methods and Tools for Decentralized Data Fusion

Contents

2.1	Introduction	21
2.2	Symbolic Information and Belief Functions	22
2.3	Metric Representation and State Filtering	29
2.4	Analysis of Covariance Intersection Filters	40
2.5	Conclusion	48

2.1 Introduction

This manuscript studies the exchange of perception information between vehicles. We chose to model this as a decentralized system, which raises several concerns. For example, information loops, illustrated in Figure 2.1, may lead a vehicle to count a same piece of information twice. To prevent this, robust methods must be employed, which is the focus of this chapter.

We will consider two types of information: metric (continuous quantities such as position or orientation) and symbolic (discrete characteristics such as the state of a traffic light or whether an object exists or not). Because information is never perfect in the real world, we will introduce how uncertain information can be represented and fused in a decentralized system.

The framework of belief functions is first introduced to represent and manage uncertain symbolic information. It will be used throughout this manuscript to represent the existence of objects, the similarity between objects and the detectability of objects in Chapter 4, then trustworthiness in cooperative peers in Chapter 5.

Then, the concept of random state vector filtering is introduced to represent uncertain metric information. A comparison of the performance of several filters is then conducted to determine the most suitable filter for decentralized data

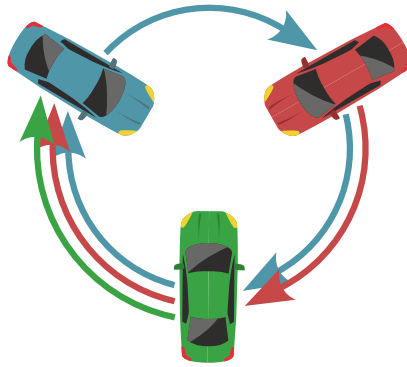


Figure 2.1: Information loop introduced between three vehicles. Follow the blue arrow from the top that represents the initial piece of information sent by the blue vehicle. It is augmented by the information of the red and green vehicle then comes back to the blue vehicle. This means that the blue vehicle will receive its own information, potentially thinking that the information is new.

fusion. These concepts will be used throughout this manuscript to estimate the position and velocity of objects.

Finally, note that information will be assumed to be temporally and spatially aligned in this chapter. Following chapters address this point.

2.2 Symbolic Information and Belief Functions

The traditional way of modeling uncertain symbolic information is with probabilities. However with probabilities, having less information about hypothesis H^1 than about hypothesis H^2 means that H^2 is more likely. When this is not the case (i.e. because H^1 and H^2 are not linked in that way), then belief functions can instead be used. This section introduces the basic concepts of this theory.

2.2.1 Representation

In their simplest form, belief functions are an extension of Bayesian probabilities to set theory. That is, a classic Bayesian probability between 0 and 1 can be distributed among several facets of a problem at once. It is particularly useful to represent the information expressed heterogeneous sources. Let us take an example to illustrate this property.

Example 2.1:

Consider an algorithm that is able to find cats, ducks or platypuses in images based on several characteristics. By detecting a beak, it might be 90% sure that either a duck or platypus is in the image, but cannot differentiate between the two. By detecting fur, it might be 90% sure that there is either

a cat or platypus. With classical probabilities, one would tend to model the ambiguity by splitting probabilities in two, resulting in:

$$\begin{cases} p(\text{cat}) &= p^{\text{beak}}(\text{cat}) \cdot p^{\text{fur}}(\text{cat}) = 0 \cdot 0.45 = 0 \\ p(\text{duck}) &= p^{\text{beak}}(\text{duck}) \cdot p^{\text{fur}}(\text{duck}) = 0.45 \cdot 0 = 0 \\ p(\text{plat}) &= p^{\text{beak}}(\text{plat}) \cdot p^{\text{fur}}(\text{plat}) = 0.45 \cdot 0.45 = 0.2025 \end{cases}$$

which yields sub-optimal results caused by the algorithm ambiguities. In addition, information that there still might be a cat or duck in the image is lost, ignoring the uncertainties of the algorithm. This is due to the inability of Bayesian probabilities to model that $1 - p(x)$ is not always $p(x)$.

This sort of limitations are lifted by belief functions. Introduced by (Dempster 1967) and refined in (Shafer 1976) then (Philippe Smets and Kennes 1994), they form a theory that bears many names, including Dempster-Shafer Theory (DST), theory of evidence, evidential framework or Transferable Belief Model (TBM).

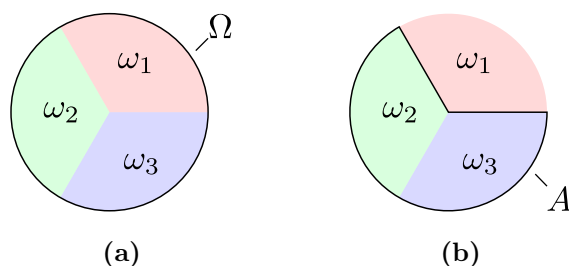


Figure 2.2: Sets composing $\Omega = \{\omega_1, \omega_2, \omega_3\}$ and subset $A = \{\omega_2, \omega_3\}$

The building blocks of this framework are mass functions, mappings $m : 2^\Omega \rightarrow [0, 1]$ where $\Omega = \{\omega_1, \omega_2, \dots\}$ is the finite set of answers to the question asked to m , called frame of discernment. There exists multiple interpretation as to what Ω should contain and what the values of m mean. However, in this manuscript we only consider the evidential interpretation where Ω is composed of mutually exclusive basic hypotheses and m describes the evidence held about any permutation of these hypotheses (subsets of Ω). Following the closed-world assumption, hypotheses are considered exhaustive.

These permutations are defined as the powerset $2^\Omega = \{\emptyset, \{\omega_1\}, \{\omega_2\}, \{\omega_1, \omega_2\}, \dots, \Omega\}$. Accessing an element of a mass function is noted $m(A)$ where A is an element of 2^Ω . $m(A)$ is the proportion of evidence attributed to A specifically and no other subset, meaning the information contributing to tell that the real answer $w \in A$ and $\omega \notin A$. To keep their probabilistic nature, mass functions should always observe the constraint that $1 = \sum_{A \in 2^\Omega} m(A)$.

Example 2.2:

Getting back to the Example 2.1, the different hypotheses our detection

algorithm can answer are cat (C), duck (D) or platypus (P), thus $\Omega = \{C, D, P\}$. Beak detection can be expressed using a mass function:

$$\begin{cases} m(\{D, P\}) & = 0.9 \\ m(\{C, D, P\}) & = 0.1 \end{cases}$$

where only non-zeros subsets are given. For the sake of clarity, another notation will be used in this work with all subsets given:

$$m^{\text{beak}} = \left[\begin{array}{cccccccc} \emptyset & \{C\} & \{D\} & \{C, D\} & \{P\} & \{C, P\} & \{D, P\} & \{C, D, P\} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0.1 \end{array} \right]$$

This means that 90% of the available evidence tends to indicate that either a duck or platypus is in the image. This is different than saying $m(\{D\}) = m(\{P\}) = 0.9$ as it would violate the 1-summed constraint or that $m(\{D\}) = m(\{P\}) = 0.45$ which would not solve the problem of the previous example.

Similarly, fur detection can be expressed with

$$m^{\text{fur}} = \left[\begin{array}{cccccccc} \emptyset & \{C\} & \{D\} & \{C, D\} & \{P\} & \{C, P\} & \{D, P\} & \{C, D, P\} \\ 0 & 0 & 0 & 0 & 0 & 0.9 & 0 & 0.1 \end{array} \right]$$

There are specific interpretations and properties attached to mass functions. To name a few

- $m(\Omega)$ is the degree of ignorance, also called the unknown set;
- $m(\emptyset)$ is the degree of conflict¹;
- Subsets that have a positive mass are called focal sets;
- A mass function without mass on Ω is called dogmatic;
- A mass function whose only focal set is Ω is called vacuous;
- A mass function whose focal sets are all singletons is called Bayesian;
- A mass function with mass only on Ω and one other focal set is called simple.

Mass is generally just the *raw* way of representing belief. It can be used in Basic Belief Assignment (BBA) to turn other forms of information into belief, for example to introduce belief in a sensor model. Once in the domain of belief, other higher-level descriptions can be used, as represented in Figure 2.3:

¹Conflict can arise when fusing two mass functions that disagree with each other. This can mean that one source is mistaken, overly confident or that the closed-world assumption is not respected, in which case the actual answer might be outside of what Ω covers.

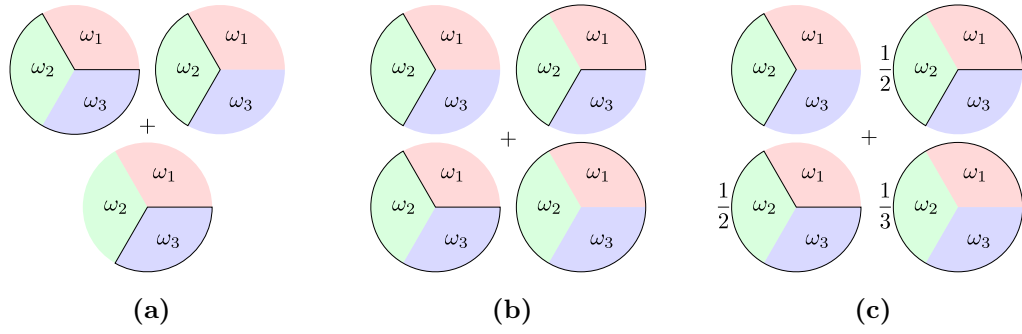


Figure 2.3: Subsets used when computing $Bel(\{\omega_2, \omega_3\})$ in (a), $Pl(\{\omega_2\})$ in (b) and $BetP(\{\omega_2\})$ in (c).

- Belief, or credibility. It is the sum of all evidence supporting A , often interpreted as the lower bound of the actual probability of A :

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad (2.1)$$

- Plausibility. It is the sum of all evidence not contradicting A , often interpreted as the upper bound of the actual probability of A :

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \quad (2.2)$$

When applied on a singleton ω , it is called the contour function:

$$pl(\omega) = Pl(\omega), \quad |\omega| = 1 \quad (2.3)$$

- Commonality. It is the quantity of knowledge on A :

$$q(A) = \sum_{B \supseteq A} m(B) \quad (2.4)$$

- Weight:

$$w(A) = \sum_{B \supseteq A} (-1)^{|B|-|A|} \ln q(B) \quad (2.5)$$

- Pignistic. It transforms evidence into probabilistic distributions, losing ambiguity and incompleteness information, but useful to make decisions:

$$BetP(A) = \sum_{B \subseteq \Omega} \frac{|A \cap B|}{|B|} \cdot \frac{m(B)}{1 - m(\emptyset)} \quad (2.6)$$

As mentioned earlier, belief functions are particularly useful to combine various pieces of evidence. In the next section, we review several methods to combine belief functions.

2.2.2 Combination of Mass Functions

The idea of combination is simply to take two mass functions m_1 and m_2 and generate a third one that summarizes the information of the two others. However, depending on how the inputs are modeled and whether they are reliable and independent, several combination rules can be used. The appropriate rule depends on the situation at hand. Thus in this section, the most common rules are reviewed based on the work of (Reineking 2014). Note that unless explicitly denoted, combination rules require m_1 and m_2 to be defined on the same frame of discernment.

2.2.2.1 Conjunctive Combination

The first combination rule, called conjunctive and introduced by (Dempster 1967) combines intersecting sets in the Bayesian sense:

$$(m_1 \odot m_2)(A) = \sum_{B \cap C = A} m_1(B) \cdot m_2(C) \quad (2.7)$$

It is commutative, associative and accepts the vacuous mass function as a neutral element. However, this combination rule can lead to the generation of conflict if there is non-intersecting masses. In many applications, conflict is best avoided, which is why a *normalized* version of Equation (2.7) have been proposed, called Dempster's rule:

$$(m_1 \oplus m_2)(A) = \begin{cases} \frac{1}{1-\eta} (m_1 \odot m_2)(A) & A \neq \emptyset, \eta = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C) \\ 0 & A = \emptyset \end{cases} \quad (2.8)$$

It is particularly useful when combining two reliable and independent sources, i.e: both sources faithfully estimate their amount of evidence and are not statically correlated with one another. This condition is important, as counter-intuitive results can arise from combining two contradictory mass functions.

Example 2.3:

Take two mass functions defined on $\Omega = \{A, B, C\}$. One believes strongly in A and slightly in B while the second strongly believe in C and slightly in B . The following tables summarizes their contents and Dempster's combination result:

		\emptyset	$\{A\}$	$\{B\}$	$\{A, B\}$	$\{C\}$	$\{A, C\}$	$\{B, C\}$	$\{A, B, C\}$
m_1		0	0.9	0.1	0	0	0	0	0
m_2		0	0	0.1	0	0.9	0	0	0
$m_1 \oplus m_2$		0	0	1	0	0	0	0	0

that is, combining two masses weakly confident in B yields a categorical mass on B and nothing else. This is due to the inability of m_1 and m_2 to

correctly assess their evidence. They both overestimated their evidence in contradictory information without expressing their uncertainties.

However, by modeling m_1 and m_2 to express evidence and not belief, this should not happen as evidence may be uncertain but never contradictory. Another way is to think of evidence is as belief constraints.

To get back to Example 2.2, beak and fur detection are modeled to express evidence in a reliable and independent manner. As such they can be combined using Dempster's rule:

$$\begin{array}{l}
 m^{\text{beak}} \\
 m^{\text{fur}} \\
 m^{\text{beak}} \oplus m^{\text{fur}}
 \end{array}
 \left[\begin{array}{c}
 \hline
 \emptyset \quad \{C\} \quad \{D\} \quad \{C, D\} \quad \{P\} \quad \{C, P\} \quad \{D, P\} \quad \{C, D, P\} \\
 \hline
 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.9 \quad 0.1 \\
 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.9 \quad 0 \quad 0.1 \\
 0 \quad 0 \quad 0 \quad 0 \quad 0.81 \quad 0.09 \quad 0.09 \quad 0.01 \\
 \hline
 \end{array} \right]$$

which properly models that detecting a beak and fur is strong evidence that a platypus is in the image, while maintaining that it might be something else due to detection errors.

There are also other methods to normalize conflict. While Dempster's rule proposes to distribute it across focal sets, Yager's rule (Yager 1987) proposes to transfer it on $m(\Omega)$:

$$(m_1 \textcircled{Y} m_2)(A) = \begin{cases} (m_1 \textcircled{\cap} m_2)(A) & A \in 2^\Omega \setminus \emptyset, \Omega \\ (m_1 \textcircled{\cap} m_2)(\Omega) + (m_1 \textcircled{\cap} m_2)(\emptyset) & A = \Omega \\ 0 & A = \emptyset \end{cases} \quad (2.9)$$

and Dubois & Prade's rule (Dubois et al. 2008) proposes to assign it to the union of corresponding focal sets:

$$(m_1 \textcircled{D} m_2)(A) = \begin{cases} (m_1 \textcircled{\cap} m_2)(A) + \sum_{\substack{B \cap C = \emptyset \\ B \cup C = A}} m_1(B) \cdot m_2(C) & A \in 2^\Omega \setminus \emptyset \\ 0 & A = \emptyset \end{cases} \quad (2.10)$$

2.2.2.2 Disjunctive Combination

However, the last two rules are not associative and their use in data fusion is limited. In these situations, and in particular when at least one source is reliable, the disjunctive rule (Philippe Smets 1993) can be used:

$$(m_1 \textcircled{\cup} m_2)(A) = \sum_{B \cup C = A} m_1(B) \cdot m_2(C) \quad (2.11)$$

that is, multiply masses everywhere they are defined. This rule is interesting to acknowledge but it will not be used in this manuscript as it is too cautious.

2.2.2.3 Cautious Combination

One of the major constraint with previous rules is that they assume sources to be independent. In (Dencœux 2008), a cautious conjunctive rule is proposed, that is resilient to information redundancy and is even idempotent ($\text{comb}(m, m) = m$). Its principle is to take the source with the least amount of information in the case that this minimal information is shared between sources. For this, both mass functions are transformed in the weight space using Equation (2.5) to combine intermediary simple mass functions:

$$w_{1\wedge 2} = \min(w_1(A), w_2(A))$$

$$m_1 \oslash m_2 = \bigoplus_{AC\Omega} \left[\begin{array}{cc} \{A\} & \Omega \\ 1 - w_{1\wedge 2} & w_{1\wedge 2} \end{array} \right] \quad (2.12)$$

2.2.2.4 Partially Overlapping Fusion

In the next chapters, detectability information will be modeled as subjective to a given point of view, represented by a dependency on the point of view in the frame of discernment. For this reason, we introduce here the method of (P. Smets 2000) to combine partially overlapping functions. As long as $|\Omega_1 \cap \Omega_2| > 0$, this method can be used to combine m_1 and m_2 on $\Omega_1 \cup \Omega_2$. It is based on conjunctive combination, normalization weights and conditioning. As a reminder, conditioning is defined as Dempster combination with the neutral element of another frame of discernment.

Example 2.4:

Let m^1 defined on Ω^1 . Conditioning m^1 on $\Omega^2 = \{\omega_1^2, \omega_2^2, \dots\}$ is realized as

$$m^1[\Omega^2] = m^1 \oplus \left[\begin{array}{cccccc} \emptyset & \{\omega_1^2\} & \{\omega_2^2\} & \{\omega_1^2, \omega_2^2\} & \dots & \{\omega_1^2, \omega_2^2, \dots\} \\ 0 & 0 & 0 & 0 & \dots & 1 \end{array} \right] \quad (2.13)$$

Partially overlapping combination is defined for all A in $2^{\Omega_1 \cup \Omega_2}$ as

$$(m_1 \oplus m_2)(A) = \frac{m_1(A_1)}{m_1[\Omega_0](A_0)} \frac{m_2(A_2)}{m_2[\Omega_0](A_0)} (m_1[\Omega_0] \oplus m_2[\Omega_0])(A_1 \cap A_2) \quad (2.14)$$

with $\Omega_0 = \Omega_1 \cap \Omega_2$, $A_0 = A \cup \Omega_0$, $A_1 = A \cup \Omega_1$, $A_2 = A \cup \Omega_2$.

2.2.3 Discounting

When a source is unreliable, its BBAs can be *discounted*. By moving a α proportion of its focal sets to the unknown, the informativeness of a mass function m is reduced. This operation is defined as

$${}_{\alpha}m = \begin{cases} (1 - \alpha) \cdot m(A) & A \in 2^{\Omega} \setminus \Omega \\ (1 - \alpha) \cdot m(A) + \alpha & A = \Omega \end{cases} \quad (2.15)$$

with $(1 - \alpha)$ representing the reliability of that source. Other forms of discounting have been proposed such as the contextual discounting introduced in (Mercier et al. 2005) where reliability is specified for each element as a vector $\alpha = \{\alpha_A\}_{A \in \Omega}$

$$\alpha m = m \circledast \left(\bigoplus_{A \in \Omega} \left[\begin{array}{cccc} \emptyset & A & \dots & \Omega \\ 1 - \alpha_A & \alpha_A & 0 & 0 \end{array} \right] \right) \quad (2.16)$$

or temporal discounting (Kurdej et al. 2013), that applies a discounting whose α depends on some elapsed time. The underlying idea is to imitate particle physics where particle *decay* exponentially, which is modeled with a half-life time parameter $t_{1/2}$. The shorter the half-life, the quicker a belief function is thus discounted:

$$\Lambda(\Delta t, t_{1/2}) := e^{-\frac{\ln 2}{t_{1/2}} \Delta t} \quad (2.17)$$

which has the advantage of being additive and associative

$$\Lambda(\Delta t_1)(\Lambda(\Delta t_2)m) = \Lambda(\Delta t_2)(\Lambda(\Delta t_1)m) = \Lambda(\Delta t_1 + \Delta t_2)m \quad (2.18)$$

Discounting will be used in the rest of this manuscript to reduce the impact of certain sources when cooperatively estimating quantities. In particular temporal discounting will be used to *filter* belief, removing information as time passes.

2.2.4 Conclusion

As we saw in this section, belief functions are a mathematical tool that extends Bayesian probabilities to combination of events. In doing so, it allows for explicit representation of uncertainties, which is particularly useful in decentralized data fusion where information can be partial between peers. In addition, it provides fusion tools that are resilient to potentially correlated sources.

2.3 Metric Representation and State Filtering

We now focus on metric information, its representation with random state vectors and its filtering.

2.3.1 Random State Vectors

Continuous quantities are usually represented using random state vectors, where the *state* of an object is represented with a multidimensional Gaussian distribution². This takes the form of a mean vector \mathbf{x} and covariance matrix \mathbf{P} whose meaning is illustrated in the following example:

²Other probability distributions are possible, but they will not be covered in this manuscript.

Example 2.5:

Consider a one-dimensional object represented by its position s . That position is uncertain, which is encapsulated within the standard deviation of s , σ_s . Its state and covariance are thus respectively

$$\mathbf{x} = \begin{bmatrix} s \end{bmatrix}, \mathbf{P} = \begin{bmatrix} \sigma_s^2 \end{bmatrix}$$

This conveys that the most likely position is s but that the real position can be somewhere else with a probability decreasing with distance to s . This is illustrated in Figure 2.4 with a Gaussian probability distribution and as uncertainty bounding with a 3σ domain.

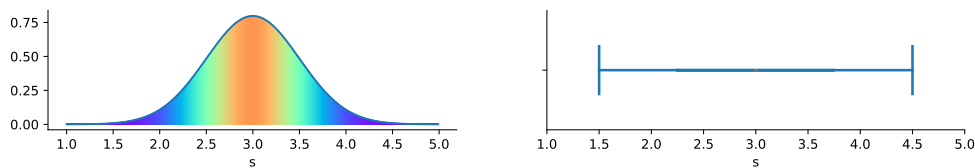


Figure 2.4: Gaussian probability distribution and 3σ domain of a one-dimensional object with a mean $s = 3$ and standard deviation $\sigma_s = 0.5$.

Such a concept can be extended to vectors of higher dimensions. This will be used to represent perceived objects in this manuscript.

Example 2.6:

Consider a 2-dimensional object. It is represented by its x and y position and associated uncertainties as

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{P} = \begin{bmatrix} \sigma_x\sigma_x & \eta\sigma_x\sigma_y \\ \eta\sigma_y\sigma_x & \sigma_y\sigma_y \end{bmatrix}$$

Here η is the correlation factor between σ_x and σ_y . Such a state is illustrated in Figure 2.5 with a 2D Gaussian probability distribution and a 3σ bounding ellipsoid.

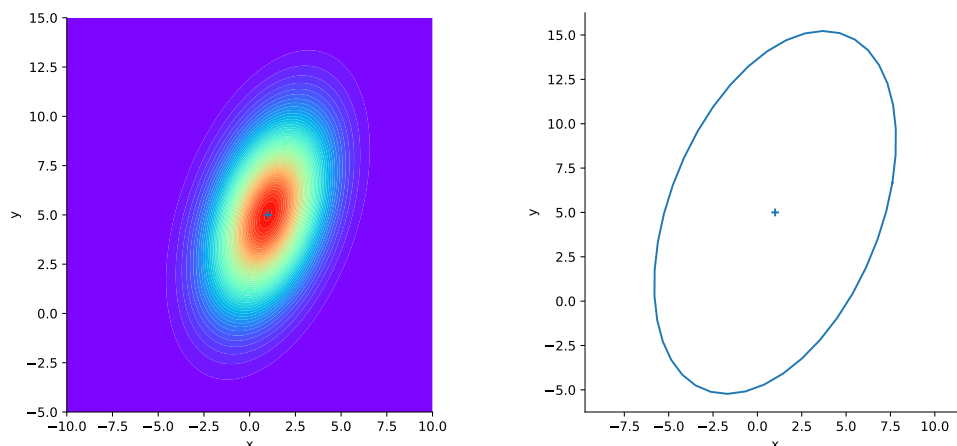


Figure 2.5: 2D Gaussian probability distribution and 3σ bound of a two dimensional object with $x = 1$, $y = 5$, $\sigma_x = 2$, $\sigma_s = 3$ and $\eta = 0.4$.

Other continuous characteristics can also be represented using this formalism, such as the speed or the heading of a vehicle. In these cases, the illustration is different (an arrow for the speed and a cone for the heading) but the underlying principle is the same: the most likely value is the mean, but other values are possible with decreasing probability.

The state of multiple objects can be represented jointly or independently, meaning that cross-covariances between objects can be represented or ignored. In the former, the state of the system is represented as a whole, whereas the latter only represents the states of individual objects separately.

Example 2.7:

Consider a system with two objects o_1 and o_2 . Their states and covariances can be represented jointly as Equation (2.19) or independently as Equation (2.20).

$$\left(\begin{array}{c} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}, \begin{bmatrix} \sigma_{x1}\sigma_{x1} & \eta_{x1y1}\sigma_{x1}\sigma_{y1} & \eta_{x1x2}\sigma_{x1}\sigma_{x2} & \eta_{x1y2}\sigma_{x1}\sigma_{y2} \\ \eta_{x1y1}\sigma_{y1}\sigma_{x1} & \sigma_{y1}\sigma_{y1} & \eta_{x2y1}\sigma_{y1}\sigma_{x2} & \eta_{y1y2}\sigma_{y1}\sigma_{y2} \\ \eta_{x1x2}\sigma_{x2}\sigma_{x1} & \eta_{x2y1}\sigma_{x2}\sigma_{y1} & \sigma_{x2}\sigma_{x2} & \eta_{x2y2}\sigma_{x2}\sigma_{y2} \\ \eta_{x1y2}\sigma_{y2}\sigma_{x1} & \eta_{y1y2}\sigma_{y2}\sigma_{y1} & \eta_{x2y2}\sigma_{y2}\sigma_{x2} & \sigma_{y2}\sigma_{y2} \end{bmatrix} \end{array} \right) \quad (2.19)$$

$$\left\{ \left(\begin{array}{c} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \\ \begin{bmatrix} \sigma_{x1}\sigma_{x1} & \eta_{x1y1}\sigma_{x1}\sigma_{y1} \\ \eta_{x1y1}\sigma_{y1}\sigma_{x1} & \sigma_{y1}\sigma_{y1} \end{bmatrix} \end{array} \right), \left(\begin{array}{c} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \\ \begin{bmatrix} \sigma_{x2}\sigma_{x2} & \eta_{x2y2}\sigma_{x2}\sigma_{y2} \\ \eta_{x2y2}\sigma_{y2}\sigma_{x2} & \sigma_{y2}\sigma_{y2} \end{bmatrix} \end{array} \right) \right\} \quad (2.20)$$

The main difference between both representations is that the former models the statistical interdependence between object estimates, while the latter considers

that they are uncorrelated. However, as illustrated in the previous example, this also means that in the joint case, the number of elements in the covariance matrix grows exponentially with the number it represents, as compared the uncorrelated case where growth is linear.

2.3.2 State Filtering

While random state vectors are useful to represent instantaneous information, their main advantage is the capacity to be filtered, that is estimated through time by periodically observing the system.

Filtering is based on the idea that a state $\mathbf{x}(t)$ can be used to represent a system at a given moment. This state evolves in a predictable manner with an evolution model described by a function \mathbf{f} . Information about the system is acquired periodically in the form of an observation vector \mathbf{y} thanks to an observation model described by a function \mathbf{h} . However, due to modeling or sensor errors, knowledge about the system is never perfect, which is modeled by $\alpha(t)$ for evolution model-induced errors and $\beta(t)$ for sensor-induced errors. They are both assumed to be centered. The following state equations summarizes these ideas:

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{f}(\mathbf{x}(t)) + \alpha(t) \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t)) + \beta(t)\end{aligned}\tag{2.21}$$

Note that this is a discretized model in the sense that it is sampled in time and not a continuous differential equation. $x(t+1)$ corresponds to the future sampling time for a given period.

2.3.2.1 Kalman Filtering

The Kalman Filter (KF), introduced in (Kalman 1960) is the most common filter in robotics and navigation. It recursively estimate states with two steps:

1. It predicts the state and its covariance using a linear evolution model \mathbf{F} and a covariance \mathbf{Q} of the model noise α :

$$\begin{aligned}\mathbf{x}(t+1|t) &= \mathbf{F}\mathbf{x}(t|t) \\ \mathbf{P}(t+1|t) &= \mathbf{F}\mathbf{P}(t|t)\mathbf{F}^T + \mathbf{Q}\end{aligned}\tag{2.22}$$

\mathbf{F} describes how \mathbf{x} and its covariance \mathbf{P} evolve between two time steps. \mathbf{Q} models that the evolution model is imperfect (e.g: air and ground friction not taken into account, acceleration not modeled, discretization of time, ...). It increases the state covariance to convey that the state becomes more imprecise with time because of modeling incompleteness or external disturbances. For the sake of clarity, notations used in the rest of this manuscript will be shortened. The current state $\langle \mathbf{x}(t|t), \mathbf{P}(t|t) \rangle$ will be noted $\langle \mathbf{x}, \mathbf{P} \rangle$ and the predicted state $\langle \mathbf{x}(t+1|t), \mathbf{P}(t+1|t) \rangle$ will be noted $\langle \mathbf{x}^+, \mathbf{P}^+ \rangle$, such that Equation (2.22) becomes:

$$\begin{aligned}\mathbf{x}^+ &= \mathbf{F}\mathbf{x} \\ \mathbf{P}^+ &= \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}\end{aligned}\tag{2.23}$$

- It updates the state and its covariance using an observation. It is a M -dimensional vector \mathbf{y} associated with an observation noise covariance \mathbf{R} that describes the variance of β . Observations are linked to the state with an observation model \mathbf{H} , a matrix of size $M \times N$, that describes how observed quantities are linked to the state.

$$\begin{aligned}
 \mathbf{K} &= \mathbf{P}^+ \mathbf{H}^T (\mathbf{H} \mathbf{P}^+ \mathbf{H}^T + \mathbf{R})^{-1} \\
 \epsilon &= \mathbf{y} - \mathbf{H} \mathbf{x}^+ \\
 \mathbf{x} &= \mathbf{x}^+ + \mathbf{K} \epsilon \\
 \mathbf{P} &= (\mathbf{I} - \mathbf{K} \mathbf{H}) \mathbf{P}^+ (\mathbf{I} - \mathbf{K} \mathbf{H})^T + \mathbf{K} \mathbf{R} \mathbf{K}^T
 \end{aligned} \tag{2.24}$$

The uncertainties of \mathbf{P} and \mathbf{R} are combined to compute a Kalman gain \mathbf{K} that represents the influence of \mathbf{y} and \mathbf{R} on the resulting state. A possible interpretation is that \mathbf{K} is a weighing factor along all dimensions of the filtered state and observation that takes the observation model and cross-covariances between dimensions into account. An innovation ϵ is computed to correct the state using through \mathbf{K} . Note that Equation (2.24) is given in the more numerically stable Joseph's form.

Example 2.8:

Consider a dynamic object that has a x and y position moving with constant velocities v_x and v_y .

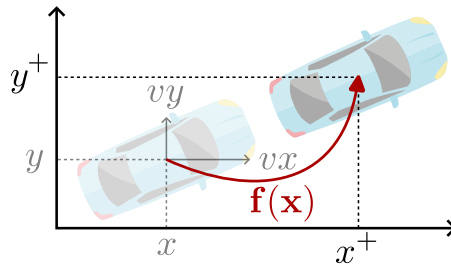


Figure 2.6: 2D Object with its current and predicted state. Covariance is not represented.

Its state is thus $\mathbf{x} = [x, y, v_x, v_y]$ and its evolution can be described with:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{v}_x = 0 \\ \dot{v}_y = 0 \end{cases}$$

$$\mathbf{f}_l(\mathbf{x}) = \begin{cases} x^+ = x + \Delta t \cdot vx \\ y^+ = y + \Delta t \cdot vy \\ vx^+ = vx \\ vy^+ = vy \end{cases} \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where \mathbf{f} is the continuous evolution model, \mathbf{f}_l is \mathbf{f} discretized with time step Δt and \mathbf{F} is \mathbf{f}_l in matricial form. Uncertainties about velocity are propagated to position as

$$\begin{aligned} \sigma_x^{+2} &= \sigma_x^2 + \Delta t^2 \cdot \sigma_{vx}^2 + q_x^2 \\ \sigma_y^{+2} &= \sigma_y^2 + \Delta t^2 \cdot \sigma_{vy}^2 + q_y^2 \end{aligned}$$

with q_x, q_y the variance of the model noise covariance \mathbf{Q} for x and y .

Thanks to a GNSS receiver and to an odometer, the system can observe the whole state:

$$\mathbf{h}(\mathbf{x}) = \begin{cases} \mathbf{y}_x = x \\ \mathbf{y}_y = y \\ \mathbf{y}_{vx} = vx \\ \mathbf{y}_{vy} = vy \end{cases} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

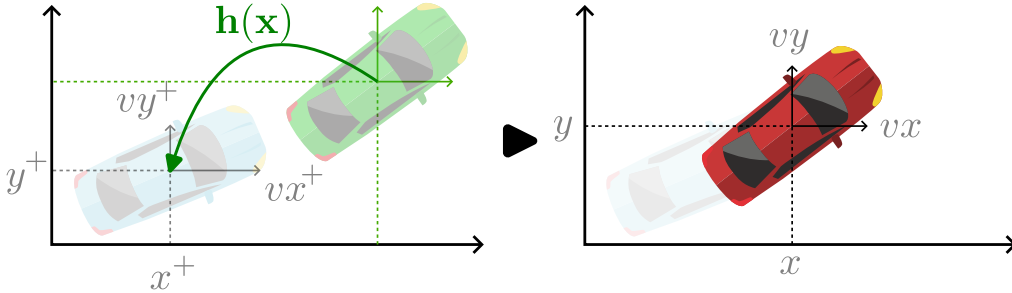


Figure 2.7: Update of a 2D object. Blue is the predicted state, green the observation and red the updated state. Covariance is not represented. Observation error exaggerated for clarity.

It is also possible to only measure a subset of the state and yet estimate it fully. This works as long as all quantities are even indirectly *observable*, with a mechanism that is best explained with an example.

Example 2.9:

Consider the previous example, but now only a GNSS receiver is available:

$$\mathbf{h}(\mathbf{x}) = \begin{cases} \mathbf{y}_x = x \\ \mathbf{y}_y = y \end{cases} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

When the state is predicted, non-diagonal terms of \mathbf{F} propagate some uncertainty from the velocity to the position, and in particular on non-diagonal terms of \mathbf{P} as a result of the correlation between the two quantities. When the state is updated, the Kalman gain \mathbf{K} of Equation (2.24) uses these correlated terms to infer how to correct the velocity from a position error.

The KF is said optimal under certain conditions:

- States and observations are affected by noises and errors that follow Gaussian distributions whose variance is known. Moreover, these distributions must be zero-centred and white (i.e: uncorrelated in time);
- States evolve and are observed through linear equations.

2.3.2.2 Extensions to the Kalman Filter

Linear modelling is rarely met in practice. Evolution and observation models are often non-linear (e.g: heading angle having a sinusoidal impact on the position). For this, there are extensions of the KF that:

- Linearize locally the evolution and observation model using Jacobian matrices, the Extended Kalman Filter (EKF) (Kalman 1960). In this case, Equation (2.23) becomes

$$\begin{aligned}\mathbf{x}^+ &= \mathbf{f}(\mathbf{x}) \\ \mathbf{F} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \\ \mathbf{P}^+ &= \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}^T\end{aligned}\tag{2.25}$$

and Equation (2.24) gets modified with

$$\begin{aligned}\mathbf{H} &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \\ \epsilon &= \mathbf{y} - \mathbf{h}(\mathbf{x}^+)\end{aligned}\tag{2.26}$$

- Apply the non-linear function to some representative points of the covariance ellipsoid to propagate the uncertainty: this is the Unscented Kalman Filter (UKF) (S. J. Julier et al. 1997b).

When noises affecting the system are biased or correlated, the KF might over-converge on a given solution and might provide inconsistent estimates. For raw sensor data, this assumption holds as long as a thorough work on their computation ensures that observations are unbiased and uncorrelated, as explored in Section 3.5. However, in decentralized data fusion, observation are already filtered states, meaning that this assumption does not hold. This inspired a plethora of other types of filter that are detailed in the following sections.

2.3.2.3 Informational Filtering

Information matrices are the dual of covariance matrices in the Gaussian case (Al Hage et al. 2019; Durrant-Whyte et al. 2001). That is, an information matrix noted \mathbf{I} (not to be confused with I the identity matrix) is defined as the inverse of the covariance $\mathbf{I} = \mathbf{P}^{-1}$. This means that if the covariance matrix conveys the uncertainty of a state, the information matrix conveys the *certainties*, or information contained in the state.

The KF can be formulated using information matrices using the Woodbury identity (Higham 2002). It is a common identity defined for arbitrary A, U, C, V matrices as:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(VA^{-1}U + C^{-1})^{-1}VA^{-1} \quad (2.27)$$

As such, the covariance matrix update of Equation (2.24) can be written as:

$$\mathbf{P}^{-1} = \mathbf{P}^{+^{-1}} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T \quad (2.28)$$

which can be interpreted as the gain of information that the observation brings to the state.

Depending on the dimensions of \mathbf{P} and \mathbf{R} , this form can be faster than the covariance form. As a general rule, if \mathbf{P} is of higher dimension than \mathbf{R} , then the traditional form (that inverts \mathbf{R}) is more suited than the informational form (that inverts \mathbf{P}) and inversely.

The informational form of filtering is a good way to introduce more efficient filtering methods for cooperative systems as we will see in the following.

2.3.3 Covariance Intersection Filtering

Covariance Intersection (CI), introduced in (S.J. Julier et al. 1997a) is a data fusion algorithm robust to arbitrary amounts of correlation between its inputs. It is very similar to the informational form of the KF (Equation (2.28)) but includes a scalar ω that makes the operation a convex combination:

$$\begin{aligned} \mathbf{P}^{-1} &= \omega\mathbf{P}^{+^{-1}} + (1 - \omega)\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \\ \mathbf{x} &= \mathbf{P} \left(\omega\mathbf{P}^{+^{-1}}\mathbf{x}^+ + (1 - \omega)\mathbf{H}^T\mathbf{R}^{-1}\mathbf{y} \right) \end{aligned} \quad (2.29)$$

The algorithm guarantees that as long as \mathbf{P} and \mathbf{R} are consistent, any value of ω will yield consistent results, as illustrated in Figure 2.8 (S.J. Julier et al. 1997a). This structural property of the covariance intersection is fundamental for fusing data exchanged between cooperative agents and potentially combined several times.

This includes completely ignoring one input or the other ($\omega = 0$ or $\omega = 1$) or taking half of both ($\omega = 0.5$). However, in practice optimizing ω for a particular goal is preferred.

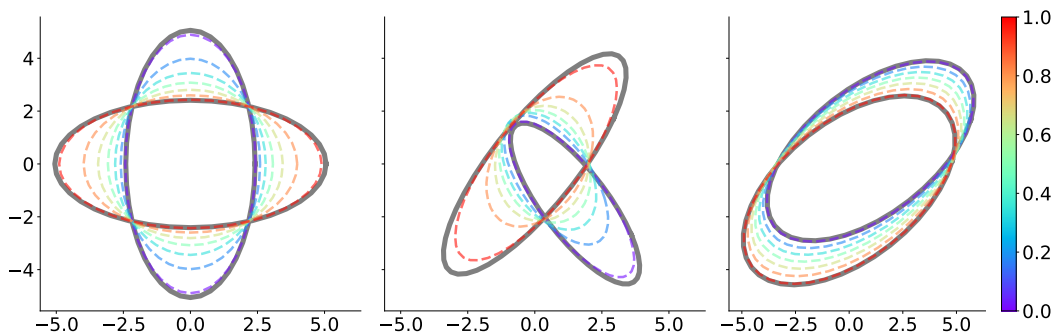


Figure 2.8: Covariance ellipsoid resulting from a CI fusion for multiple values of ω .

Volume Minimization ω is classically determined by minimizing the size of the resulting covariance matrix, which can be done by minimizing the determinant or trace of the resulting covariance matrix:

$$\begin{aligned} \omega &= \underset{\hat{\omega}}{\operatorname{argmin}} \det \left(\hat{\omega} \mathbf{P}^{+-1} + (1 - \hat{\omega}) \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \right)^{-1} \\ &= \underset{\hat{\omega}}{\operatorname{argmin}} \det^{-1} \left(\hat{\omega} \mathbf{P}^{+-1} + (1 - \hat{\omega}) \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \right) \end{aligned} \quad (2.30)$$

Compared to the Kalman update, which intuitively finds the ellipsoid included within the intersection, the CI finds the ellipsoid covering the intersection, as illustrated in Figure 2.9.

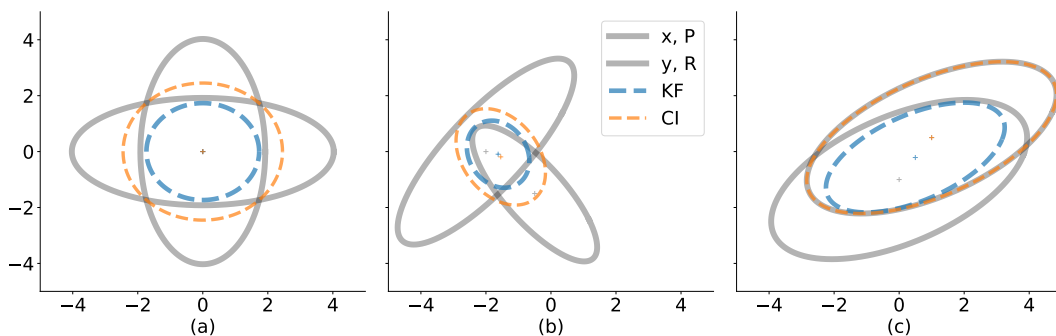


Figure 2.9: Comparison of Kalman update with CI over three situations: a baseline in which inputs are orthogonal and centred in (a), an edge case in which inputs are orthogonal but non centred in (b) and an edge case in which inputs are aligned and non centred in (c).

Several closed-form approximations of Equation (2.30) have been proposed and many variants have been developed to make the calculations faster.

For simplification, let us suppose in the following that the observation model matrix \mathbf{H} is the identity and that \mathbf{P} and \mathbf{R} are the same size.

Fast CI (Niehsen 2002) introduced the Fast Covariance Intersection (FCI):

$$\omega = \frac{\text{trace}(\mathbf{P}^+)}{\text{trace}(\mathbf{P}^+) + \text{trace}(\mathbf{R})} \quad (2.31)$$

which is simple but does not approximate Equation (2.30) particularly well in some cases, meaning that covariance of minimum volume is not always found.

Improved Fast CI To improve on the FCI, the Improved Fast CI (IFCI) have then been proposed in (Franken et al. 2005):

$$\omega = \frac{\det(\mathbf{P}^{+-1} + \mathbf{R}^{-1}) - \det \mathbf{R}^{-1} + \det \mathbf{P}^{+-1}}{2 \cdot \det(\mathbf{P}^{+-1} + \mathbf{R}^{-1})} \quad (2.32)$$

which yields very similar results at the expense of more computation.

Information Theoretic CI Another method is the Information-Theoretic Covariance Intersection (ITCI) of (Yimin Wang et al. 2012) that aims at providing consistent outputs over a larger number of situations:

$$\omega = \frac{D(\mathbf{x}^+, \mathbf{P}^+, \mathbf{y}, \mathbf{R})}{D(\mathbf{x}^+, \mathbf{P}^+, \mathbf{y}, \mathbf{R}) + D(\mathbf{y}, \mathbf{R}, \mathbf{x}^+, \mathbf{P}^+)} \quad (2.33)$$

where D is the Kullback-Leibler divergence, which is defined for Gaussian distributions as

$$D(\mathbf{x}, \mathbf{P}, \mathbf{y}, \mathbf{R}) = \frac{1}{2} \left[\ln \frac{\det \mathbf{R}}{\det \mathbf{P}} + (\mathbf{x} - \mathbf{y})^T \mathbf{R}^{-1} (\mathbf{x} - \mathbf{y}) + \text{tr}(\mathbf{P}\mathbf{R}^{-1}) - |\mathbf{x}| \right] \quad (2.34)$$

Inverse CI Other goals can also be targeted, such as the Inverse CI (ICI) introduced (Noack et al. 2017) which aims at providing less pessimistic but still consistent estimation. It uses the informational form (Equation (2.28)) to remove the common information $\langle \gamma, \Gamma \rangle$ between $\langle \mathbf{x}, \mathbf{P} \rangle$ and $\langle \mathbf{y}, \mathbf{R} \rangle$.

The common information is found by minimizing ω :

$$\begin{aligned} \omega &= \underset{\hat{\omega}}{\text{argmin}} \text{trace} \left(\mathbf{P}^{+-1} + \mathbf{R}^{-1} - (\hat{\omega} \mathbf{P}^+ + (1 - \hat{\omega}) \mathbf{R})^{-1} \right)^{-1} \\ \gamma &= \omega \mathbf{x}^+ + (1 - \omega) \mathbf{y} \\ \Gamma &= \hat{\omega} \mathbf{P}^+ + (1 - \hat{\omega}) \mathbf{R} \\ \mathbf{P}^{-1} &= \mathbf{P}^{+-1} + \mathbf{R}^{-1} - \Gamma^{-1} \\ \mathbf{x} &= \mathbf{P} \left(\mathbf{P}^{+-1} \mathbf{x}^+ + \mathbf{R}^{-1} \mathbf{y} - \Gamma^{-1} \gamma \right) \end{aligned} \quad (2.35)$$

Kalman Form of the CI Finally, according to (Héry et al. 2021) the CIF can be written in a Kalman form with an observation matrix \mathbf{H} as:

$$\begin{aligned} \mathbf{K} &= \frac{\mathbf{P}^+}{\omega} \mathbf{H}^T \left(\mathbf{H} \frac{\mathbf{P}^+}{\omega} \mathbf{H}^T + \frac{\mathbf{R}}{1-\omega} \right)^{-1} \\ \epsilon &= \mathbf{y} - \mathbf{H} \mathbf{x}^+ \\ \mathbf{x} &= \mathbf{x}^+ + \mathbf{K} \epsilon \\ \mathbf{P} &= (\mathbf{I} - \mathbf{K} \mathbf{H}) \frac{\mathbf{P}^+}{\omega} (\mathbf{I} - \mathbf{K} \mathbf{H})^T + \mathbf{K} \frac{\mathbf{R}}{1-\omega} \mathbf{K}^T \end{aligned} \quad (2.36)$$

with $\omega \neq 0 \neq 1$ optimized with

$$\omega = \underset{\hat{\omega}}{\operatorname{argmin}} \det \left((\mathbf{I} - \mathbf{KH}) \frac{\mathbf{P}^+}{\hat{\omega}} \right) \quad (2.37)$$

Cases where $\omega = 0$ or $\omega = 1$ are handled separately to avoid divisions by zero.

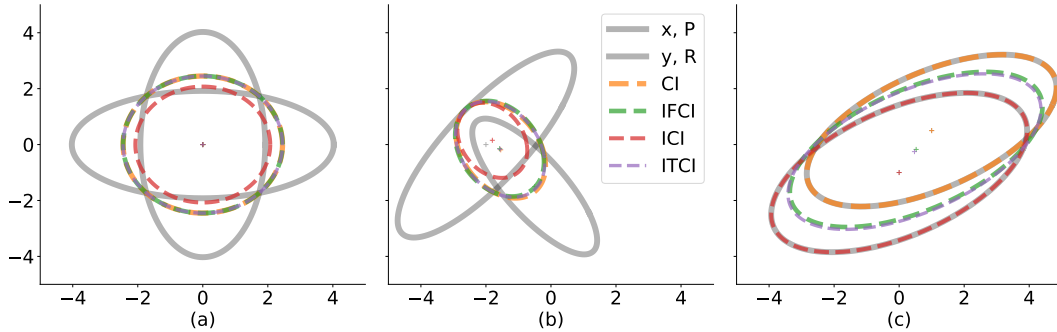


Figure 2.10: Comparison of CI, IFCI, ICI and ITCI over the three previous situations.

As it can be seen in Figure 2.10, all variants yield similar results in the nominal situations but differ in behavior for edge cases. The ICI on the other hand is the most different and can be seen as an intermediary between the Kalman update lower bound and CI upper bound.

2.3.4 Split Covariance Intersection Filtering

The Split CI (SCI) algorithm have first been mentioned in early work on CI by (Simon Julier et al. 2001) and have since been brought to the field of intelligent vehicles by (Hao Li et al. 2013). It aims at combining the optimality of the Kalman update and the consistency of the CI by splitting the error affecting the state estimate $\langle \mathbf{x}, \mathbf{P} \rangle$ in two parts:

- A *dependent* (temporally or spatially correlated) error ϵ_d whose covariance is characterized by matrix matrix \mathbf{P}_d ;
- An *independent* (perfectly uncorrelated) error ϵ_i whose covariance is characterized by matrix \mathbf{P}_i ;

such that the covariance matrix of the total estimation error $\epsilon = \epsilon_d + \epsilon_i$ is:

$$\mathbf{P} = \mathbf{P}_d + \mathbf{P}_i \quad (2.38)$$

Intuitively, the SCIF applies the CI to the dependent part and the Kalman update to the independent part (Pierre et al. 2018). Using a Kalman form, the prediction step is:

$$\begin{aligned} \mathbf{x}^+ &= \mathbf{F}\mathbf{x} \\ \mathbf{P}_d^+ &= \mathbf{F}\mathbf{P}_d\mathbf{F}^T + \mathbf{Q}_d \\ \mathbf{P}_i^+ &= \mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{Q}_i \end{aligned} \quad (2.39)$$

and the update step is:

$$\begin{aligned}
\mathbf{P}^+ &= \frac{\mathbf{P}_d^+}{\omega} + \mathbf{P}_i^+ \\
\mathbf{R} &= \frac{\mathbf{R}_d}{1-\omega} + \mathbf{R}_i \\
\mathbf{K} &= \mathbf{P}^+ \mathbf{H}^T (\mathbf{H} \mathbf{P}^+ \mathbf{H}^T + \mathbf{R})^{-1} \\
\mathbf{x} &= \mathbf{x}^+ + \mathbf{K} (\mathbf{y} - \mathbf{H} \mathbf{x}^+) \\
\mathbf{P} &= (\mathbf{I} - \mathbf{K} \mathbf{H}) \mathbf{P}^+ (\mathbf{I} - \mathbf{K} \mathbf{H})^T + \mathbf{K} \mathbf{R} \mathbf{K}^T \\
\mathbf{P}_i &= (\mathbf{I} - \mathbf{K} \mathbf{H}) \mathbf{P}_i^+ (\mathbf{I} - \mathbf{K} \mathbf{H})^T + \mathbf{K} \mathbf{R}_i \mathbf{K}^T \\
\mathbf{P}_d &= \mathbf{P} - \mathbf{P}_i
\end{aligned} \tag{2.40}$$

with ω optimized on:

$$\omega = \underset{\hat{\omega}}{\operatorname{argmin}} \det((\mathbf{I} - \mathbf{K} \mathbf{H}) \mathbf{P}^+) \tag{2.41}$$

Where \mathbf{Q}_d and \mathbf{Q}_i are matrices characterizing the covariance of the evolution model errors, and \mathbf{R}_d and \mathbf{R}_i characterize the covariance of the observation errors. One of the main difficulty with the SCIF is to estimate which parts of \mathbf{Q} and \mathbf{R} are dependent and which are not. This issue is discussed in Section 2.4.4.

2.4 Analysis of Covariance Intersection Filters

In this section, published in (Lima, Bonnifait, et al. 2021), we conduct a comparisons between the Kalman, CI and SCI filters in cooperative or standalone situations. This will illustrate several limitations of the KF and CI filters. These filters are compared in simulation using a simple multi-vehicle perception system.

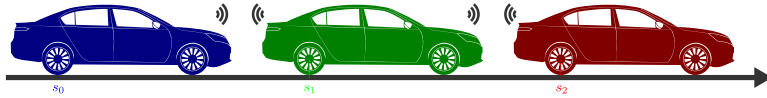


Figure 2.11: Situation composed of three vehicles in one-dimension.

In this system, illustrated in Figure 2.11, three vehicles follow each other on a one-dimensional line with constant velocity. The system is thus modeled with $O = \left\{ \mathbf{x}_1 = \begin{bmatrix} s_1 & v_1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} s_2 & v_2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} s_3 & v_3 \end{bmatrix} \right\}$. Each vehicle estimates the system state independently but they help each other by exchanging their perceptual information.

To do the data fusion, each vehicle loops over the following operations:

1. **Standalone update of the ego state:** The ego state is updated using on-board sensors providing a position measurement (for instance a GNSS

receiver). Letting Σ^{GNSS} be the variance of its measurement noise. This observation is modeled as:

$$\begin{aligned}\mathbf{H}^{\text{GNSS}} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \mathbf{R}^{\text{GNSS}} &= \Sigma^{\text{GNSS}}\end{aligned}\tag{2.42}$$

2. **Standalone update of other states:** The state of the other two vehicles is updated using on-board sensors providing relative position measurements (for instance a LiDAR). These are first transformed in a common frame of reference using the observer ego state, with the state uncertainty being propagated onto the observation. As such, letting Σ^{LiDAR} be the variance of the measurement noise, this observation is modeled as:

$$\begin{aligned}\mathbf{H}^{\text{LiDAR}} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \mathbf{R}^{\text{LiDAR}} &= \mathbf{H}^{\text{LiDAR}} \mathbf{P}_k \mathbf{H}^{\text{LiDAR}T} + \Sigma^{\text{LiDAR}}\end{aligned}\tag{2.43}$$

3. **Communication:** Estimates and covariances are exchanged with each other using wireless communication.
4. **Cooperative update:** The state of the other two vehicles is updated using the received measurements.

In the following simulations, the observation noises Σ^{GNSS} and Σ^{LiDAR} have been set to 0.1^2 and 0.2^2 respectively with the evolution noise set to 0.12^2 . The specific simulation code have been published in (Lima, Bonnifait, et al. 2021) and available online³.

2.4.1 CI Filtering Comparison with Kalman Filtering

In this first comparison, the estimation error of one vehicle about itself is plotted in Figure 2.12 for three filtering combinations:

- In blue, a KF is used in both standalone and cooperative steps;
- In red, a CIF is used in both standalone and cooperative steps;
- In green, a KF is used in the standalone step and a CIF is used in the cooperative step. This combination is denoted Kalman-Covariance Intersection Filter (K-CIF).

From these, it is clear that the KF and CIF yield similar results in terms of accuracy when using standalone information (Figures 2.12a and 2.12c), but vary significantly in confidence, with the CIF uncertainty bounds being far more cautious. However, when using cooperative information, the KF over-converges on a wrong velocity in Figure 2.12d, resulting in a rapidly increasing position error in

³<https://gitlab.utc.fr/multiception/multiception/-/snippets/59>

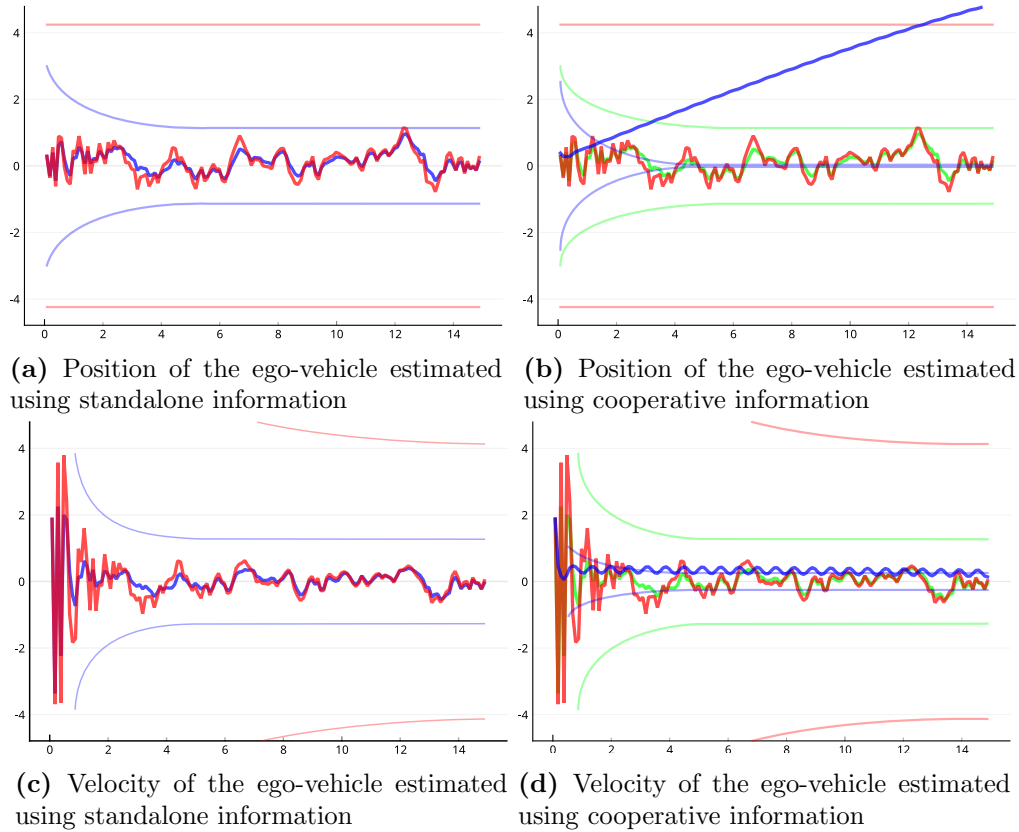


Figure 2.12: Estimation errors and $\pm 3\sigma$ uncertainty bounds. Estimate of the KF in blue, CIF in red and K-CIF in green. In (a) and (c), estimates of the KF and K-CIF are the same.

Figure 2.12b. At the same time, the CIF and K-CIF maintain an accurate and confident estimation. This can be explained by the fact that the cooperative step uses measurements with correlated errors, due to information loops introduced by the state exchanges. In the case of the K-CIF, the standalone results are as optimal as the KF by definition, and it can be seen that the CIF *replicates* the best estimate in the cooperative step.

This illustrates the resiliency of the CIF to information affected by correlated errors, but also highlights two of its limitations, further analyzed in Sections 2.4.2 and 2.4.3:

- The lack of convergence visible by the uncertainty bounds that are not reducing over time in Figures 2.12a and 2.12b. This is due here to observations whose covariances are aligned ;
- The slow convergence of unobserved quantities as visible by the red curve of Figure 2.12c between 0 and 2 seconds being very noisy.

2.4.2 Convergence Issues with Similarly Shaped Observation Covariances

A common complaint about the CIF is that it can result in pessimistic estimates and even loss of information in extreme cases (Seeliger et al. 2014). One of the reasons behind this is the lack of convergence of the filter. To illustrate this, consider a common filtering situation:

1. A single sensor provides similarly shaped observation covariances regularly. Here, shape refers to the shape of the covariance matrix. For example in Figure 2.10, covariances of (a) and (b) have different shapes while in (c) they are similarly shaped, or *quasi aligned*;
2. The first measurement initializes the state;
3. At the next time step, the state is predicted, which increases its covariance matrix;
4. A new measurement with similar looking covariance is incorporated using Equation (2.29). A predicted observation $h(x^+)$ is drawn from the predicted state. As ω is found to minimize the resulting covariance volume, it will naturally tend to ignore the predicted observation and focus on the measurement. This is because the predicted observation covariance is broadly-speaking an increased version of the past measurement covariance. Taking any part of the filtered covariance cannot reduce the resulting covariance by definition, and in most cases would even increase the resulting volume. As such, the computed ω gives almost all the weight to the measurement.

In such cases, the CIF has a tendency to produce an output that closely resembles the observation. As opposed to the intuitive result illustrated in Figure 2.13, the CIF does not perform the intersection of two intervals as it uses only the covariance and not the state estimate (Héry et al. 2017). As such, when fusing covariances aligned with one another, no uncertainty reduction occurs. This means that the CIF is best suited with orthogonal, or complementary sources, or at least when inputs are already filtered.

2.4.3 Slow Convergence with Partial Measurement

In most literature about the CIF, the measurement model is rarely mentioned and when it is, it is assumed complete (i.e. the state is completely observed) (Hao Li et al. 2013; Pierre et al. 2018). However, while partial measurements (i.e. \mathbf{H} is not full rank) are well handled by the KF, the CIF and its convex combination tends towards predicted states. This is because in order to integrate a partial measurement, the optimization of Equation (2.30) or Equation (2.37) has to balance between gaining information on the measured part and losing information on the unmeasured part.

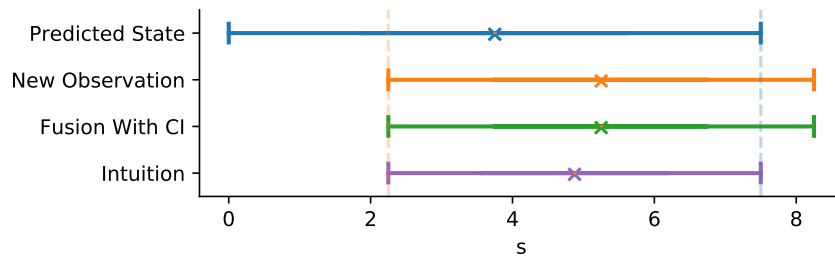


Figure 2.13: Illustration of CI in a simple one dimension case using state interval representation of Figure 2.4. A blue predicted state is fused with an orange observation, resulting in the green fusion. It is different from the intuition one might have about the "intersection" of intervals represented in purple.

Example 2.10:

Consider a static two-dimensional object $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}$. Due to sensor limitation, only its x coordinate is measured:

$$\mathbf{P} = \begin{bmatrix} 2 & 0.1 \\ 0.1 & 2 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 1 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The curves in Figure 2.14 correspond to the volume of resulting covariances, which is what ω is optimized on. It can be seen that both information and Kalman form give the same results, in particular that the volume tends to infinity as ω gets close to 0.

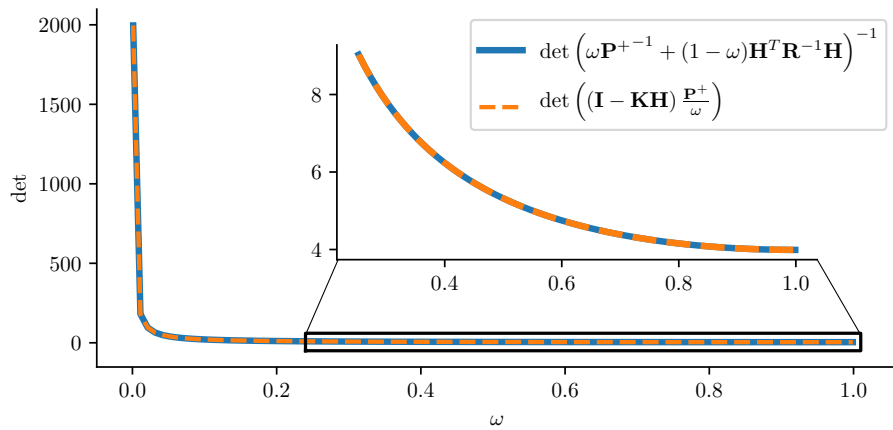


Figure 2.14: Curve of the fused covariance matrix determinant for ω between from 0 to 1 in the case of a partial measurement. The region between 0.25 and 1 is zoomed in.

More visually, on Figure 2.15, one can see that in order to improve the x estimate, the y estimate must be worsened. A compromise is found close to ignoring y because the measurement covariance is significantly smaller than that of the estimate.

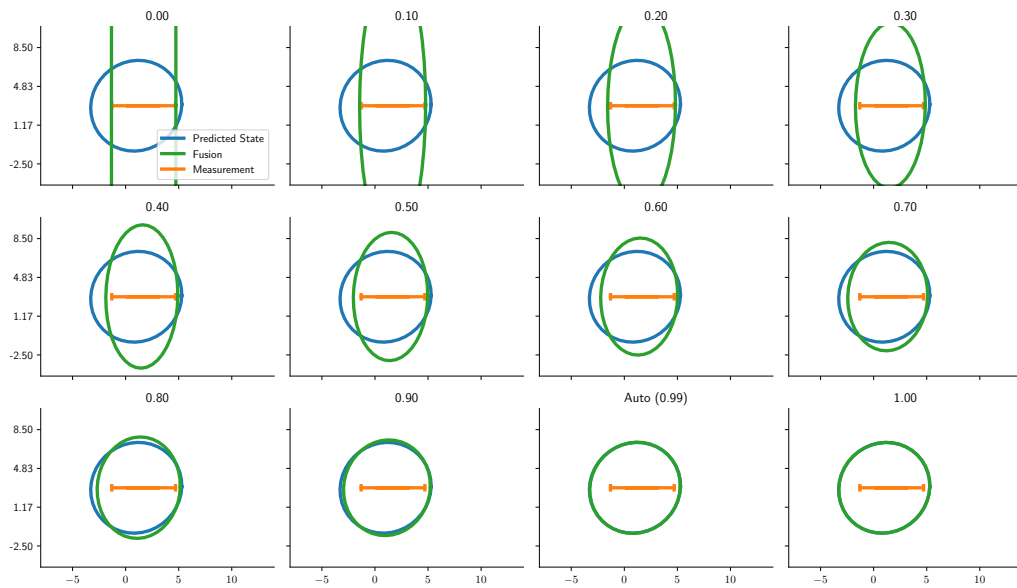


Figure 2.15: Fusion of a state (blue) and partial measurement (orange) for different values of ω . The optimal ω in the sense of Equation (2.30) is denoted *auto* and equals 0.99.

In practice, this effect might be counteracted by the issue of similarly shaped observation covariances, such that the system is able to converge, although slowly. As these issues do not appear when measurements are complete and complementary, we can conclude that the CIF is adapted to fuse complete states but not to filter incompletely observed states. Combining a KF with a CIF as in Figure 2.12 thus seems to be a good solution. However, this requires an explicit separation

between standalone and cooperative data as well as having to manage update functions manually to respect that separation. In the next section, we study how the SCIF can be used to remove this requirement.

2.4.4 SCI Comparison with a Kalman-CI Combination

In order to assess the performance of the Split Covariance Intersection Filter (SCIF), the same simulation as in Section 2.4.1 is used. Figure 2.16 summarizes the results on a given vehicle and compares it to a combination of Kalman updates for standalone data and CI for cooperative data.

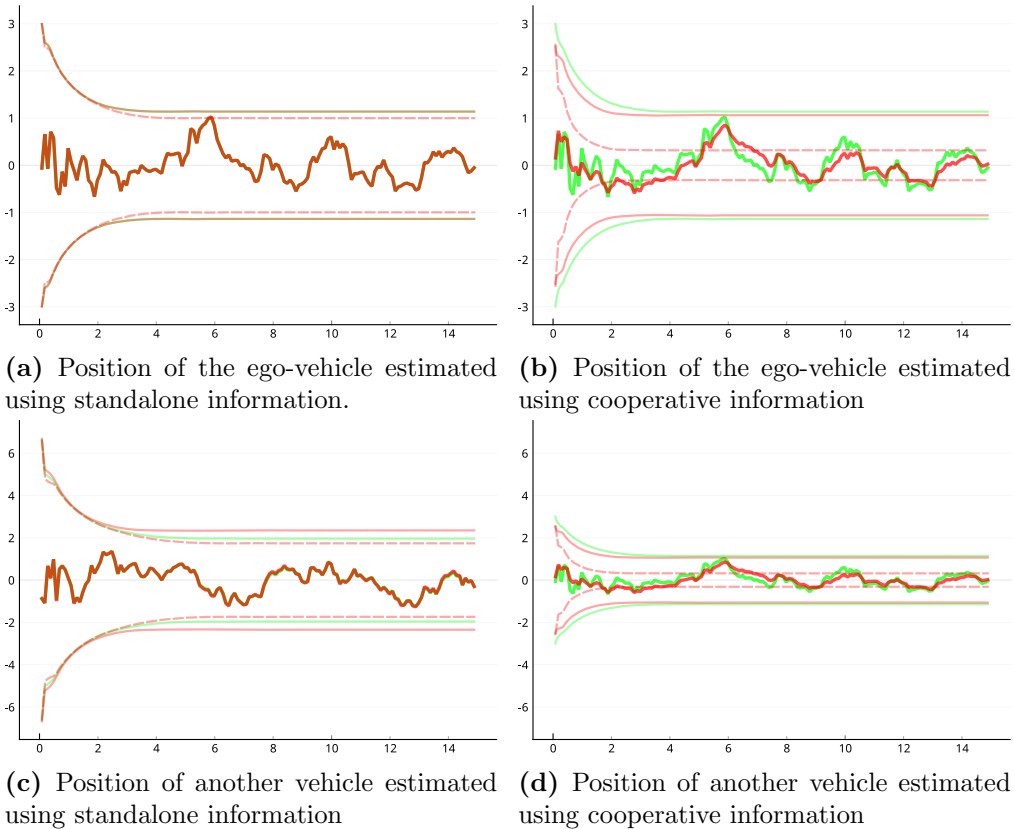


Figure 2.16: Estimation errors and $\pm 3\sigma$ confidence domains. KF+CI in green and SCIF in red. Continuous lines are the total confidence domains and dashed the independent part. In (a), curves are superimposed.

When a vehicle updates its state using only standalone information, the Figure 2.16a the SCIF yields quasi-identical results to the KF. This is expected, as the SCIF resolves to a KF in the absence of dependent uncertainty. However, one can see the apparition of a small amount of dependent uncertainty. This is due to the evolution model that generates dependent uncertainty as \mathbf{Q}_d is not null. This is particularly noticeable in Figure 2.16b, where the cooperative CI copies the standalone estimate that had a mean quadratic error of 0.30 m, while the split CI slightly improves it bringing the mean quadratic error to 0.25 m. This is due to the remaining independent noise that is processed by the Kalman component of the SCIF while consistency is maintained thanks to the dependent noise that is

processed by the CI component. As mentioned in Section 2.4, pre-filtering inputs resolves most CI shortcomings, which is in essence what the SCIF does. When estimating the states of other vehicles, these conclusions still hold but another effect is also visible. As the current ego-position is used to estimate the position of another vehicle, the ego-position covariance is added as a source of dependent observation noise. This introduces dependent noise in the filtered state that thus produces more cautious estimates, as visible by the confidence bound being larger than the KF. These curves do however bring an issue to our attention: the SCIF can produce inconsistent estimates. This is because it resolves to a KF when there is no dependent noise. If no dependent noise is introduced, either by the evolution model or by the observations, the same issue of over-convergence as in Figure 2.12 can appear.

These conclusions lead us to consider in the next section the tuning of the error covariance matrices used by the SCI filter.

2.4.5 Tuning SCIF Evolution and Observation Models

As dependent noise is central to the consistency of the SCIF, its introduction either through the evolution or observation model has to be well understood. Existing work (Hao Li et al. 2013) and (Pierre et al. 2018) tend to interpret dependent evolution noise as a way to model the temporal correlation of an object, which is a good start but insufficient to tune a whole filtering scheme.

Let us define

$$\begin{cases} \mathbf{Q}_d = \nu \mathbf{Q} \\ \mathbf{Q}_i = (1 - \nu) \mathbf{Q} \end{cases}, \quad \nu \in [0, 1] \quad (2.44)$$

that is, tuning Equation (2.38) is reduced from two matrices to one matrix and a scalar ν that works as a dependency factor. A situation where a subset of the covariance matrix is dependent and another is independent is not common, in particular in the field of perception.

In a SCIF, the independent evolution covariance \mathbf{Q}_i still characterizes the model-introduced error as classically done in Kalman filtering. The dependent evolution noise is used to represent the temporal correlation of a filtered state, but also to model the part of the estimation error that has an unknown degree of correlation with peers in a cooperative situation.

To tune them, we propose the following methodology:

1. Tune \mathbf{Q} with $\nu = 1$ until results are satisfying using classic metrics like accuracy and consistency,
2. Progressively reduce ν towards 0 until consistency stop meeting the required consistency performance.

For the covariance matrix of the observation noise, let us define the following tuning strategy with another real value γ to fix:

$$\begin{cases} \mathbf{R}_d = \gamma \mathbf{R} + \mathbf{R}_e \\ \mathbf{R}_i = (1 - \gamma) \mathbf{R} \end{cases}, \quad \gamma \in [0, 1] \quad (2.45)$$

R_e represents the part of the observation noise that is not dependent on the value of γ . When tuning, it is supposed known or given by an external computation. It can model uncertain a priori information or pose uncertainty propagation.

When tuning the covariance matrix of a sensor, \mathbf{R} can be obtained with classical methods (e.g: error plotting or consistency evaluation as in Section 3.6).

The γ factor is useful to manage cross-correlation between errors but it can also be used to manage temporal correlation due to the processing applied on sensor data.

For example:

- The Mobileye smart camera contains a tracker that does not provide a white observation noise. Because the part of dependent covariance cannot be assessed since it is a black box, a cautious choice is to choose a model with a value of ν close to 1;
- A LiDAR point cloud processing stage can be modeled with a value of ν close to 0 because two consecutive scans can be reasonably assumed to have decorrelated measurement errors;
- A processing that uses multiple sensor outputs such as LiDAR buffering (Lima, Welte, et al. 2020) or SLAM can be modeled with a value of ν tuned to characterize multiple factors such as the number of outputs used or how they are combined.

Fusing estimates provided by a cooperative peer does not require particular measures. Indeed, temporal and spatial statistical dependencies should be encapsulated in \mathbf{P}_d by each peer.

For example, let $(\mathbf{x}^{\text{other}}, \mathbf{P}_i^{\text{other}}, \mathbf{P}_d^{\text{other}})$ be the communicated estimate. It can be fused with the local estimate by applying:

$$\begin{aligned} \mathbf{y} &\leftarrow \mathbf{x}^{\text{other}} \\ \mathbf{R}_d &\leftarrow \mathbf{P}_d^{\text{other}} \\ \mathbf{R}_i &\leftarrow \mathbf{P}_i^{\text{other}} \end{aligned}$$

2.5 Conclusion

In this chapter, we have introduced several methods to decentralized data fusion that will serve as a basis for the rest of this manuscript.

For symbolic information, we have reviewed belief functions, which provide a general framework to the representation and data fusion of uncertain information at the symbolic level. Several fusion operators have been presented which will be useful to manage the existence of tracked objects and the trust in other agents with which a vehicle cooperates to improve its perception. Among these operators, the cautious rule seems promising thanks to its robustness to dependency between the information sources. However, it is known to have a lack of convergence in practice. We will therefore prefer to use the conjunctive rule in the rest of this work. It is interesting to note this issue is, to some extent, similar to that of the CIF which has to be combined with a KF in order to bring more information on the estimated state. It could be interesting to derive a fusion rule resembling the SCIF by splitting dependent and independent symbolic information and combining a cautious and conjunctive rule.

For metric information, we have introduced Bayesian state estimation and filtering. We have shown that the CIF or SCIF are especially adapted to decentralized data fusion as they are resilient to arbitrary levels of correlation. In particular, the SCIF can provide more accurate estimates while maintaining consistency in the case of information redundancy. This is a powerful and generic data fusion method, provided that one knows how to tune its parameters. We have proposed a strategy to do so. SCIF will be the metric fusion method that will be used in the rest of this work.

In the next chapter, these tools are used to represent several aspects of objects detected using our experimental platform, such as the object states and existence estimation.

Chapter 3

Sensor Processing and Tracking

Contents

3.1	Introduction	51
3.2	Objects and Free Space Detection	51
3.3	Multi-Object Tracking	60
3.4	Perception Evaluation	64
3.5	Description of the Perception System Used in this Work	69
3.6	Evaluation	78
3.7	Conclusion	82

3.1 Introduction

Perception is a key component of any autonomous robotic system. It makes robots able to adapt to dynamic and opened environments. This is even more important with autonomous vehicles, whose environment might contain Vulnerable Road Users (VRU). The focus of this manuscript is decentralized multi-sensor data fusion. However, due to restrictions on the communication medium discussed later, raw sensor data cannot be communicated. Relevant perception must thus first be extracted from raw data by each vehicle. This process is described in this chapter, using LiDAR sensors as an illustration. First, we review algorithms used in the literature to extract meaningful perceptual information (i.e: objects, free space), track and evaluate it in Sections 3.2 to 3.4. We then present our experimental setup and software implementation and experimental results in Sections 3.5 and 3.6.

3.2 Objects and Free Space Detection

Sensors used for perception (e.g. cameras, LiDARs) provide complex data. Interpreting their content is called detection and starts with cleaning up that data for downstream algorithms. This task is also called pre-processing and is described

in Section 3.2.1. Then several approaches for detecting objects and free-space are introduced in Sections 3.2.2 to 3.2.4 and are illustrated using our sensors and implementations. For additional literature on the subject, the reader may refer to (Mao et al. 2022; Qian et al. 2022).

3.2.1 Sensor Pre-Processing

Raw sensor data often require several steps before they can really be processed. For example, cameras use lenses to gather light, which distorts the resulting image. For LiDAR sensors, pre-processing starts with coordinate change. Indeed, as will be introduced later in Section 3.5.1.1, current automotive 3D LiDARs work by rotating laser beams. The lightest and thus preferred way to retrieve point-cloud information is a list of ranges. As it is easier to work in Cartesian space, ranges are first transformed using the following:

$$\begin{cases} x &= r_i \cdot \sin(\theta_i) \cdot \cos(\psi_j) \\ y &= r_i \cdot \sin(\theta_i) \cdot \sin(\psi_j) \\ z &= r_i \cdot \cos(\theta_i) \end{cases} \quad (3.1)$$

with r_i the i -th range measure associated with the i -th angle of the LiDAR head θ_i and ψ_j the orientation of the j -th laser beam on the rotating head.

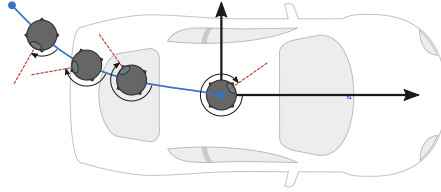


Figure 3.1: Several successive poses of a LiDAR head as it rotates while the vehicle moves.

If the LiDAR is attached on a moving vehicle, the resulting point-cloud will be distorted. This is due to the significant acquisition time of LiDAR measurements that is in the order of 100 ms. During this time, the vehicle moves, as illustrated in Figure 3.1, which results in points being projected in the wrong place up to dozens of centimeters if vehicle motion is not compensated. A solution to this is to keep track of the vehicle's kinematics $(\dot{x}, \dot{y}, \dot{\theta})$ using an IMU and wheel speed sensors, which is sampled at a much higher frequency. When a LiDAR rotation is over, the time t_i associated to each point i is retrieved. Assuming 2D motion, each point p_i can be undistorted by first estimating the LiDAR's pose with respect to the final one at time t_i , $\langle dx_i, dy_i, d\theta_i \rangle$ by back-integrating the motion between t_i and the point-cloud global time t_0 . By denoting Δt the sampling period between two beams, we can write:

$$\begin{cases} dx_i &= \sum_{j=i}^1 \Delta t \sqrt{\dot{x}_j^2 + \dot{y}_j^2} \cdot \cos(\dot{\theta}_j) \\ dy_i &= \sum_{j=i}^1 \Delta t \sqrt{\dot{x}_j^2 + \dot{y}_j^2} \cdot \sin(\dot{\theta}_j) \\ d\theta_i &= \sum_{j=i}^1 \Delta t \dot{\theta}_j \end{cases} \quad (3.2)$$

Points can then be transformed as

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}^{\text{undist}} = \begin{bmatrix} \cos(d\theta_i) & -\sin(d\theta_i) & 0 & dx_i \\ \sin(d\theta_i) & \cos(d\theta_i) & 0 & dy_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}^{\text{distor}} \quad (3.3)$$

which yields a list of undistorted Cartesian points $P = \{p_i\}_i$. For example, on our vehicles, this effect results in blurry shapes and shifts that are best illustrated when point-clouds are accumulated as the vehicle moves, as shown in Figure 3.2.

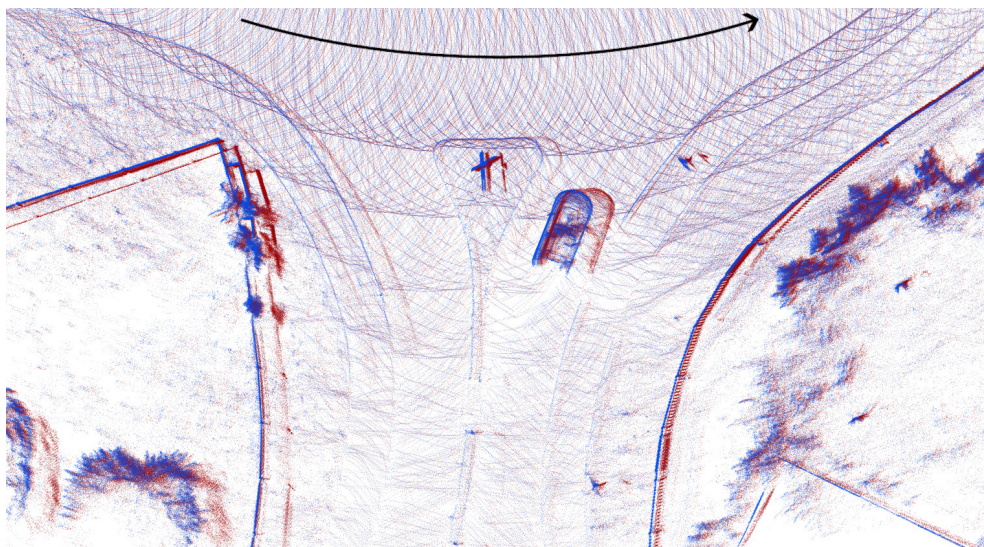


Figure 3.2: point-clouds accumulated over 5 seconds of moving in a roundabout. The red point-cloud shows the distortion caused by the vehicle motion, and the blue point-cloud are their undistorted counterpart. Note the car and sign duplication caused by the varying point of view and the fuzziness of the left and right fences.

The next step is generally to filter the point-cloud to reduce its size (and thus its complexity). Filtering can be applied as a function of range, height or intensity. These filters are defined as keeping only points whose characteristics are within a pre-defined range;

$$\begin{aligned} P^{\setminus r} &= \{p_i\}_{i \in P, r_{\min} < \|p_i\| < r_{\max}} \\ P^{\setminus z} &= \{p_i\}_{i \in P, z_{\min} < z_i < z_{\max}} \\ P^{\setminus i} &= \{p_i\}_{i \in P, i_{\min} < i_i < i_{\max}} \end{aligned} \quad (3.4)$$

This can be used to reduce the area of interest as in Figure 3.3 or even highlight features of the environment as in Figure 3.4.

More advanced approaches filter points belonging to the ground and others, such as (Zermas et al. 2017) or (Jiménez et al. 2021). The first iteratively estimates the ground plane and separates points based on their distance to that plane. The second places a ground point below the sensor then recursively propagates the

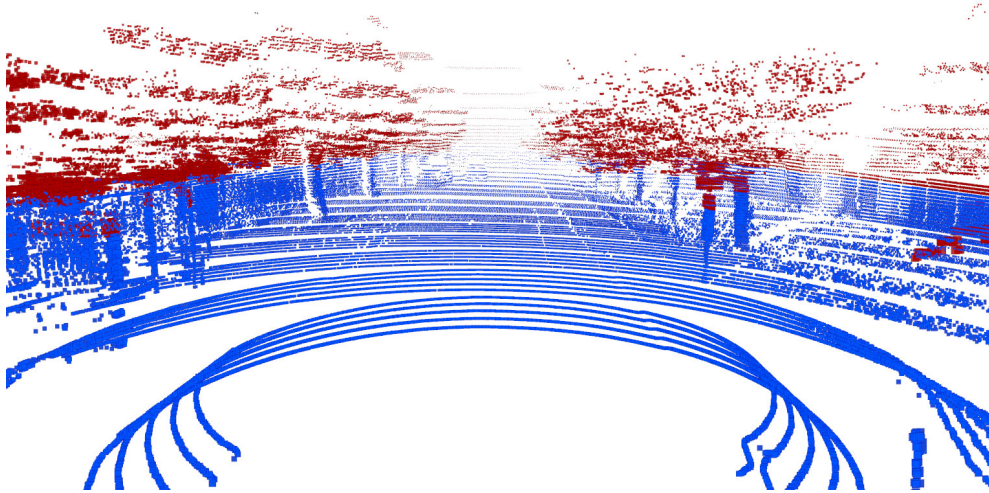


Figure 3.3: Reference point-cloud in red. $P^{\setminus rz}$ filtered point-cloud in blue. Points are cut above the sensor, below the ground and further than 150 m to limit the area of focus. Both are accumulated over a second for the sake of clarity.

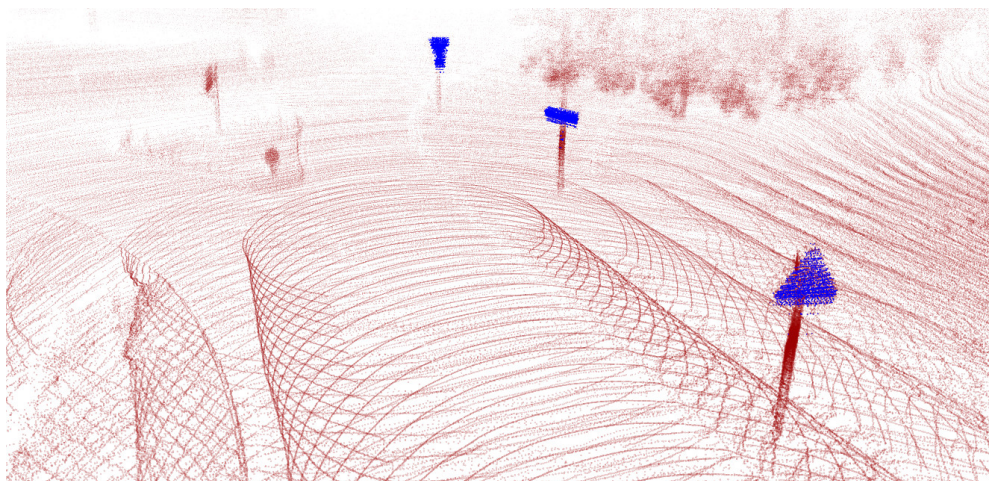


Figure 3.4: Original point-cloud in red and intensity filtered point-cloud in blue. Notice how signs on the right are highlighted while those on the left are not. This is because only one side of road signs is reflective.

ground class to neighbors depending on their absolute height and relative slope. The method of (Jiménez et al. 2021) provides better results for the same level of complexity as (Zermas et al. 2017) and as such is the preferred method in this work.

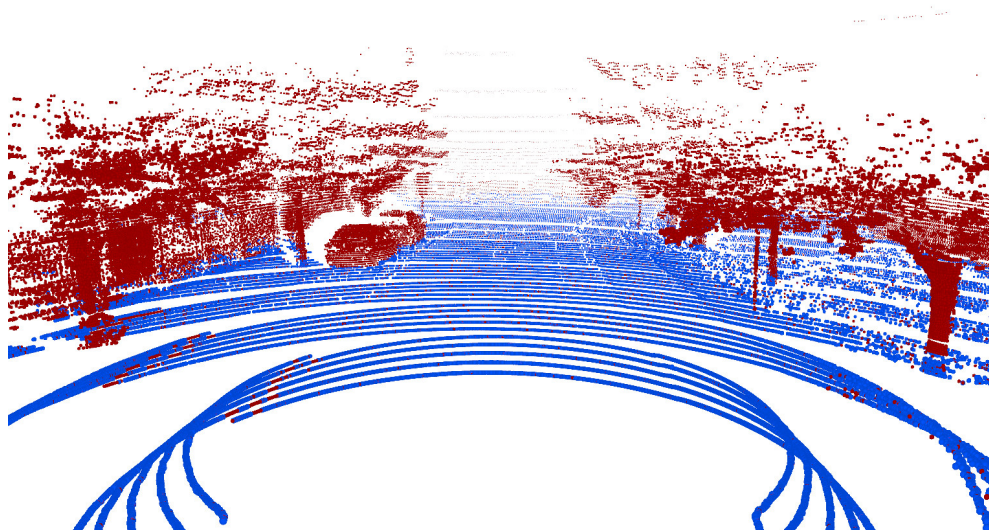


Figure 3.5: Reference point-cloud in red, ground point-cloud filtered with the method of (Jiménez et al. 2021) in blue.

3.2.2 Model Based Object Detection

Clustering points is useful to exploit the spatial dependency of points. The principle is that points close together must be a part of the same object. There are varying degrees of complexity behind clustering algorithms. The simplest one is Euclidean clustering where the distance between every point is computed, and classes are propagated when distances are below a given threshold. More complex algorithms can be used, in particular by replacing the distance with a cost that accounts for many aspects. As an example, take Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996) that also computes the number of neighbors before propagating a class, thus preventing noise propagation through low density regions. Finally, it can be seen that these algorithms are heavy (complexity of $O(N^2)$ for Euclidean clustering and $O(N^3)$ for DBSCAN). This is because points are not considered to be organized and thus all permutations must be computed. To limit the required computations, there are methods to organize point-clouds, such as octrees (Meagher 1982) that recursively split space in two, allowing for rapid binary tree searches. Clustering algorithms can also be adapted to certain point-cloud organization, such as the Scan Line Run (SLR) (Zermas et al. 2017) that clusters along rings then merges classes between rings.

Clusters correspond to a given shape that can be represented in many ways, but the more generic one is the bounding box. In 3D, it is composed of a position x, y, z , orientation φ, ψ, θ and size l, w, h . As we focus on 2D terrestrial navigation,

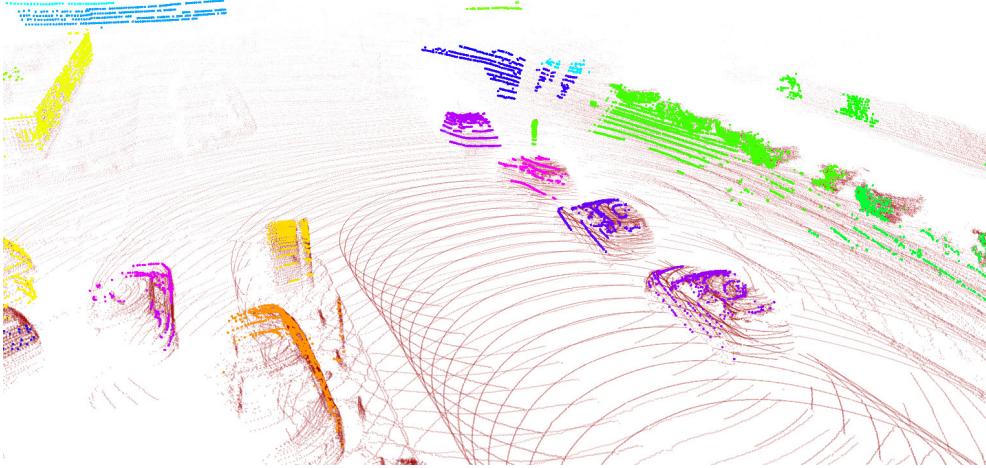


Figure 3.6: Reference point-cloud in red and clustered point on top, where different colors mean different clusters.

Algorithm 1 Bounding Box Fitting

Input: Cluster P

Output: Position x, y, z , heading θ and dimensions l, w, h

$$\begin{aligned}
 \check{x}, \check{y}, \check{z} &\leftarrow \text{PCA}(P) && \triangleright \text{Sorted principal component vector of } P \\
 \theta &\leftarrow \text{atan2}(\check{y}, \check{x}) && \triangleright \text{atan taking quadrant into account} \\
 P_{\theta} &\leftarrow \text{rotate}(\theta, P) && \triangleright \text{Rotates points of } P \text{ by } \theta \\
 \begin{bmatrix} x & y & z \end{bmatrix} &\leftarrow \begin{bmatrix} \frac{x^{\max} + x^{\min}}{2} & \frac{y^{\max} + y^{\min}}{2} & \frac{z^{\max} + z^{\min}}{2} \end{bmatrix} \\
 \begin{bmatrix} l & w & h \end{bmatrix} &\leftarrow \text{rotate}\left(-\theta, \begin{bmatrix} x_{\theta}^{\max} - x_{\theta}^{\min} & y_{\theta}^{\max} - y_{\theta}^{\min} & z_{\theta}^{\max} - z_{\theta}^{\min} \end{bmatrix}\right)
 \end{aligned}$$

φ and ψ can be ignored. The procedure to fit a bounding box over a cluster P^l , given in Algorithm 1, is to find its main orientation through Principal Component Analysis (PCA), aligning with that orientation, computing the size as the distance between the maximum and minimum points in the x, y, z axes then transforming the size back in the original orientation.

However this method gives unaligned boxes as depicted in Figure 3.7. This is due to the PCA yielding an erroneous orientation depending on the observed shape. This problem is better understood when illustrated, as in Figure 3.8.

3.2.3 Deep Learning Based Object Detection

When something is too complex to properly be modeled, machine learning methods that learn what to detect can be used. Neural networks have been used more and more in recent years as they can process highly complex data in exchange for massive amounts of training data. It is hard to keep track of every advances in the field, in particular for cameras whose state-of-the-art results are rapidly improving. Because of this, only generic approaches will be reviewed here.

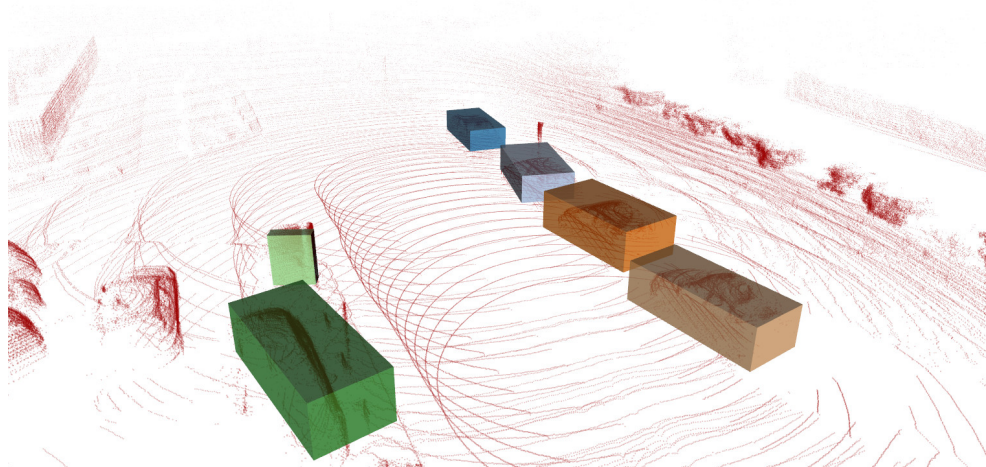


Figure 3.7: Accumulated reference point-cloud in red. Snapshot bounding boxes are fitted over each cluster using Algorithm 1. Only objects close to the road are shown here. One can notice the bad alignment of the green bounding box here because the vehicle is partially seen from the side.

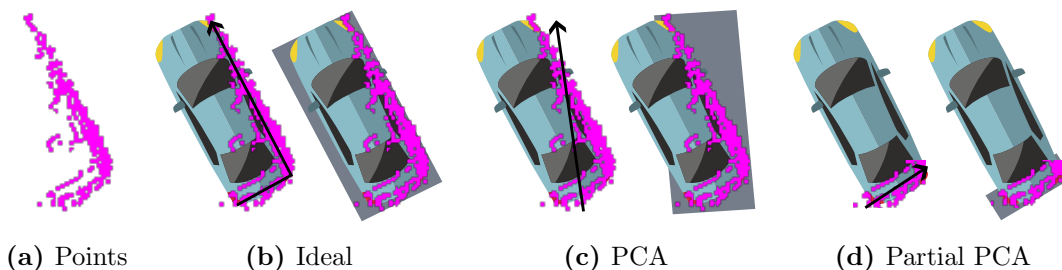


Figure 3.8: Bird eye view of a particular cluster point-cloud, heading and bounding boxes. The ideal heading and bounding boxes differs from what the PCA yields due to the principal component being across the car width and length. This issue is worse when a car is only one side is seen.

In the camera world, the best generic and fast deep-learning algorithm is YOLO, that exists in many versions, the latest at the moment of writing being the sixth (C. Li et al. 2022). The basic principle behind them is to divide the image in cells in which a Convolutional Neural Network (CNN) is ran to detect and classify shapes. The network output is a list of bounding boxes in image coordinates associated with class probabilities.

For LiDARs, early approaches used cylindrical projection (Milioto et al. 2019; Yuan Wang et al. 2018) or Bird Eye View (BEV) (Ku et al. 2018) to transform the point-cloud into an image with w , y , z , depth and intensity channels in order to reuse existing 2D CNN architectures. The output of such networks are 3D bounding boxes associated with class probabilities. The main issue with those methods is that they do not adapt well to new sensors and require re-training. This limitation is starting to be lifted with networks such as Cylinder3D (X. Zhu et al. 2021) that applies a CNN on a cylindrical voxel before propagating

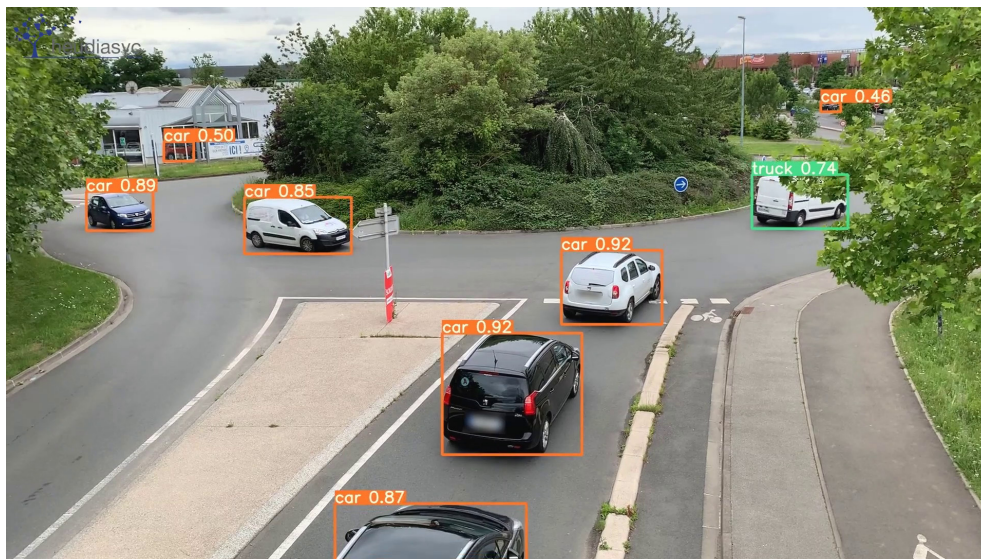


Figure 3.9: Image with bounding boxes classified by YOLOv5. Results obtained within the Tornado project taking place in the city of Rambouillet.

voxel classes to points within, (Deng et al. 2021) that applies a recursive CNN to Cartesian voxels or (Lang et al. 2019) that uses vertical columns to organize the point-cloud. Here, the output is a class probability associated with each point. The results of such a network on our LiDAR data is illustrated in Figure 3.10. Finally, it is possible to tightly couple images and point-clouds before detection, as in (X. Wu et al. 2022; Zhang et al. 2021) where the authors project a point-cloud into a camera image and apply a deep learning detection algorithm in the image to fuse the form and distance information. In most instances, networks are trained on the Kitty dataset (Behley et al. 2019; Geiger et al. 2012) and as such are particularly good at detecting and classifying cars.

3.2.4 Free Space

In most approaches reviewed previously, objects are detected as bounding boxes, which can only express where objects have been seen. However for planning and maneuvering, it is where objects are not present that is important. Naively considering that everywhere an object is not explicitly seen must be free is erroneous as objects can simply be missed, and thus a complementary representation of what is measured as free is mandatory.

Most approaches rely on deep neural networks to detect road from camera or point-clouds (Z. Chen et al. 2019) though LiDAR-based methods can take advantage of the strong geometric information. For example, the approach of (Capellier et al. 2018) is to classify point-clouds as ground and non-ground (see Section 3.2.1) and considers that areas where the incident laser beam was low enough are drivable. When free/non-free segmentation is done in the image plane, the result can be transformed in 3D space by back-projecting the image plane using an homography. This process, called Inverse Perspective Mapping (IPM) approximates a Bird Eye View as long as the road is planar.

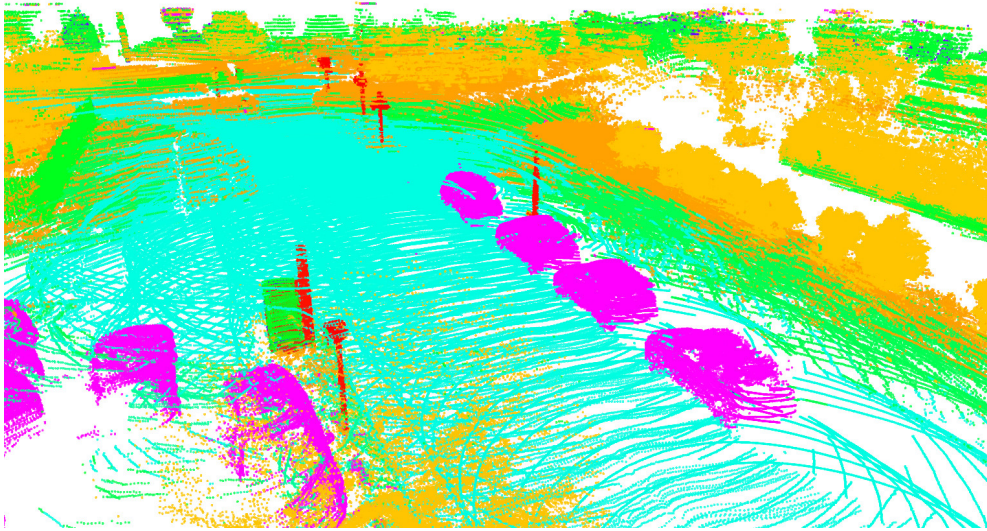


Figure 3.10: Accumulated points classified by Cylinder3D. In pink are point classified as cars, road in cyan, signs in red, vegetation in orange and infrastructure in green. Results obtained in the city of Compiègne.

Free areas can be represented either sparsely or densely. Sparse representations (sometimes also called parametric representations) use polygons and describe the 2D boundaries of free areas as viewed from the top. A good example of this is (Luthardt et al. 2017) where vertices encode where and free space stops and whether the transition is caused by an obstacle or simply getting out of field of view. In addition, they provide a method to fuse such polygons from different points of view. Dense representations include grids, where the ground plane is discretized with cells of fixed size that encode the state of space at a particular point.

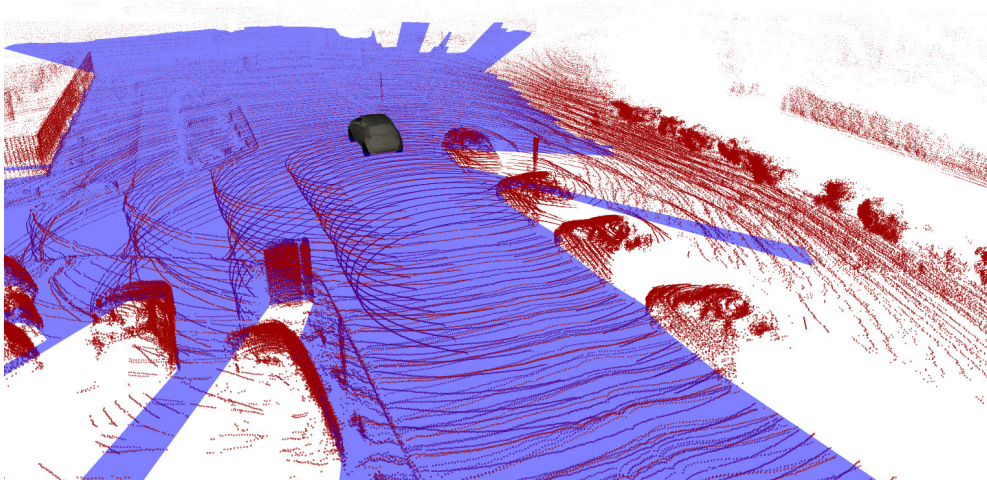


Figure 3.11: Point-cloud and measured free space polygon. As the polygon is computed from the vehicle position, the vehicle is depicted in grey.

Originally, Bayesian grids have been proposed by (Elfes 1989) with cells containing the probability of an object occupying it. This has since been extended to evidential occupancy grids in which each cell contain a mass function defined on $\{\text{Free}, \text{Occupied}\}$ (Capellier et al. 2018; Moras et al. 2011; Nuss et al. 2018; Yu et al. 2014), which provides the advantage of explicitly representing unobserved areas. In a more planning-oriented manner, grids have been used in (Lacoste et al. 2021) to represent a spatial *hazard* in each cell. The collision risk of a path is computed through a function that accounts for objects mass and speed.

3.3 Multi-Object Tracking

Once raw data has been processed from sensors, it can be combined with other sensors. To do so requires a common world in the form of a Local Dynamic Map (LDM). A LDM is usually realized through *tracking*, where the states of multiple objects are filtered using the methods described in Section 2.3.2. However, there are more tasks to handle in a tracker than only estimating states and their covariances. First, there are multiple objects to track, meaning that the right observations must be matched with the right tracks. This will be presented in Section 3.3.1. In addition, because objects are not always in view or relevant and because sensors are not necessarily reliable, the lifespan of tracks must be managed, as studied in Section 3.3.2.

3.3.1 Data Association for Object Tracking

The problem of association deals with the matching of N objects with M observations. Indeed, as the observed objects might differ from the LDM either because of their dynamics or because of sensor noise, the correspondence between the two views must be realized. We suppose here that the two are synchronized and in the same frame of reference. As reviewed in (Vo et al. 2015), there are multiple ways to match tracks with observations that can be categorized in two families: hard and soft decision. The first decides a 1-to-1 matching between objects of the two views such that filtering can do a single update, while soft decisions delays the 1-to-1 matching or even does N -to- M matching on all objects of both views.

Hard Decisions The basic methodology in hard decision is to compute an association cost between all tracks and all observations then finding the association that yields the smallest overall cost. More formally, let d be the cost function that can be more or less complex depending on the dynamics of the scene being tracked. The simplest form simply measures the geometric distance between the track $\langle \mathbf{x}, \mathbf{P} \rangle$ and the observation $\langle \mathbf{y}, \mathbf{R} \rangle$:

$$d^{\text{euc}}(\mathbf{x}, \mathbf{y}) := |\mathbf{x} - \mathbf{y}| \quad (3.5)$$

A finer form uses the *Mahalanobis distance*, a distance normalized by the covariance, which provides better results when the covariance matrices \mathbf{P} and \mathbf{R} is significant. Assuming that both vectors share dimensions, it is defined as:

$$d^{\text{mah}}(\mathbf{x}, \mathbf{y}) := \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{P} + \mathbf{R})^{-1} (\mathbf{x} - \mathbf{y})} \quad (3.6)$$

Other non-metric quantities can be compared in the cost computation. (Duraismy et al. 2015) proposed to use Negative Log Likelihood Ratio (NLLR) to incorporate attributes such as the classification or dimensions in the cost computation. Another way to take the class associated with two objects \mathbf{x}_c and \mathbf{y}_c into account is using a confusion matrix \mathbf{C} that describes the cost of associating a class with another. For example, associating a car with a small truck is less surprising and thus less costly than associating it with a pedestrian, which is represented by a higher cost at $\mathbf{C}(\text{car}, \text{small_truck})$ than $\mathbf{C}(\text{car}, \text{pedestrian})$. The association cost can thus be written as:

$$d^{\text{con}}(\mathbf{x}, \mathbf{y}) := d^{\text{mah}}(\mathbf{x}, \mathbf{y}) + \mathbf{C}(\mathbf{x}_c, \mathbf{y}_c) \quad (3.7)$$

Finally, (N. Zoghby et al. 2013) proposed a generic framework to compare two objects and derive their *associativeness*. It uses belief functions defined on $\{A_{ij}, \mathcal{A}_{ij}\}$ being "objects i and j correspond or do not". The idea is to calculate mass functions for the different properties and combine them using the conjunctive rule as

$$m = m_p \odot m_v \odot m_c \quad (3.8)$$

Here, the mass functions refer to:

- Position, with α_p the degree of confidence, λ_p an arbitrary positive coefficient and $d_{p,ij}$ the Mahalanobis distance of the position

$$\begin{cases} m_p(\{A_{ij}\}) & = \alpha_p \cdot e^{-\lambda_p d_{p,ij}} \\ m_p(\{\mathcal{A}_{ij}\}) & = \alpha_p \cdot (1 - e^{-\lambda_p d_{p,ij}}) \\ m_p(\{A_{ij}, \mathcal{A}_{ij}\}) & = 1 - \alpha_p \end{cases} \quad (3.9)$$

- Velocity, with α_v the degree of confidence, λ_v an arbitrary positive coefficient and $d_{v,ij}$ the Mahalanobis distance of the velocity:

$$\begin{cases} m_v(\{\mathcal{A}_{ij}\}) & = \alpha_v \cdot (1 - e^{-\lambda_v d_{v,ij}}) \\ m_v(\{A_{ij}, \mathcal{A}_{ij}\}) & = 1 - \alpha_v \cdot (1 - e^{-\lambda_v d_{v,ij}}) \end{cases} \quad (3.10)$$

- Class with c the belief function for classification on $\Omega = \{\text{Car}, \text{Bicycle}, \dots\}$. Two objects with the same class are not necessarily the same, but two objects with different classes are likely not the same:

$$\begin{cases} m_c(\{\mathcal{A}_{ij}\}) & = \sum_{A \cap B \neq \emptyset} {}^i m(A) {}^j m(B) \\ m_c(\{A_{ij}, \mathcal{A}_{ij}\}) & = 1 - \sum_{A \cap B \neq \emptyset} {}^i m(A) {}^j m(B) \end{cases} \quad (3.11)$$

In any case, a first step called gating can be realized to save on computations, where obvious non-associations are excluded using simple distance metrics such as the Euclidean or Mahalanobis distance.

The previous methods are used to fill a cost matrix C which is used to find the best assignment using various algorithms. The most basic one is the Global Nearest

Neighbors (GNN) that applies optimization algorithms such as the Hungarian algorithm to the following problem

$$\min \sum_i \sum_j C_{i,j} X_{i,j} \quad (3.12)$$

where X is a boolean matrix such that $X_{i,j} = 1$ if and only if row i is assigned to column j . By doing so, the GNN finds the lowest overall cost.

Soft Decision Hard decision algorithms are vulnerable to cross trajectories between objects, as the cost computation can always be incomplete and minimizing it can lead to mis-associations in close situations. (Houenou et al. 2012) proposed to consider track history to decrease this effect but that approach can be computationally expensive. Another possibility is to avoid making ambiguous associations. In the Joint Probabilistic Data Association (JPDA) and Joint-JPDA family of approaches (Yaakov Bar-Shalom et al. 2009), all observations impact all tracks with a weighting factor that denotes the similarity between the two. The resulting state is thus conceptually a weighted sum of all its inputs. In the Multiple Hypothesis Tracking (MHT) family of approaches, the choice of association is delayed. Likely associations duplicate the LDM similar to parallel universes. When ambiguities are lifted and some universes are more likely than others, the less likely are pruned to maintain computation requirements low.

3.3.2 Track Management

Autonomous navigation takes place in highly dynamic environments where objects constantly get in and out of view and change relevance in the navigation task. To maintain the quality of objects in the LDM while keeping relatively low computation times, the lifespan of tracks must be managed, that is create, maintain and delete them. In general, this is done through a track attribute that gets updated and used to determine what to do with the track.

Standalone Existence Estimation In (Aeberhard 2017; Aeberhard et al. 2011), the existence of objects is done at two levels, sensor and fusion. At the sensor level, existence is represented by a Bayesian probability $p(\exists \mathbf{x})$. It is updated by first predicting the previous probability

$$\begin{cases} p(\exists \mathbf{x})^+ &= p^{\text{persist}}(\mathbf{x}) \cdot p(\exists \mathbf{x})^- + p^{\text{birth}}(\mathbf{x}) \cdot p(\bar{\exists} \mathbf{x})^- \\ p(\bar{\exists} \mathbf{x})^+ &= (1 - p^{\text{persist}}(\mathbf{x}) \cdot p(\exists \mathbf{x})^-) + (1 - p^{\text{birth}}(\mathbf{x}) \cdot p(\bar{\exists} \mathbf{x})^-) \end{cases} \quad (3.13)$$

where $p(\cdot)^-$ are the prior probabilities and $p(\cdot)^+$ are the predicted probabilities. It uses a birth probability p^{birth} that models how likely a new object is to appear and a persistence probability p^{persist} that models how much an object detected at time t should still be detected at time $t + 1$. The latter is defined as:

$$p^{\text{persist}}(\mathbf{x}(t)) = p^{\text{fov}}(\mathbf{x}(t)) \cdot p^{\text{occ}}(\mathbf{x}(t)|O(t)) \quad (3.14)$$

where

- p^{fov} models how likely an object is to be detected given the sensors specifications (e.g: range, opening);
- p^{occ} models how likely an object is to be hidden by another.

The existence probability is then updated as:

$$\begin{cases} p(\exists \mathbf{x}) &= \eta p^{\text{detect}}(\mathbf{x}) \cdot p(\exists \mathbf{x})^+ \\ p(\bar{\exists} \mathbf{x}) &= \eta p^{\text{clutter}}(\mathbf{x}) \cdot p(\bar{\exists} \mathbf{x})^+ \end{cases} \quad (3.15)$$

using normalizing factor η and a probability of detection modeled as:

$$p^{\text{detect}}(\mathbf{x}(t)) = p^{\text{fov}}(\mathbf{x}(t|t-1)) \cdot p^{\text{meas}}(\mathbf{y}(t|t-1)) \cdot p^{\text{quality}}(\mathbf{x}(t)) \quad (3.16)$$

where

- p^{meas} models how likely a track is to correspond to a real object. It can for example use the classification quality or prior information;
- p^{quality} models the track quality, for example using the successive innovations.

The probabilities of birth p^{birth} and clutter p^{clutter} are determined by analyzing the sensor.

At the fusion level, existence is represented with belief functions. The frame of discernment used is $\Omega = \{\exists, \bar{\exists}\}$ and combinations are realized with the conjunctive rule. Belief functions are predicted using discounting

$$m(t+1|t) = \chi m(t|t) \quad (3.17)$$

and can be corrected two ways. Either a track is associated, in which case a trust and visibility probabilities are used:

$$\begin{cases} {}^j m(\exists \mathbf{x}) &= p^{\text{trust}}(j) \cdot {}^j p^{\text{persist}}(\mathbf{x}) \cdot {}^j p(\exists \mathbf{x}) \\ {}^j m(\bar{\exists} \mathbf{x}) &= p^{\text{trust}}(j) \cdot {}^j p^{\text{persist}}(\mathbf{x}) \cdot [1 - {}^j p(\exists \mathbf{x})] \\ {}^j m(\exists \mathbf{x}, \bar{\exists} \mathbf{x}) &= 1 - p^{\text{trust}}(j) \cdot {}^j p^{\text{persist}}(\mathbf{x}) \end{cases} \quad (3.18)$$

or it is not associated, in which case

$$\begin{cases} {}^j m(\exists \mathbf{x}) &= 0 \\ {}^j m(\bar{\exists} \mathbf{x}) &= p^{\text{trust}}(j) \cdot {}^j p^{\text{persist}}(\mathbf{x}) \\ {}^j m(\exists \mathbf{x}, \bar{\exists} \mathbf{x}) &= 1 - p^{\text{trust}}(j) \cdot {}^j p^{\text{persist}}(\mathbf{x}) \end{cases} \quad (3.19)$$

The trust probability p^{trust} is used as a weighting factor meant to fill in the discrepancies between sensors, and in their case was derived from a ROC curve.

Finally, tracks are deleted, confirmed or un-confirmed using thresholds τ_d , τ_c and τ_{uc} as illustrated in Figure 3.12.

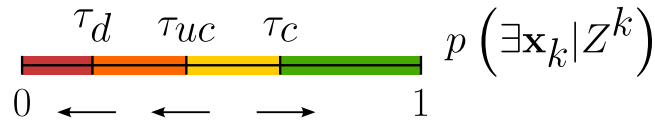


Figure 3.12: Object existence stages in track management (from (Aeberhard 2017)). Four zones are defined: scheduled for deletion in red, unconfirmed in orange and confirmed in green. The yellow zone is an hysteresis to let object that fell below confirmed to still be maintained.

Cooperative Existence Estimation On the other hand, in (N. E. Zoghby et al. 2014) objects are managed directly using belief functions on the frame $\Omega = \{\text{Object}, \text{NonObject}\}$. They are first created using the object age a , a sensor reliability factor β and an arbitrary positive coefficient k

$$\begin{cases} j_m(\text{Object}) & = \beta \cdot (1 - e^{-ka}) \\ j_m(\text{NonObject}) & = \beta \cdot (e^{-ka}) \\ j_m(\text{Object}, \text{NonObject}) & = 1 - \beta \end{cases} \quad (3.20)$$

The fusion of multiple sensors is realized by discounting objects from other peers with a constant, predicting them using timely discount

$$m(t|t-1) = e^{-\Delta t} m(t-1|t-1) \quad (3.21)$$

and correcting based on whether tracks were associated or not

$$\begin{cases} m = j_m \circledast j_m & \text{if associated} \\ m = \lambda^i m & \text{if track } i_o \text{ is not associated} \\ m = j_m & \text{if observation } j_o \text{ is not associated} \end{cases} \quad (3.22)$$

Here, the cautious rule of combination is used when the system is cooperative and the conjunctive otherwise.

3.4 Perception Evaluation

Once a perception system has been built, it needs to be evaluated. A good perception system have low missed-detection and false-detection rates while providing accurate objects localization. To evaluate perception performance, a ground truth must first be acquired, a process reviewed in Section 3.4.1. On-line perception is then compared to that ground truth in order to compute some metrics that measure the perception quality, as reviewed in Section 3.4.2.

3.4.1 Perception Ground Truth

Ground truth are manually (or sometimes semi-manually) labeled sensor data. There exist as many types of ground truth as there are perception tasks. For low-level sensor detection, we can for example cite the COCO dataset (Lin et al.

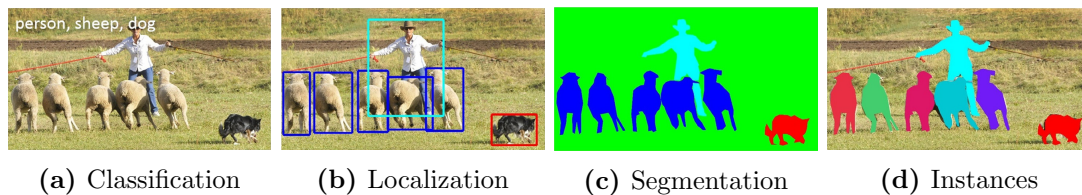


Figure 3.13: Types of information usually found in perception ground truth images. Taken from (Lin et al. 2014).

2014) that provides pixel-level segmentation and classification over 80 classes and 330k images.

In the field of automotive perception, the nuScenes (Caesar et al. 2020) is one of the biggest dataset with fifteen hours of labeled 3D bounding boxes. Before that, the Kitti vision benchmark (Geiger et al. 2012) was one of the most widely used dataset. It provides multiple forms of ground truths over various driving situations, including 2D and 3D bounding boxes, pixel-wise segmentation and recently started including track ground truths. On a related matter, Waymo provides a data set on motion prediction which is based on perception in (Ettinger et al. 2021) where they provide tracked 3D bounding boxes.

3.4.2 Evaluation Metrics

The generic tool to evaluate a binary classifier is to consider a world composed of two classes: positive and negative (e.g. sick or not sick, object or not object). A classifier is tasked with separating data points in this world in the right class, as illustrated in Figure 3.14.

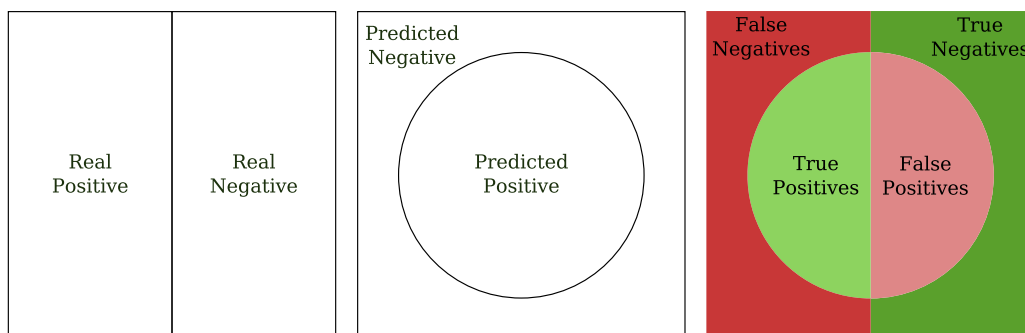


Figure 3.14: Illustration of terms associated to the evaluation of binary classifiers.






Table 3.1: Terms associated to the evaluation of binary classifiers.

	Real positive	Real negative
Predicted positive	True Positive (TP)	False Positive (FP)
Predicted negative	False Negative (FN)	True Negative (TN)

The classification is compared to a ground truth, yielding the four quantities summarized in Table 3.1 depending on whether the classification is correct or

not. The number of correct predictions is compared to the number of incorrect predictions using several ratios summarized in Table 3.2.

Table 3.2: Ratios used in binary evaluation and their meanings.

TP Rate Recall	TN Rate	FP Rate	FN Rate	Precision
$\frac{TP}{TP + FN}$	$\frac{TN}{TN + FP}$	$\frac{FP}{FP + TN}$	$\frac{FN}{FN + TP}$	$\frac{TP}{TP + FP}$
				

However, there is generally a trade-off between the number of False Positives (FPs) and False Negatives (FNs) with binary classifiers. They can be more or less cautious in their classification by setting the entry bar higher or lower. This is generally controlled by a threshold on some classification probability. To represent this ambiguity, tools such as the Receiver Operating Characteristics (ROC) curve have been developed. This curve, exemplified in Figure 3.15, is constructed by computing the FPR and TPR for different thresholds. To summarize the model in a single value, the ROC area under the curve can be used. Following the same idea, another useful tool is the Precision-Recall (PR) curve, where the precision and recall are computed for various thresholds, as illustrated in Figure 3.15. These curves can be summarized with scalars such as the $f1$ -score

$$F1 = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \quad (3.23)$$

There are other types of metrics specific to perception tasks. We can for example cite the Intersection over Union (IoU), mainly used in the field of machine vision to compare detected bounding boxes. It is defined for two polygons A and B as

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.24)$$

Other metrics can be used to measure the difference between two images, most of which are summarized in (Ashraf et al. 2017).

In general, the metric error of matched objects is computed in the working frame (i.e. 2D or 3D position) or even in the complete state vectors (e.g: pose, velocities). In these cases, the classical method is to compute the Normalized

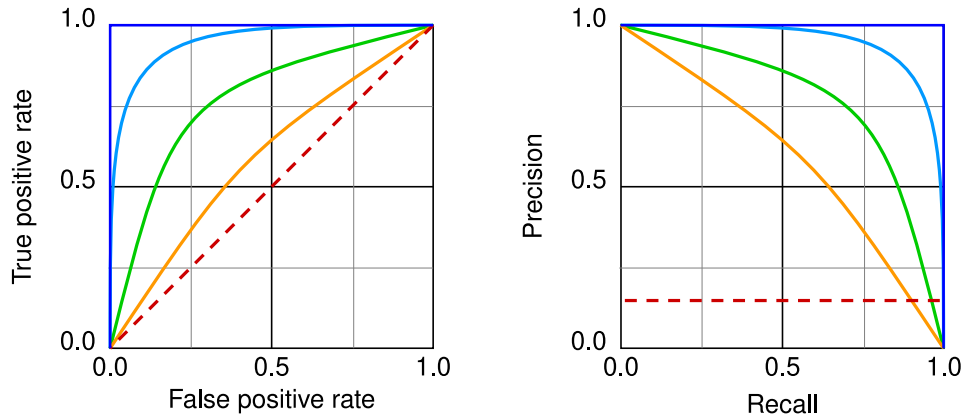


Figure 3.15: Comparison of four models using ROC and PR curves, with the best model being dark blue and worst being red.

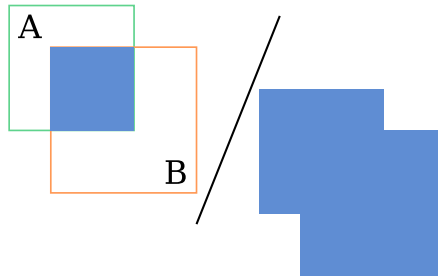


Figure 3.16: Example of IoU.

Estimation Error Squared (NEES) (another name for the Mahalanobis distance of Equation (3.6)) or the Root Mean Square Error (RMSE) for all times:

$$RMSE(t) = \sqrt{\frac{1}{|TP|} \sum_{(i,j) \in \psi(t)} \|\mathcal{G}_i(t) - \mathcal{O}_j(t)\|^2} \quad (3.25)$$

with \mathcal{G} the set of ground truth objects, \mathcal{O} the set of perceived objects and ψ the set of index couples matching \mathcal{G} and \mathcal{O} . This is for example done in (Ambrosin, Alvarez, et al. 2019) by computing an average over all the matched objects.

There are also metrics quantifying the quality of tracks, such as trajectory-based metrics introduced in (B. Wu et al. 2006) or the Multi-Object Tracking Accuracy (MOTA) and Multi-Object Tracking Precision (MOTP) introduced in (Bernardin et al. 2008). MOTA quantifies the tracking quality in terms of misdetections and track switching ID while MOTP quantifies the average distance between matched pairs of ground truth and tracks:

$$MOTA = 1 - \frac{\sum_t (FP(t) + FN(t) + ID(t))}{\sum_t (TP(t) + FN(t))} \quad (3.26)$$

$$MOTP = \frac{\sum_t \sum_{(i,j) \in \psi} (\|\mathcal{G}_i(t) - \mathcal{O}_j(t)\|)}{\sum_t TP(t)} \quad (3.27)$$

However as noted in (Milan et al. 2013), this kind of metric assumes equal weight between all types of errors (FP , FN , ID , distance). Reflecting that some errors

are worse than others requires manual tuning of weights, which is a complicated task.

To simplify this, the Optimal Sub-Pattern Assignment (OSPA) metric has been introduced in (Schuhmacher et al. 2008) to combine precision and accuracy focused metrics with just two parameters. Here, $|\mathcal{O}|$ is the number of perceived objects, c is the cut-off distance in the association procedure and p is a weighting factor between the precision and accuracy parts:

$$OSPA = \left[\frac{1}{|\mathcal{O}|} \left(\overbrace{c^p \cdot (|\mathcal{O}| - |\mathcal{G}|)}^{\text{Association Error}} + \overbrace{\sum_{\langle i,j \rangle \in \psi} \min(c, d^{\text{maha}}(\mathcal{O}_i, \mathcal{G}_j))^p}^{\text{Metric Error}} \right) \right]^{\frac{1}{p}} \quad (3.28)$$

Several tracking approaches make use of this metric such as (Vasic et al. 2016; Yoon et al. 2022).

OSPA has been refined in (Barrios et al. 2017) with COLA to account for false alarms and missed detection more equally and preventing saturation to c when most objects are too far from one another:

$$COLA = \left[(|\mathcal{G}| - |\mathcal{O}|) + \sum_{\langle i,j \rangle \in \psi} \left(\frac{\min(c, d^{\text{maha}}(\mathcal{O}_i, \mathcal{G}_j))}{c} \right)^p \right]^{\frac{1}{p}} \quad (3.29)$$

As summarized in (Hoss et al. 2022; Yan Song et al. 2022), there is a plethora of tracking evaluation metrics that put more or less emphasis on either association or precision. As a result, the Higher Order Tracking Accuracy (HOTA) have been proposed in (Luiten et al. 2021) to provide a neutral but global metric. It is defined as a measure of trajectory matching penalized by unmatched detections:

$$HOTA = \int_0^1 \sqrt{\frac{\sum_{\langle i,j \rangle \in \psi(\alpha)} \mathcal{A}(i, j)}{|TP_\alpha| + |FN_\alpha| + |FP_\alpha|}} d\alpha \quad (3.30)$$

$$\mathcal{A}(i, j) = \frac{TPA(i, j)}{TPA(i, j) + FNA(i, j) + FPA(i, j)}$$

with

- $TPA(i, j)$ the number of times the path of \mathcal{O}_i matched with the path of \mathcal{G}_j
- $FPA(i, j)$ the number of times the path of \mathcal{O}_i did not match with the path of \mathcal{G}_j
- $FNA(i, j)$ the number of times the path of \mathcal{G}_i did not match with the path of \mathcal{O}_j

as illustrated in Figure 3.17.

Finally, some approaches focus on the impact of perception on the navigation and thus develop navigation-centric metrics, such as (Kim et al. 2015) which adapts

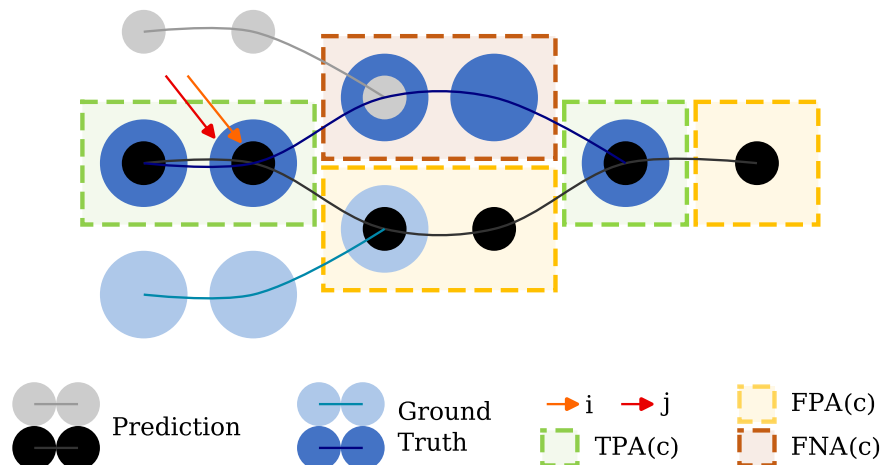


Figure 3.17: Principle of HOTA $\mathcal{A}(c)$ calculation, taken from (Luiten et al. 2021). Here two tracks (grey and black) are associated for each time step with their ground truth (respectively light and dark blue). Starting from the point indicated by red and orange arrows, the number of correct or incorrect associations is accumulated.

the TTC, (Miucic et al. 2018) which compares timing of several V2X applications. In (Phillion et al. 2020), perception quality is measured as a divergence from a reference trajectory and in (Schiegg et al. 2021) awareness about the environment is measured.

Despite its capacity to capture various aspects of perception, HOTA will not be used in the rest of this manuscript as it is too oriented towards tracking and not *perception integrity*. Instead, simpler metrics will be used to evaluate several aspects separately, such as the PR-curve for free space detection, PR-curve for object detection or RMSE for object accuracy.

3.5 Description of the Perception System Used in this Work

In this section, we describe how the experimental data used in the following chapters is retrieved. We first introduce the laboratory experimental platform. Then we present how we implemented car and traffic sign detection and tracking using LiDAR sensors and methods introduced in the literature review. In parallel, a free space detector that provides free space polygons have been implemented following the method of (Yu et al. 2014), illustrated in Section 3.2.4.

3.5.1 Experimental Setup

3.5.1.1 Hardware

The Heudiasyc laboratory owns three Renault Zoe embedded with sensors. They are depicted in Figure 3.18 and are named after their color: zoebue, zoegrey and zoewhite. Two of them, zoegrey and zoewhite are robotized and can be controlled by software, while zoebue is mainly used to record datasets. The

three cars are equipped with various sensors, that can be classified in two main families: proprioceptive (measuring the vehicle’s own state) and exteroceptive (measuring outside elements).



Figure 3.18: The three Renault Zoe used at the Heudiasyc laboratory.

GNSS receiver Global Navigation Satellite System (GNSS) is a technology providing global positions and times to terrestrial vehicles thanks to satellite signals. A receiver is necessary to access GNSS signals or even compute a pose. The one embedded in our vehicles is a Septentrio AsteRx SB Pro Connect (Figure 3.19b). It can receive signals coming from several constellations: GPS controlled by the United States of America (USA), GLONASS by Russia, Galileo by Europe and BeiDou by China. However, GNSS suffers from disturbances such as urban canyons or multi-paths (Zabalegui et al. 2020; N. Zhu et al. 2018) in urban configurations and currently only achieve meter-level positioning. Upcoming GNSS technologies such as Real Time Kinematics (RTK) or Precise Point Positioning (PPP) are expected to bring more accurate positions (Du et al. 2021).



(a) GNSS antenna (b) Septentrio AsteRx SB (c) NovAtel SPAN-CPT

Figure 3.19: GNSS-based sensors used at the Heudiasyc laboratory.

Inertial Measurement Units Inertial Measurement Units (IMUs) measure the accelerations and kinematics of the vehicle, that is to say the v_x , v_y and v_z

velocities, and the roll $\dot{\varphi}$, pitch $\dot{\psi}$ and yaw $\dot{\theta}$ angular velocities. It is generally composed of six sub-sensors, three accelerometers and three gyrometers, each for the three axes x , y and z . Other sources can also be used to retrieve these, such as wheel tops used to measure a linear velocity by counting the number of wheel turns in a given period of time. Our vehicles are equipped with a NovAtel SPAN-CPT (Figure 3.19c) that combines IMUs with GNSS receivers. This combination allows a centimeter-level position when post-processed with PPK, which is why they are used as ground-truth for vehicle localization.

LiDAR To measure their surroundings, the vehicles are equipped with Light Detection And Ranging sensors. This type of sensor works by firing infrared laser beams and measuring the time they takes to bounce back to compute distances. This makes LiDARs excellent at measuring distances, generally providing centimeter-level measurements. In addition LiDARs work in most lighting conditions as they produce their own light. Finally, as the returned signal depends on what it bounced back on, the reflectivity of surrounding surfaces can also be measured.

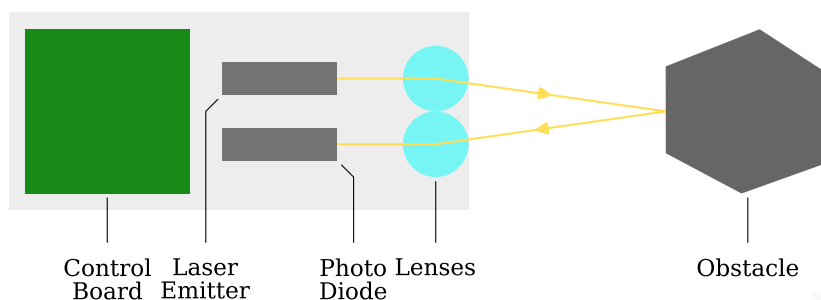


Figure 3.20: Simplified view of a LiDAR firing a laser beam.

Stacking multiple layers of LiDAR and pointing at varying angles makes a multi-layer LiDAR that can measure distances along a vertical slice. Rotating a stack of LiDAR layers around an axis makes a rotating LiDAR that can scan along horizontal slices, called rings. Calculating Cartesian coordinates from distances returned during a full rotation makes a *point-cloud*. LiDAR sensors currently used in the automotive field generally produce 360 degree FoVs up to 150 or 200 m with an horizontal resolution a thousand points across 32 to 64 rings.

The experimental vehicles are equipped with two LiDARS: a 32-ring Velodyne VLP-32C (Figure 3.21a) and a Hesai Pandora (Figure 3.21c) composed of five cameras and a 40-ring Pandar 40P. In addition, a Velodyne 128-ring VLS-128 (Figure 3.21b) lent by Renault has been used as a static road side unit during some experiments. Their respective point-clouds are illustrated in Figure 3.22.

Despite their accuracy, LiDAR sensors suffer several issues that prevent them from wider use. They are vulnerable to fog, snow or rain as laser beams can be reflected or refracted by snowflakes and raindrops. They are sparse, as rings placed a few degrees of one another can end up meters apart when far away from the sensor. They are costly, heavy and hard to integrate on vehicles. Finally,

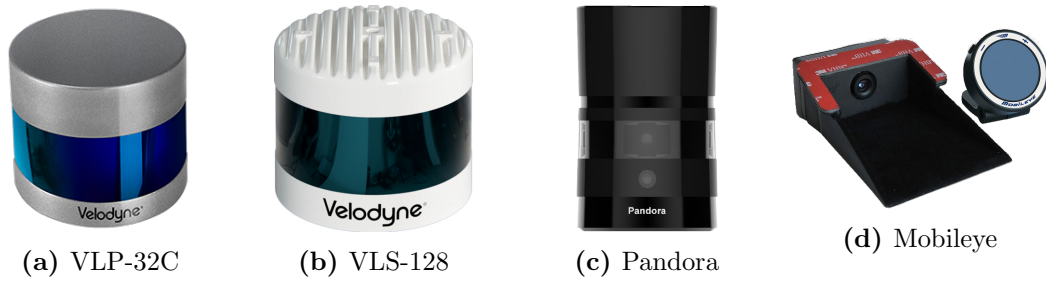


Figure 3.21: LiDAR sensors and cameras used in this research.

they lack reliability, as they are made of precise optics and electronics built upon a rapidly moving part. There are upcoming variations to help with some of these issues, such as the *solid-state* LiDARs that replace rotation with a series of electromagnetic interference (Y. Li et al. 2022), which removes any moving parts and is more compact.

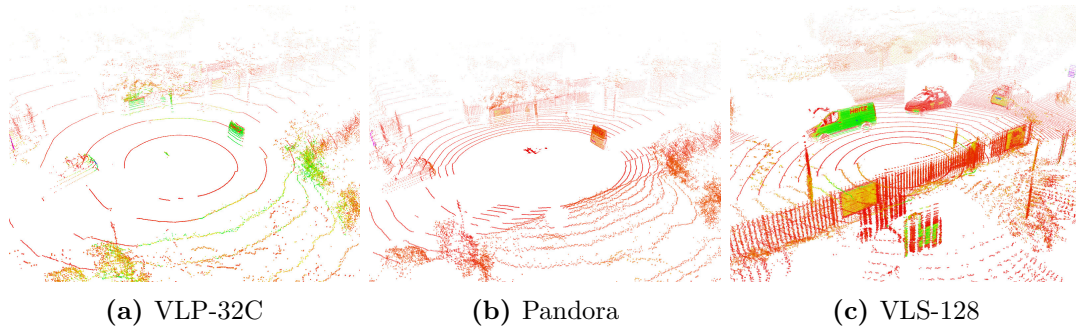


Figure 3.22: Example of point-clouds produced by previously mentioned LiDARs. Colors represent the material reflectivity from purple absorbing to red highly reflective. Notice how the point-cloud is composed of rings stretching outwards from the origin, how dense the Pandora is compared to the VLP32C and the VLS128 to the Pandora.

Camera Cameras are sensors that mimic Human eyes by capturing ambient light reflected back at it. As such, they provide dense images of the environment with color information. Their fields of view are narrower and shorter than LiDARs due to physical limitations of their photoelectric sensors. In addition, they can only provide 2D information as light-rays are projected on the surface of the photoelectric sensor. Finally, they depend on lighting conditions, cannot work at night and can be blinded by sunlight. However, because they are easy to manufacture, cost-effective and easy to integrate in a vehicle, cameras have become the preferred sensor in the automotive field.

The vehicles are equipped with two kinds of cameras. The Hesai Pandora (Figure 3.21c) provide four grey-scale cameras on each side and a front-facing color camera, as in Figure 3.23. The second is a Mobileye Smart Camera (Figure 3.21d) that does not provide images but the result of perception algorithms (e.g: pedestrian, car, sign, free space detection).



Figure 3.23: Images returned by the Hesai Pandora cameras.

New types of cameras are being researched to overcome the mentioned issues. For example, stereo-cameras aim at measuring distances by comparing image returned by two parallel cameras. Infrared cameras (Y. Li et al. 2022) sense infrared radiated by living beings and can thus see them without visible light. Event cameras (Brebion et al. 2022) are another kind of upcoming sensor whose pixels report brightness changes independently and instantaneously, instead of reporting all at once periodically. This makes event cameras capable of wider dynamic ranges and shorter reaction times.

3.5.1.2 Software

In addition to cars and sensors, a series of software libraries have been developed at the Heudiasyc laboratory. They are all based on the Robotic Operating System (ROS) and as such are organized as packages and independent nodes. The main ones that were developed and contributed to in the context of this PhD are:

Table 3.3: ROS packages developed and contributed to in the context of this PhD. Some are private for intellectual property reasons.

<code>lidar_utils</code>	Algorithms for detection in LiDAR point-clouds
<code>perception_utils</code>	Higher-level detection algorithms
<code>grid_map</code>	Occupancy grids
<code>map_server</code>	Management of an HD map
<code>datasets</code>	Recording and management of datasets
<code>multiception</code>	Data fusion and cooperative algorithms

As noted by the variety of modules, a significant effort has been made to make the implementations of this PhD reusable to other lab members. In particular, the detection modules have been used in the Tornado project and several past thesis.

Finally, an HD map is used to provide context and apply environment specific algorithms (i.e: filtering objects on the road). It is composed of precisely located features (i.e: road center and border, road signs, traffic lights, crosswalks) and of their semantic interactions.

3.5.2 Cars and Traffic Signs Detection using LiDAR

In order to study the fusion of perception, one must first be able to perceive. As a proof of concept, simple LiDAR model-based car, road sign and free space detectors have been developed. This covers most common situations found near roads while being sufficiently easy to implement and interpret.

The overall architecture is presented in Figure 3.24. The motion compensation and cylindrical filtering are common processing steps implemented as defined in Section 3.2.1.

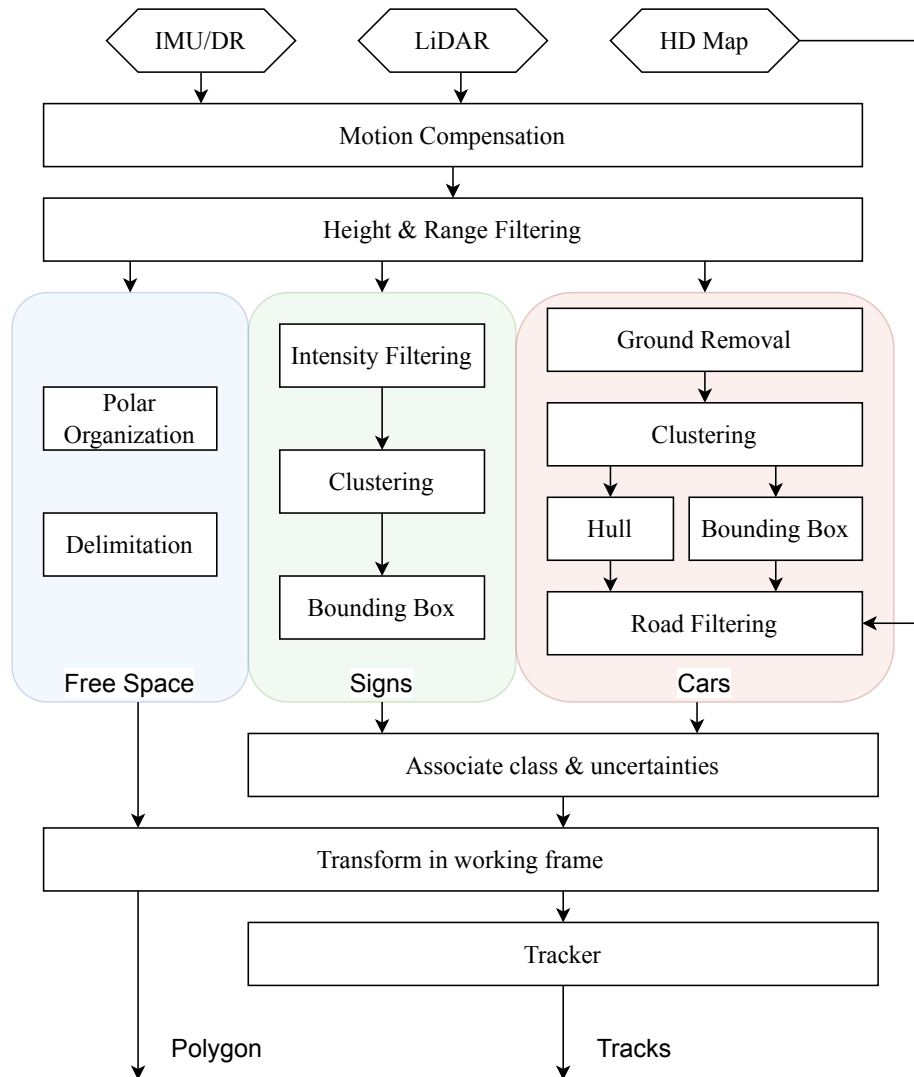


Figure 3.24: LiDAR processing pipeline.

Road signs are covered with a highly retro-reflective material and as illustrated in Figure 3.4 are easily detected using a LiDAR's reflectivity measurement. Points that remain after filtering on intensity are mainly sign-points, though there can remain some false positives (rain, leaves, license plates). These are removed by clustering high-intensity points with DBSCAN and ensuring clusters have a

minimal number of points in them. A bounding box can then be fitted on clusters with Algorithm 1.

For car detection, the idea is that a cluster of points above the road is most certainly a car. To achieve this, the approach of (Jiménez et al. 2021) is used to remove ground points from the point-cloud to clearly separate clusters from one another in the following step. Bounding boxes are then fitted on car clusters, keeping only those above the road. To do this, the convex hulls of clusters are computed and compared to the road polygon extracted from the High Definition (HD) map.

Cars and signs are represented using bounding boxes that describe the center x and y , heading θ and length, width, height l , w , h of an object. In the case of signs, θ is the normal to the sign surface.

A bounding box is defined by:

$$b = \langle x, y, \theta, l, w, h \rangle \quad (3.31)$$

A given sensor j provides two sensor referenced bounding box lists at time t as

$${}^{j,j}\mathbf{B}^{\text{car}}(t) = \{{}^{j,j}b, \dots\} \quad , \quad {}^{j,j}\mathbf{B}^{\text{sign}}(t) = \{{}^{j,j}b, \dots\} \quad (3.32)$$

where the j,j notation means \cdot from the point of view of j and whose frame of reference is also attached to j .

Bounding boxes are then turned into proper objects, which are the basis for tracking and communication afterwards, defined as:

$$o = \langle \mathbf{x}, \mathbf{P}_i, \mathbf{P}_d, \mathcal{I}, c, Z, m^\exists \rangle \quad (3.33)$$

where $\mathbf{x} = [x, y, \theta, v, \dot{\theta}]$ is the object state (position, heading, linear and angular velocities), \mathbf{P}_i and \mathbf{P}_d its associated independent and dependent covariance matrix, $\mathcal{I} = \langle l, w, h \rangle$ is the object size (length, width and height), $c \in \{\text{car}, \text{sign}, \dots\}$ is the object class, Z is an arbitrary set of characteristics (e.g: color, license plate number) and m^\exists represents an estimation of object existence (as described in Section 4.3.4). Unknown values are initialized with a large covariance. For example for cars, this process is

$$\left\{ \begin{array}{l} {}^{j,j}x_o = {}^{j,j}x_b \\ {}^{j,j}y_o = {}^{j,j}y_b \\ {}^{j,j}\theta_o = {}^{j,j}\theta_b \\ {}^{j,j}v_o = 0 \\ {}^{j,j}\dot{\theta}_o = 0 \\ {}^{j,j}m_o^\exists = m_j^\exists \end{array} \right. \quad \left\{ \begin{array}{l} {}^{j,j}c_o = \text{car} \\ {}^{j,j}z_o = \emptyset \\ {}^{j,j}\mathbf{P}_o^{\theta, \theta} = \sigma_j^\theta \sigma_j^\theta \end{array} \right. \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.34)$$

Because a sensor uncertainty makes more sense being expressed in polar coordinate rather than euclidean coordinates¹, covariance matrices are aligned with the

¹i.e: a camera is "imprecise in depth but good at estimating angles", not "precise in front and less on its sides"

line of sight between the sensor and object. For this, the range and angle precision σ^r and σ^{rad} form an intermediary polar covariance matrix that is rotated by the angle between sensor j and object o , $\text{rad}_o^j = \text{atan2}(y_o, x_o)$:

$$\begin{bmatrix} {}^{j,j}\mathbf{P}^{x,x} & {}^{j,j}\mathbf{P}^{x,y} \\ {}^{j,j}\mathbf{P}^{y,x} & {}^{j,j}\mathbf{P}^{y,y} \end{bmatrix}_o = R^{\text{rad}_o^j} \cdot \begin{bmatrix} \sigma^r \sigma^r & 0 \\ 0 & \sigma^{\text{rad}} \sigma^{\text{rad}} \end{bmatrix} \quad (3.35)$$

σ_j^r , σ_j^{rad} , σ_j^θ , ν_j and m_j^\exists are parameters that have to be adapted for a given sensor and its associated detection algorithm j . As a reminder from Section 2.4.5, ν is the dependency factor that distributes observation noise between dependent and independent for later SCI-tracking. Considering the aforementioned processing, detected objects can be considered independent from each other both in time and space and a ν_j close to 0 can be fixed. On the other hand, when detection algorithms introduce correlated errors (e.g: buffering, filtering or prior information), this can be represented with ν_j close to 1.

Car and sign objects are then grouped, transformed in the working frame (accounting for transformation uncertainty, see Section 4.2.2.3) and sent to a tracker.

$${}^j\mathbf{O} = \mathbf{T}_j \cdot ({}^{j,j}\{o, \dots\}_{\text{car}} \cup {}^{j,j}\{o, \dots\}_{\text{sign}}) \quad (3.36)$$

3.5.2.1 Sensor Tracking

Because some sensors only provide already tracked information, such as the Mobileye, other sensor are also tracked in order to have standardized outputs to other modules. Thus, the final step of our LiDAR processing is to track perceived objects. Such sensor-level trackers only interact with a single sensor, and as such do not require alignment or complex existence estimation. For example, given a particular LiDAR j , the prediction, association and update are performed once a new detection list ${}^j\mathbf{O}(t)$ is received. Objects are filtered using a SCIF and independently from one another because the evolution model depends on the object class. Three models are considered, based on (Khan 2019; Sandblom et al. 2014). For cars and other wheeled objects, a constant turn model is used:

$$\mathbf{f}^{\text{CT}}(\mathbf{x}) = \begin{cases} \dot{x} = v \cdot \cos \theta \\ \dot{y} = v \cdot \sin \theta \\ \dot{\theta} = \dot{\theta} \\ \dot{v} = 0 \\ \ddot{\theta} = 0 \end{cases}, \quad \mathbf{Q}_d^{\text{CT}} = \begin{bmatrix} \sigma_x^{\text{CT}^2} & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^{\text{CT}^2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta^{\text{CT}^2} & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^{\text{CT}^2} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\dot{\theta}}^{\text{CT}^2} \end{bmatrix} \quad (3.37)$$

For road signs and other static objects, as static evolution model is used:

$$\mathbf{f}^{\text{ST}}(\mathbf{x}) = \begin{cases} \dot{x} = 0 \\ \dot{y} = 0 \\ \dot{\theta} = 0 \end{cases}, \quad \mathbf{Q}_d^{\text{ST}} = \begin{bmatrix} \sigma_x^{\text{ST}^2} & 0 & 0 \\ 0 & \sigma_y^{\text{ST}^2} & 0 \\ 0 & 0 & \sigma_\theta^{\text{ST}^2} \end{bmatrix} \quad (3.38)$$

Finally, for non-wheeled or static objects (e.g: pedestrians), a constant velocity model is applied:

$$\mathbf{f}^{\text{CV}}(\mathbf{x}) = \begin{cases} \dot{x} = vx \\ \dot{y} = vy \\ \dot{v}_x = 0 \\ \dot{v}_y = 0 \end{cases}, \quad \mathbf{Q}_d^{\text{CV}} = \begin{bmatrix} \sigma_x^{\text{CV}^2} & 0 & 0 & 0 \\ 0 & \sigma_y^{\text{CV}^2} & 0 & 0 \\ 0 & 0 & \sigma_v^{\text{CV}} x^2 & 0 \\ 0 & 0 & 0 & \sigma_v^{\text{CV}} y^2 \end{bmatrix} \quad (3.39)$$

Every σ in Equations (3.37) to (3.39) is left as a tuning parameter. Predicted tracks are then noted ${}^j\mathcal{O}(t|t-1)$.

Association is done using a GNN assignment and the cost function of Equation (3.7), based on Mahalanobis distances and class discrepancies. This assignment function noted \mathcal{A} yields a mapping ψ between predicted tracks and observations:

$$\psi = \mathcal{A}(o(t|t-1), {}^j o(t)) \quad (3.40)$$

Associated tracks get independently updated using Equation (2.40) by replacing $\langle \mathbf{x}, \mathbf{P}_i, \mathbf{P}_d \rangle$ with updated tracks ${}^j\mathcal{O}(t|t-1)$ and $\langle \mathbf{y}, \mathbf{R}_i, \mathbf{R}_d \rangle$ by observations ${}^j\mathcal{O}(t)$, yielding a list of updated tracks ${}^j\mathcal{O}(t|t)$. Non-associated observations create new tracks as detailed in next section.

3.5.2.2 Track Management

Table 3.4: Summary of state and existence tracking parameters

Notation	Domain	Description
σ_*^{CT}	R^{+*}	Dependent evolution noise for constant turn tracks
σ_*^{CV}	R^{+*}	Dependent evolution noise for constant velocity tracks
σ_*^{ST}	R^{+*}	Dependent evolution noise for static tracks
E^{birth}	$[0, 1]$	Birth evidence of existence
E^{forget}	$[0, 1]$	Delete tracks whose information is below this
$t_{1/2}^{\exists}$	R^{+*}	Time to losing half a track existence information
E^{updt}	$[0, 1]$	Increase in existence when associated

In parallel to state filtering, tracks have to be created and deleted. There are two levels of management complexity in this manuscript. The first one, sensor-level is detailed in the following, while more advanced fusion-level is detailed in Section 4.3.4. Existence is managed with belief functions m_o^{\exists} associated with each track o . It is defined on $\Omega^{\exists} = \{E, \cancel{E}\}$ to express evidence supporting that the object is relevant and really exists E or is certainly clutter \cancel{E} .

Similar to (Aeberhard et al. 2011) and as shown on Figure 3.12, tracks are created from un-associated observations with a birth existence $m_{\text{birth}}^{\exists}$:

$$m_{\text{birth}}^{\exists} = \begin{bmatrix} \emptyset & \{\exists\} & \{\cancel{A}\} & \{\exists, \cancel{A}\} \\ 0 & E^{\text{birth}} & 0 & 1 - E^{\text{birth}} \end{bmatrix} \quad (3.41)$$

where E^{birth} is a scalar parameter controlling how likely new objects are.

They are then deleted when not enough evidence of their existence is left:

$$m_o^{\exists}(\Omega^{\exists}) > E^{\text{forget}} \quad (3.42)$$

where E^{forget} is a scalar parameter controlling how soon to delete unlikely tracks.

When tracks are predicted, their existence is decayed according to Equation (2.17) to account for the timeliness of evidence about existence.

$$m_o^{\exists}(t|t-1) = {}_{\Lambda(\Delta t, t_{1/2}^{\exists})} m_o^{\exists}(t-1|t-1) \quad (3.43)$$

where $t_{1/2}^{\exists}$ is a scalar duration parameter controlling how quickly existence decreases.

Finally, existence is increased when tracks are associated. For this, the predicted existence is combined with a constant existence increase

$$m_o^{\exists}(t|t) = m_o^{\exists}(t|t-1) \oplus \left[\begin{array}{cccc} \emptyset & \{\exists\} & \{\exists\} & \{\exists, \exists\} \\ 0 & E^{\text{updt}} & 0 & 1 - E^{\text{updt}} \end{array} \right] \quad (3.44)$$

where E^{updt} is a scalar parameter controlling the existence increase.

Afterwards, tracks from all embedded sensors are fused at the ego-level, as will be explained in Chapter 4. However, we first focus on the evaluation of tracks at the sensor-level.

3.6 Evaluation

In this section, we apply the approaches detailed in Section 3.4 to evaluate both a Mobileye smart camera and our LiDAR detector on cars and road-signs detections.

3.6.1 Evaluation of Sign Detection

For road sign detection, a car equipped with both sensors has been driven around the city of Compiègne for 2 km. The path taken while recording, ground truth road signs and accumulated detections are depicted in Figure 3.25.

The ground truth for signs has been extracted from an HD map. They are associated with sign detections using a GNN and a class dependent cost Equation (3.7). A gate of 4 m has been chosen for the LiDAR and 20 m for the Mobileye. Their matching error is plotted in Figure 3.26 aligned with the vehicle.

In Figure 3.26a, one can see that the detector is not significantly biased in both directions with a mean error of 0.01 and -0.02 m and has a good accuracy with a RMSE of 0.36 m for a variance of 0.09 m. On the other hand, in Figure 3.26b one can see that the Mobileye suffers from a significant bias towards the vehicle of -1.13 and 0.39 m. In addition, while the accuracy is correct in the lateral

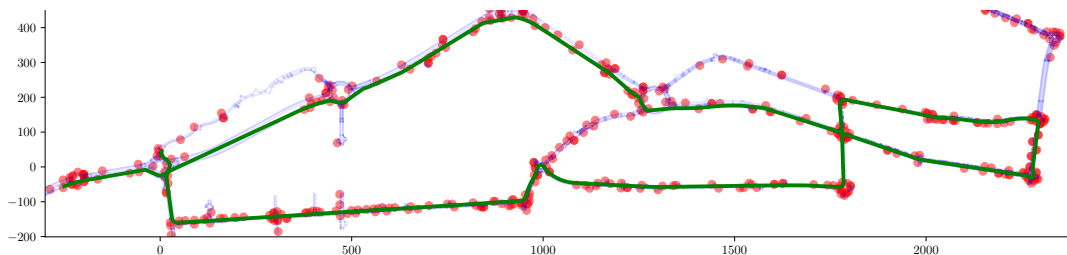


Figure 3.25: Trajectory and data used in sign detection. In blue are the road borders described in an HD map, in red are the ground-truth road-signs and in green the car trajectory.

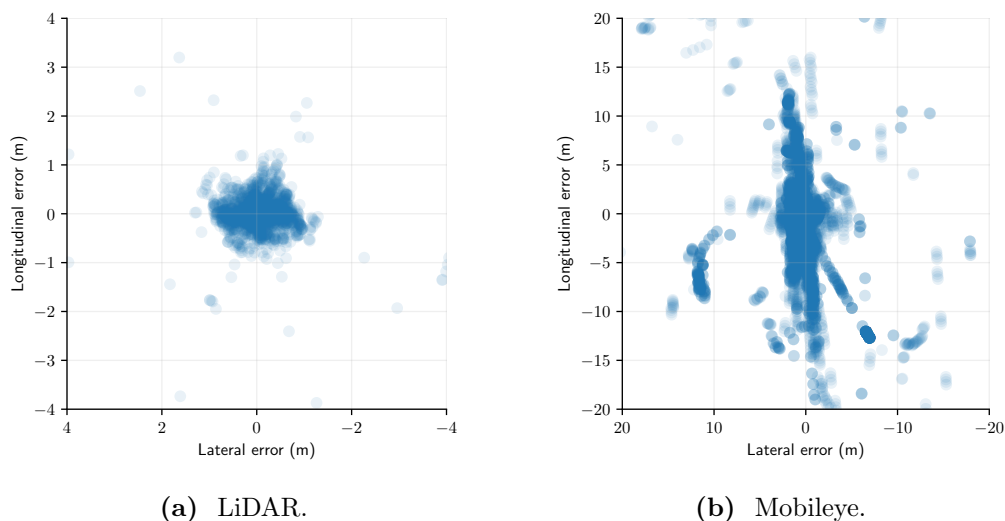


Figure 3.26: Ego-aligned association error between LiDAR/Mobileye road-sign detection and ground-truth.

direction (2.37 m of lateral RMSE) it is particularly inaccurate in the longitudinal direction (5.99 m of longitudinal RMSE). This provides a global accuracy of 3.60 m for a variance of 13.13 m (see Table 3.5), certainly due to projection errors, which makes the Mobileye too inaccurate to be considered for some tasks.

However, there are discrepancies between road-signs stored in the HD map and reality, because some have been moved, added or removed since mapping. This is not an issue for evaluating the accuracy, as a manual verification for obvious false positives has been realized. However this means that detection rates cannot be properly evaluated. Additionally, our Mobileye and LiDAR detectors do not provide a confidence score that could be used to construct a precision-recall curve.

Table 3.5: Evaluation of sign detection for a total of 560 signs on the trajectory

	Mean x error (m)	Mean y error (m)	RMSE (m)	RMSE variance (m^2)	TPR (%)
Mobileye	-1.13	-0.39	3.60	13.13	24
LiDAR	0.01	-0.02	0.36	0.09	33

3.6.2 Evaluation of Car Detection

For car detection, a car equipped with both sensors has been driven around a roundabout for several turns. Two other cars equipped with a NovAtel SPAN-CPT were used as ground-truth.

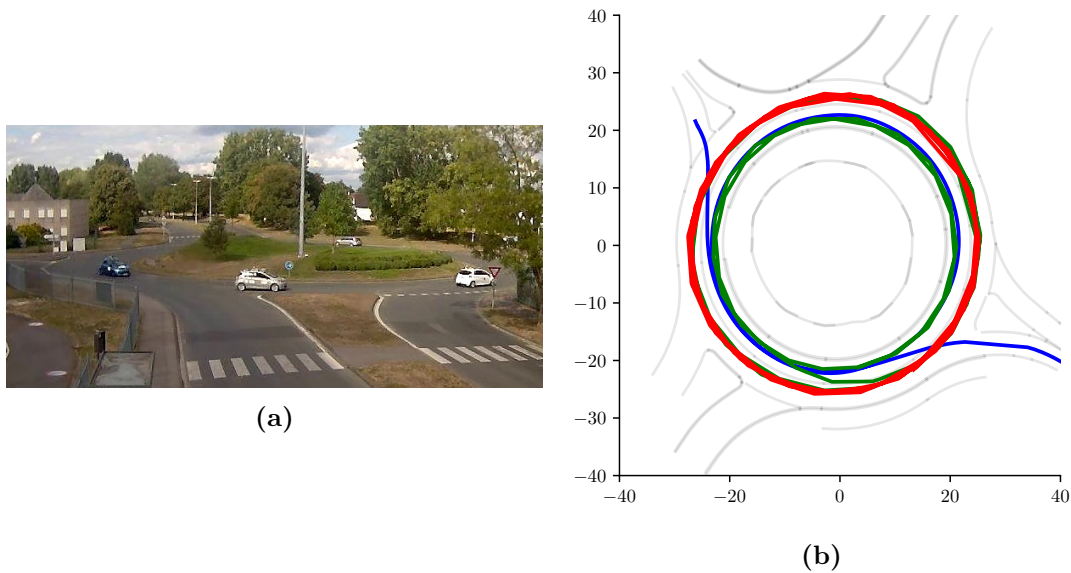


Figure 3.27: Scene and trajectories used in car detection. In grey are the road borders described in the HD map. The three vehicle trajectories are plotted in blue, green and red.

Detected and ground-truth signs are associated using a GNN and a class dependent cost Equation (3.7). Their matching error is plotted in Figure 3.28 aligned with the vehicle.

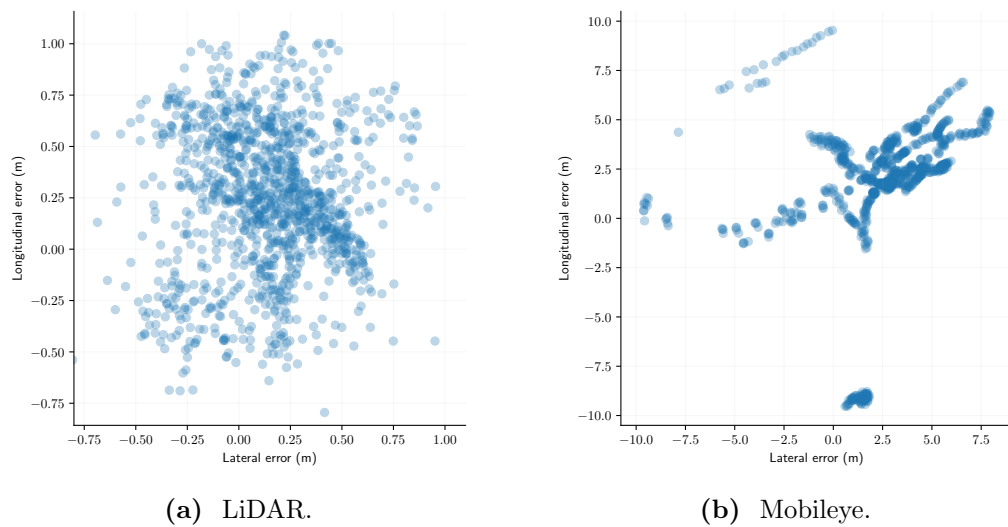


Figure 3.28: Ego-aligned association error between LiDAR/Mobileye car detection and ground-truth.

In Figure 3.28a and Table 3.6, one can see that the LiDAR detector is not significantly biased with a mean error of 0.15 and 0.06 m and has a good accuracy with a RMSE of 0.40 m for a variance of 0.10 m. On the other hand in Figure 3.28b, one can see that the Mobileye suffers from a significant bias away from the vehicle of 1.61 and 2.36 m. In addition, it has a global accuracy of 4.95 m for a variance of 6.12 m, certainly due to projection errors, which makes the Mobileye too inaccurate to be seriously considered for this task. Once again, a complete ground-truth is not available as other vehicles were present in the scene. Thus false positives cannot be evaluated and precision-recall curves cannot be constructed. Results are provided in Table 3.6.

Table 3.6: Evaluation of car detection

	Mean x error (m)	Mean y error (m)	RMSE (m)	RMSE variance (m^2)	TPR (%)
Mobileye	1.68	2.43	4.80	5.56	0.42
LiDAR	0.24	0.13	0.48	0.14	0.72

Finally, the estimated size of the detected cars has an impact on their position and it is clearly a weak point of our LiDAR car detector. In Figure 3.29, the size error of detected objects are compared on both sensors. One can see that the Mobileye has predefined values that tends to be underestimated, with a RMSE of 0.88 m for a variance of 0.09 m. The same underestimation appears on the LiDAR detections, with a RMSE of 0.75 m for a variance of 0.11 m. In particular a strong bias in both length and width of 0.5 m can be observed. This can mainly be explained by an effect introduced in Figure 3.7. When vehicles are only partially seen, detected bounding boxes only cover part of the object, resulting in underestimated sized. The analyzed sequence being a roundabout, vehicles always only see one corner of their neighbors well, resulting in this bias.

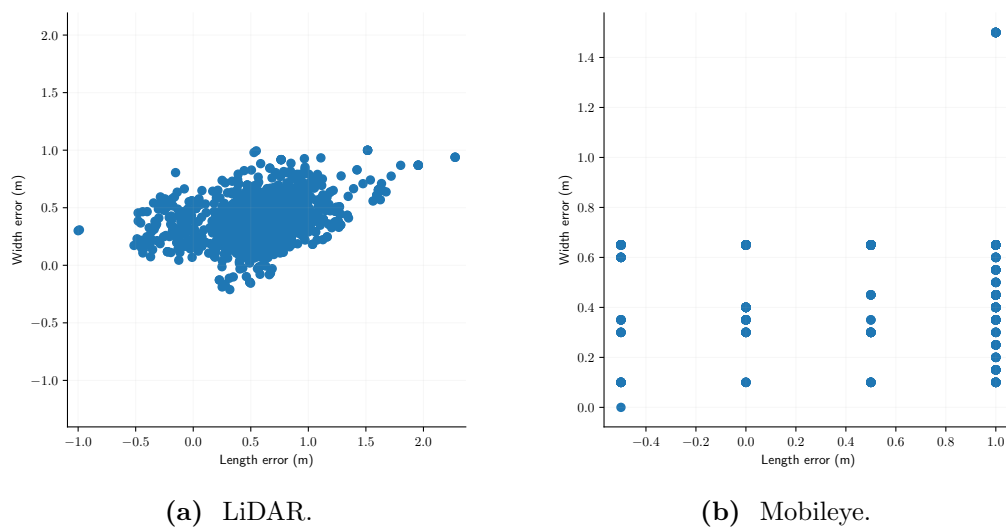


Figure 3.29: Size error between LiDAR/Mobileye car detection and ground-truth.

3.7 Conclusion

In this chapter, we have introduced methods to extract objects and free-space from sensor data. We have presented our methods and algorithms to extract cars, road-signs and free-space from LiDAR point clouds. Then, we have described how object detections are tracked using a classical strategy for the track management in order to provide predictable tracks to the downstream fusion module in charge of the cooperative perception.

An experimental evaluation has illustrated that LiDAR tracks are metrically accurate and consistent, while Mobileye tracks are not. As such, the Mobileye will be ignored in the rest of this manuscript. LiDAR object detection is functional but can be improved. In particular, incorrect car size estimation can cause miss-associations with detections coming from other points of view which can be an issue for cooperative perception. Using size priors, corner-based tracking or machine-learning-based detection could improve this aspect. Secondly, sensor-level tracking might profit from more advanced algorithms such as Multiple Hypothesis Tracking (MHT) to reduce association errors or Interacting Multiple Model (IMM) to adapt the evolution model based object classes. Track management could also be adapted to object classes (e.g. static object can remain longer than dynamic). Finally, the evaluation detection rates for objects and free-space has not been done in this chapter. It will be addressed in the next chapter using particular methodologies that will be introduced.

Chapter 4

Cooperative Perception in a Trustworthy Network

Contents

4.1	Introduction	83
4.2	Review of Cooperative Perception	84
4.3	Fusion of Multiple Points of View	95
4.4	Evaluation of Cooperative Perception	108
4.5	Conclusion	119

4.1 Introduction

A single sensor is generally insufficient to cover a whole scene and leveraging the complementarities of multiple sensors is often the solution. In the previous chapter, sensor-centric perception has been presented. In the present chapter, the perception of multiple sensors are fused. By modeling cooperative peers as deported sensors, we propose an architecture capable of fusing standalone or cooperative perception in a generic way.

In this chapter, the cooperative perception problem is first presented with a literature review in Section 4.2. We then propose a generic architecture to fuse the perception of multiple points of view in Section 4.3. It is based on an object similarity metric, an improved object existence estimator and *evidential detectability*. Detectability is used to represent where sensors or peers are capable of detecting objects. Using an evidential formulation, free space can jointly be represented, providing a convenient tool to track objects across fields of view. Finally, experimental results obtained with three vehicles are presented and analyzed in Section 4.4.

4.2 Review of Cooperative Perception

Thanks to advances in communication technologies, machines can exchange wirelessly further and faster than ever before. This idea is being investigated in several fields of robotics such as localization, control or the topic at hand: perception. By exchanging perception vehicles can see further and behind obstacles, greatly improving their knowledge of the environment. This last aspect is particularly useful at the decision level.

The following is a review of how perception sharing is realized in the literature.

4.2.1 Communication for Intelligent Transportation Systems

In order to communicate, computers require a common language in the form of standards. There are numerous standards, each with advantages and limitations. They are reviewed in this first section.

4.2.1.1 Medium

For the communication medium, which consists in radio waves and low-level networking, there are two standards. The oldest one, called Dedicated Short Range Communication (DSRC), is based on a type of WiFi adapted to outdoor communications (IEEE 802.11p) and is mostly supported by north-American and European regulation agencies. Its newer competitor, called Cellular V2X (C-V2X), is based on the upcoming 5G cellular network, and is mostly supported in the Asian market. Both promise millisecond-level latency and up to a kilometer of range (Keysight 2018), though C-V2X and its multi-frequency is less likely to drop packets in dense areas (Nguyen et al. 2017; Ko-PER Project 2014). C-V2X also has the advantage of using existing equipment for both peer-to-peer communication and internet access, whereas DSRC requires dedicated On-Board Units (OBUs) and Road Side Units (RSUs).

C-V2X thus seems more promising than DSRC but was not available at the time of carrying out experiments. As it does not change what data is exchanged and how it is used, the rest of this manuscript will assume use of DSRC.

4.2.1.2 Messages

What is exchanged over the medium heavily depends on the use case. We will focus on Intelligent Transportation Systems (ITS) oriented messages. There are two main families of message definition. The north American side, backed by the Society of American Engineer (SAE), developed the SAE J2735 (SAE-J2735 2022) specification, that defines the following messages:

- Basic Safety Message (BSM): information about the sending vehicle such as pose, speed, size or light states ;

- MAP/ (SPaT) messages: local mapping and timings for traffic lights, stop or right-of-way signs.

The European side, backed by the European Telecommunications Standards Institute (ETSI) developed the 102 894-2, 302 637-2 and 302 637-3 specifications (see ETSI 2019), that defines various messages, summarized in Figure 4.1:

- The Cooperative Awareness Message (CAM), which is similar to the BSM as it provides information about the ego vehicle pose and intentions ;
- The Decentralized Environment Notification Message (DENM), which provides information about local events (accident, roadworks, etc), their cause and who is concerned ;
- The Collective Perception Message (CPM), which provides information about on-board sensors and perceived objects (pose, size, class) ;
- The MAP/SPaT, which is an adaptation of the SAE MAP/SPaT messages.

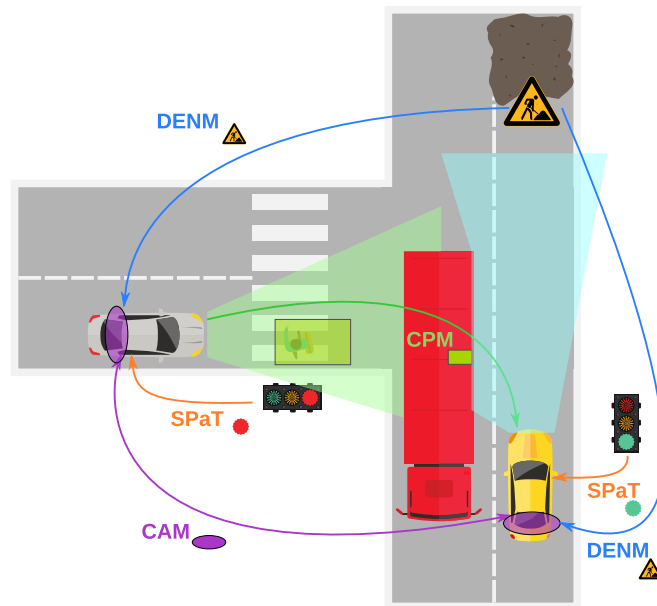


Figure 4.1: Summary of ETSI messages between two communicating cars.

In the rest of this manuscript, the CPM will be considered as the standard supporting our research. It is widely studied (Caillot et al. 2022) and fulfills most needs of CP. As such, the content of CPMs is here detailed to introduce what is available to CP algorithms. As summarized in Figure 4.2, the CPM specifies several characteristics about the sender such as its identifier, pose, size, type (vehicle, pedestrian, infrastructure) and velocities. Then, on-board sensors are listed through their type (e.g: mono/stereo camera, LiDAR, RADAR) and Field of View (FoV) (range, opening, polygon). Two aspects of perception are then exposed: measured free space as a polygon and a list of perceived objects. As

Cooperative Perception Message																	
Header			Emitter Data			Perception Data								Security			
Message Type	Message ID & Time	Reference Position	Station Type & State		Sensors			Objects [1-128]					FS [1-128]			Signature	Certificate
			MAP Message	Segment ID	Sensor ID & Type	Relative Position	Perception Area	Object ID	Confidence	Meass. Time	State			Dimensions	Sensor ID		
									Position	Angles	Velocities	Accelerations					

Figure 4.2: Summary of fields in the Collective Perception Message. Adapted from (Ansari et al. 2021).

summarized in Figure 4.2, objects are defined by an identifier, time of measurement, confidence, classification, state and size.

To limit the size of exchanged messages, fields are limited in precision and most are optional. According to (Allig and Wanielik 2019a; Thandavarayan et al. 2019; Zhou et al. 2022), the size CPMs vary from 100 to 1000 bytes depending on the complexity of exchanged information.

4.2.1.3 Contents of Cooperative Perception Messages

How to fill the aforementioned fields is still relatively open. (Günther et al. 2016) argues that non-tracked, raw sensor data is preferable but the rest of the community seems to use tracked data (Allig and Wanielik 2019a; Caillot et al. 2022). There are also more and more interest in active cooperative perception, where not all objects are sent all the time. For example, in (Thandavarayan et al. 2019) rules based on object type, position change Δx , velocity change Δv and last time sent Δt are defined to reduce the number of objects sent (see Figure 4.3). This system demonstrated reduced communications without significant loss in awareness. In (Zhou et al. 2022), redundant objects are not sent by computing

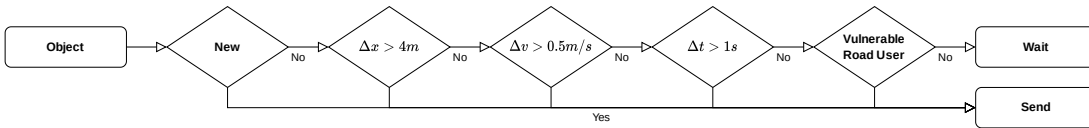


Figure 4.3: Generation rules defined by (Thandavarayan et al. 2019) for sending an object or not.

the awareness increase brought by each object and by deciding whether to send an object based on this. More works are reviewed in (Delooz et al. 2022).

Finally, CPMs are meant to be broadcasted to surrounding peers without additional hops or geo-casting, contrary to other messages such as the DENM.

4.2.1.4 Security in Vehicular Networks

As communications take place in an open network, computers and the vehicle they control are open to attacks. As such, security measures are mandatory in order to limit the attack surface. Contrary to most cases, encryption is not the solution here, as messages are meant to be readable by anyone. However, they should not be written by anyone, and for this a Public Key Infrastructure (PKI) can be used. The principle of a PKI, summarized in Figure 4.4 is to supply private/public key pairs such that any receiving peer can cryptographically verify the identity of a sender (Chowdhury et al. 2017; Madl 2021). However, this architecture makes tracking a peer at a global scale very easy, which is why sub-keys (or pseudonyms) can be used to locally change identity.

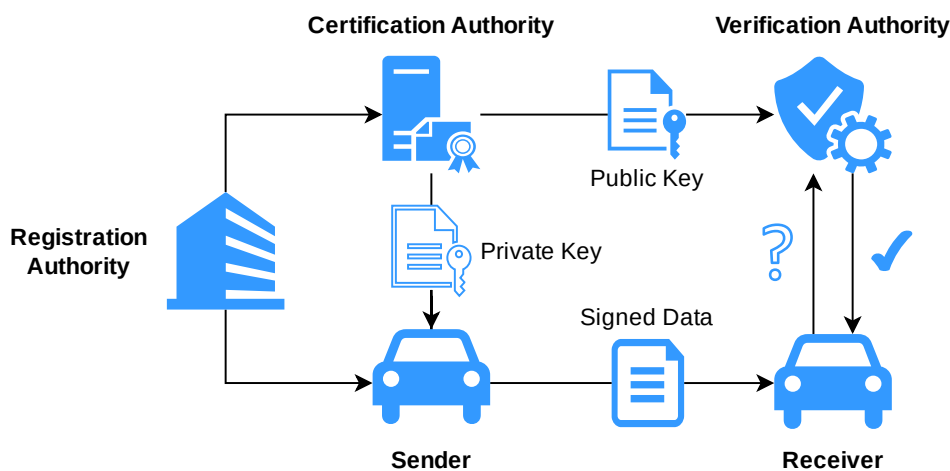


Figure 4.4: Example of Public Key Infrastructure (PKI). Adapted from (Danquah et al. 2020).

In this chapter, we will consider these measures sufficient to ensure that we are in a trusted network, though as explored in the next chapter, other methods should be used to verify this hypothesis. Indeed, it is still possible for determined attackers to find ways around this authentication or simply for peers to be faithfully mistaken.

4.2.2 Cooperative Track-To-Track Fusion

4.2.2.1 Cooperative Fusion Architectures

Now that vehicles can communicate with each other, let us review how exchanged information is typically fused. As summarized in works as early as (Herpel et al. 2008), fusion can happen at several points in the sensor processing stack:

Exchanging and fusing early information amounts to exchanging raw sensor data and realizing normal sensor detection methods in the extended data. For example, in (H. Li et al. 2011) and (Kim et al. 2015) images from cameras are exchanged between following vehicles and used to approximate *see-through* with

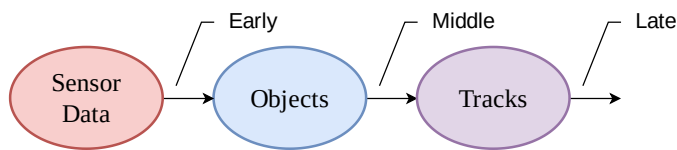


Figure 4.5: Illustration of early to late fusion in sensor processing.

leading vehicles. In (Sridhar et al. 2019), images are exchanged to infer the depth of objects thanks to the difference of point of view and common features. Similarly, (Q. Chen et al. 2019) exchange point clouds and detect objects in the concatenated point cloud. However, while images can be sent as a video stream thanks to efficient compression algorithms, LiDAR point clouds are more difficult to compress and thus new methods have been derived. For example, in (T.-H. Wang et al. 2020), a neural network is used to find features in point clouds and compress them. Then it is sent and fused by accumulation on the receiver side. However, despite these efforts, raw data is still too heavy to be tractable in dense situations. A lighter alternative is thus to apply detection algorithms and exchange their results. Object detections are lighter while still representing most of the important information contained in raw data. These objects are communicated as instantaneous measurements (i.e: not tracked) bounding boxes according to the CPM standard (Günther et al. 2016). However, because they are raw information, these objects have to be exchanged as often as possible for a receiver to properly track them, which is against generation rules described in Section 4.2.1.3. To reduce communication and tracking-related loads, already-tracked objects can instead be exchanged. This requires less frequent exchanges while maintaining predictability and seems to be the current consensus (Allig and Wanielik 2019a; Gabb et al. 2019; Rauch et al. 2012). According to (Allig and Wanielik 2019b), exchanging variance and acceleration information drastically improves the tracking accuracy on the receiver side, but exchanging covariance (non-diagonal variance) have negligible impact.

Another common distinction depends on whether approaches are centralized or decentralized. In (Gabb et al. 2019) for example, infrastructure sensors are fused on a centralized server before being sent to vehicles. Most tracking approaches separate what they locally and cooperatively perceived as illustrated in Figure 4.6 for (Allig and Wanielik 2019a; Günther et al. 2016; Rauch et al. 2012).

4.2.2.2 Cooperative State Filtering

In terms of data fusion, while certain approaches are geometry based, such as (Z. Song et al. 2022), cooperative objects are generally fused by filtering their state vectors. Several filters introduced in Section 2.3.2 have been compared in the literature over the same contexts. For example in (Ambrosin, Alvarez, et al. 2019), a Covariance Intersection Filter (CIF) is used. (Seeliger et al. 2014) compared several variants of the CIF and found that the improved-fast-CI was the best compromise between performance and accuracy. Later, (Gabb et al. 2019) found that the IMF performed too closely to the CIF to justify the requirement for

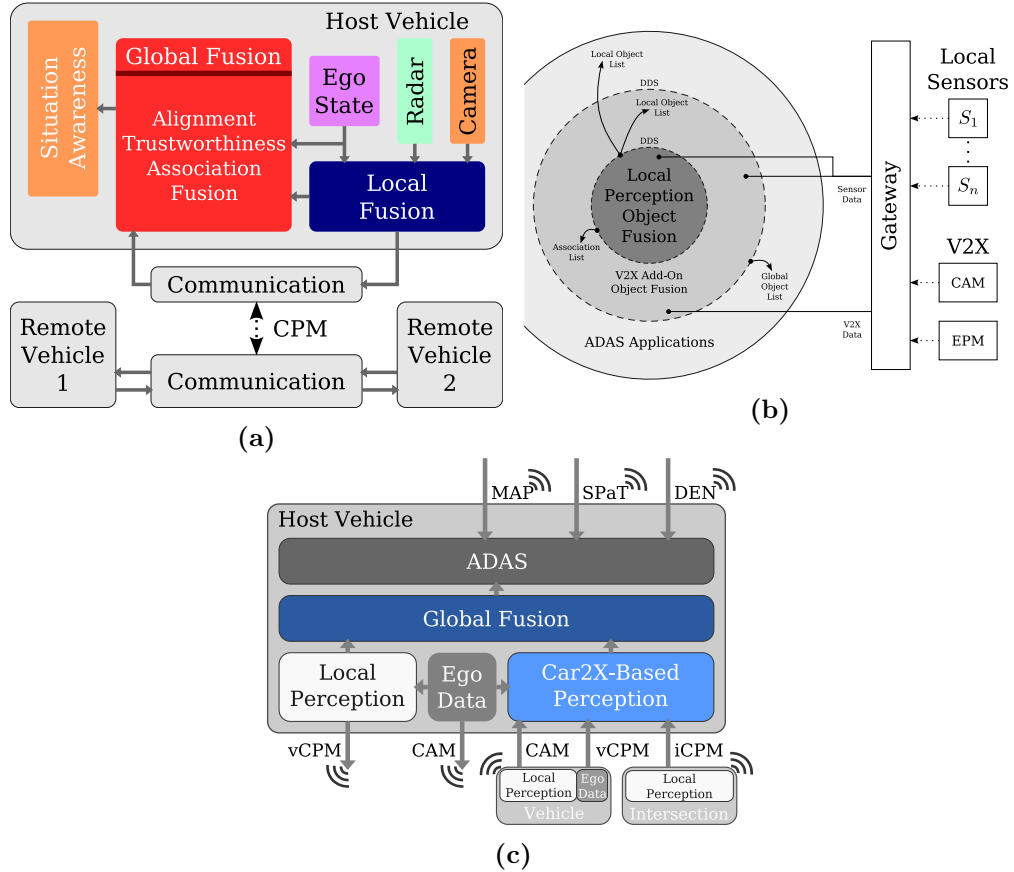


Figure 4.6: Fusion architectures used in a) Rauch et al. 2012, b) (Allig and Wanielik 2019a) and c) (Günther et al. 2016).

additional knowledge about inter-dependence. Finally, there is a significant part of the research community that focuses on the use of the Probability Hypothesis Density (PHD) for this type of application (Nuss et al. 2018; Vasic et al. 2016).

Up to this point, tracks and observations were considered aligned in space and time. However in practice, sensors have different points of view, and work at different rates. The next sections address these differences.

4.2.2.3 Spatial Alignment

An observation, track or any other type of information is always referenced against a certain origin in space, called frame of reference. In the case of autonomous navigation, we often use four standard frames (Allig and Wanielik 2019a):

1. **Earth frame E :** Reference system used for example by a Global Navigation Satellite System (GNSS) generally expressed in the form of geographic coordinates (longitude L , latitude l , altitude a) relative to the center of Earth as specified by WGS84¹. It can also be given in Earth-Centered

¹<https://wiki.gis.com/wiki/index.php/WGS84>

Earth-Fixed (ECEF), a Cartesian reference system centered on the Earth. While global, these systems are not adapted to terrestrial navigation;

2. **Working frame W** : Cartesian frame defined from a plane tangential to the Earth's curvature at a given reference point. The x , y and z axes are respectively defined as East-North-Up (ENU). Reliable only around the reference point but more adapted to terrestrial navigation as it allows 2D projections. This is generally the common frame between multiple vehicles;
3. **Mobile frame M_i** : Cartesian frame attached to vehicle v_i as it moves;
4. **Sensor frame S_j** : Cartesian frame attached to sensor s_j of a vehicle.

Figure 4.7 summaries these four frames and how they are connected to each other. The frame F in which a state \mathbf{x} is referenced is noted ${}^F\mathbf{x}$. F is generally omitted for the sake of readability when \mathbf{x} is referenced to the working frame.

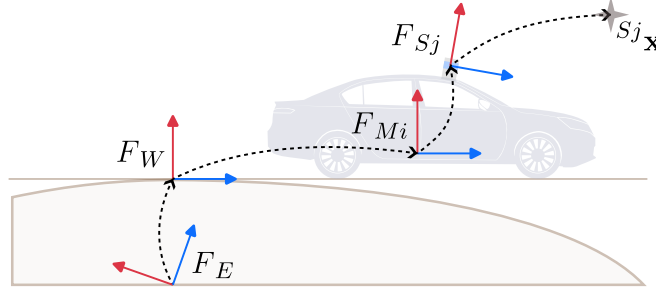


Figure 4.7: Four main frames of reference used in terrestrial navigation. Arrows represent the transformation from one frame to the next. A detected point \mathbf{x} is referenced here in the sensor frame.

The transformation from F_E to F_W can be done as follows. Let us define a reference point p_r with known Cartesian coordinates in ECEF ${}^{ECEF}p_r$ and geographic WGS84 coordinates ${}^{WGS84}p_r = [L_r, l_r, a_r]$. It can then be used to transform an arbitrary point p between ECEF ${}^{ECEF}p$ and ENU ${}^{ENU}p$ with

$${}^{ENU}p = \begin{bmatrix} -\sin(L_r) & \cos(L_r) & 0 \\ -\sin(l_r)\cos(L_r) & -\sin(l_r)\sin(L_r) & \cos(l_r) \\ \cos(l_r)\cos(L_r) & \cos(l_r)\sin(L_r) & \sin(l_r) \end{bmatrix} \times ({}^{ECEF}p - {}^{ECEF}p_r) \quad (4.1)$$

Later transformations (i.e: from F_W to F_M and F_M to F_S) are affine (composition of a θ -rotation around the z axis R^θ and translation t) and as such can be represented using matrix multiplications on homogeneous coordinates.

Example 4.1:

Let A and B two reference frames. In two-dimensions, the homogeneous

transformation ${}^A\mathbf{T}_B$ expressing the frame B in A and is defined as:

$${}^A\mathbf{T}_B = \begin{bmatrix} R^\theta & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos({}^A\theta_B) & -\sin({}^A\theta_B) & {}^Ax_B \\ \sin({}^A\theta_B) & \cos({}^A\theta_B) & {}^Ay_B \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation of one (or many) points is done by a single matrix multiplication:

$${}^A \begin{bmatrix} x_1 & x_2 & \dots \\ y_1 & y_2 & \dots \\ 1 & 1 & \dots \end{bmatrix} = {}^A\mathbf{T}_B \times {}^B \begin{bmatrix} x_1 & x_2 & \dots \\ y_1 & y_2 & \dots \\ 1 & 1 & \dots \end{bmatrix}$$

Applying the inverse transformation can be done by inverting the forward transformation:

$${}^B\mathbf{T}_A = {}^A\mathbf{T}_B^{-1}$$

which can be done efficiently by taking advantage of the matrix construction as:

$${}^A\mathbf{T}_B^{-1} = \begin{bmatrix} R^{-\theta} & -R^{-\theta}t \\ 0 & 1 \end{bmatrix}$$

Finally, thanks to the nature of homogeneous matrices, multiple transformations can be applied at once by multiplying their matrices in advance:

$${}^A\mathbf{T}_Z = {}^A\mathbf{T}_B \times {}^B\mathbf{T}_C \times \dots \times {}^Y\mathbf{T}_Z$$

Angles can also be transformed. In 2D, an addition is sufficient.

$${}^B\theta = {}^A\theta_B + {}^B\theta$$

It is also clear that for velocities, movement should be compensated from one frame to another. To do so the same formalism as with positions can be used. Let $\mathbf{v} = \begin{bmatrix} vx & vy \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\theta) & v \cdot \sin(\theta) \end{bmatrix}$ a velocity vector. The transformation of such vectors from frame B to A is defined as

$${}^A \begin{bmatrix} vx \\ vy \\ 1 \end{bmatrix} = \begin{bmatrix} \cos({}^A\theta_B) & -\sin({}^A\theta_B) & {}^Avx_B \\ \sin({}^A\theta_B) & \cos({}^A\theta_B) & {}^Avy_B \\ 0 & 0 & 1 \end{bmatrix} \times {}^B \begin{bmatrix} vx \\ vy \\ 1 \end{bmatrix} \quad (4.2)$$

Finally, as points and transformations can be uncertain, (Smith et al. 1986) proposed a compound operator that transform covariances while accounting for the transformation uncertainty. Let \mathbf{P} be a covariance matrix defined as

$$\mathbf{P} = \begin{bmatrix} \sigma_{x,x} & \sigma_{x,y} & \sigma_{x,\theta} \\ \sigma_{y,x} & \sigma_{y,y} & \sigma_{y,\theta} \\ \sigma_{\theta,x} & \sigma_{\theta,y} & \sigma_{\theta,\theta} \end{bmatrix}$$

Considering that the transformation errors are independent from each other, the transformation of a covariance matrix ${}^B\mathbf{P}$ expressed in frame B to a frame A is done, knowing the uncertainty matrix ${}^A\mathbf{P}_B$ between the two frames, as:

$$\mathbf{J}_1 = \begin{bmatrix} 1 & 0 & {}^A y_B - {}^B y \\ 0 & 1 & {}^B x - {}^A x_B \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{J}_2 = \begin{bmatrix} \cos({}^A\theta_B) & -\sin({}^A\theta_B) & 0 \\ \sin({}^A\theta_B) & \cos({}^A\theta_B) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

$${}^A\mathbf{P} = \mathbf{J}_1 {}^A\mathbf{P}_B \mathbf{J}_1^T + \mathbf{J}_2 {}^B\mathbf{P} \mathbf{J}_2^T$$

This can for example be used to transform a relative position measurement $\langle \mathbf{y}, \mathbf{R} \rangle$ from a sensor frame into the working frame using the vehicle pose estimation of $\langle \mathbf{x}, \mathbf{P} \rangle$.

4.2.2.4 Temporal Alignment

In real tracking scenarios, sensors have different refresh rates (e.g: 30 Hz for cameras, 10 Hz for LiDARs), which can lead to inconsistencies if not properly managed. If observations arrive in the right order, the tracker naturally provides a mechanism to incorporate them appropriately, by iteratively predicting tracks to observation times and correcting them.

Example 4.2:

Let s_1 and s_2 be two sensors. On a given period of time, s_1 produces five observations and s_2 produces four. Upon reception of the first observation, here from s_1 , a track is initialized. Upon reception of a second observation, this time from s_2 , the track is predicted to the observation time before being updated.

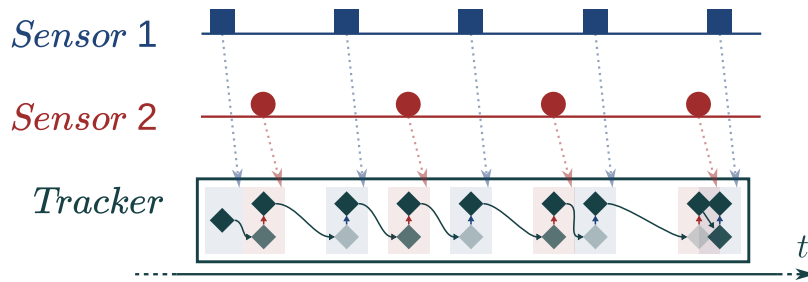


Figure 4.8: Simple asynchronous tracking with two sensors.

4.2.2.5 Out-Of-Sequence Observations

In certain tracking scenarios with several sensors, observations can be received Out-Of-Sequence (OOS) due to important processing time or communication delays.

Example 4.3:

Let s_1 and s_2 two sensors. Both of them have delay but that of s_2 is longer than tracking steps, as illustrated in Figure 4.9.

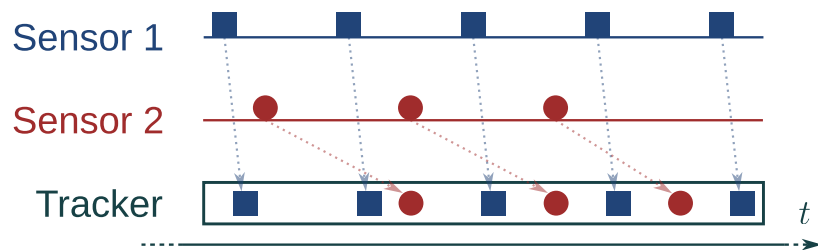


Figure 4.9: Example of out-of-sequence observations with two sensors

This time, it is not handled naturally by trackers and, according to (Muntzinger et al. 2010), several methods exist to handle OOS observations:

1. Ignore: When delays are short enough, the Δt between track and observation time can be ignored without significant impact.
2. Delayed: If the application is not real-time and OOS are known, in-sequence observations can be buffered until the OOS is received. However, this causes lags and increased covariance.
3. Observation forward prediction: When observations are tracks (such as with T2TF), they contain information necessary to be extrapolated by definition. A solution, illustrated in Figure 4.10, can thus be to predict the OOS track up to current time before incorporating it (Allig and Wanielik 2019a). The issue is that prediction is a lossy operation due to evolution models that are imprecise and extend covariance. Using this solution is thus lightweight but sub-optimal.

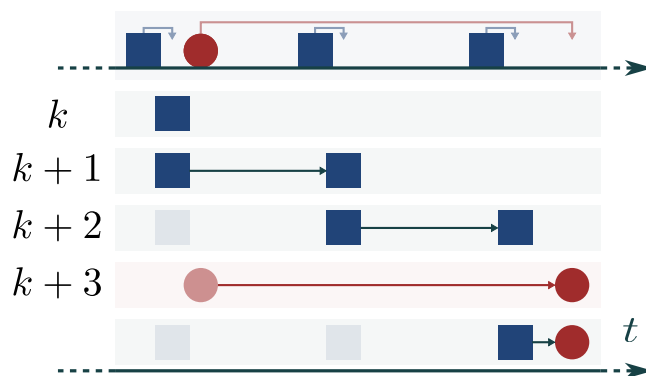


Figure 4.10: OOS integration using observation forward prediction.

4. Reprocess: Observations and tracks are buffered. When a OOS is received, tracks after the emission time are dropped, the OOS is processed then observations received after the OOS are processed sequentially, as in Figure 4.11. This method is exact since no information is lost, but heavy because past tracks and observations must be stored and every tracking step between OOS emission and reception must be redone.

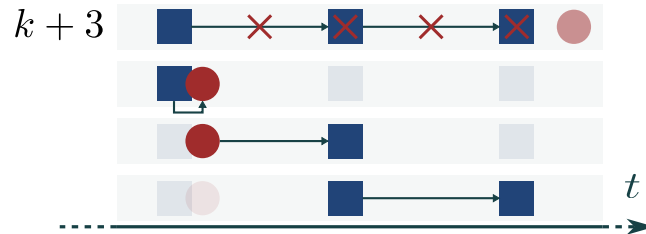


Figure 4.11: OOS integration using reprocess

5. Track backward prediction (retrodiction): The current track is predicted with negative time steps down to OOS time. The OOS is incorporated then the track is predicted back to its original time as in Figure 4.12. There exists an important literature on this aspect because of the strong correlation between backward and forward process noise, with optimal (Nettleton et al. 2001) or sub-optimal (Y. Bar-Shalom 2002) proposals. In any case, the application to non-linear systems over long delays is not expected to yield satisfactory results.

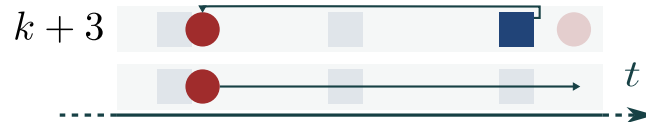


Figure 4.12: OOS integration using retrodiction

6. Forward-Prediction Fusion and Decorrelation (FPFD) from (Rheaume et al. 2008): This method creates a pseudo-observation by predicting the OOS tracks with and without first updating using the OOS observation. By removing the common information between both versions of the track, a decorrelated pseudo-observation containing the information gained by fusing is obtained. It can then be combined with the current track, as depicted in Figure 4.13. It has been shown to be optimal with delays up to one tracking step but sub-optimal above.

4.2.3 Evaluation Methods for Cooperative Perception

As noted in (Hurl et al. 2020), there is a lack of common methodologies and data sets to evaluate cooperative perception systems. Indeed, there does not seem

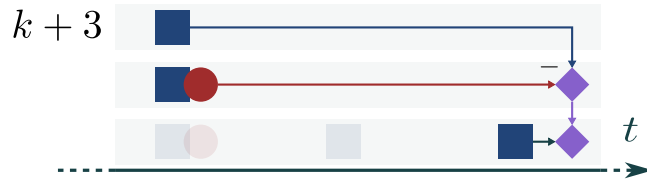


Figure 4.13: OOS integration using FPDF

to exist a real data set covering cooperative perception situations but there are simulated data sets such as OPV2V (Xu et al. 2022) and standalone data sets where cooperation is simulated such as (Q. Chen et al. 2019) with Kitti. There is however the *LUMPI* dataset (Busch et al. 2022) illustrated in Figure 4.14, composed of multiple points of view from static road side units.

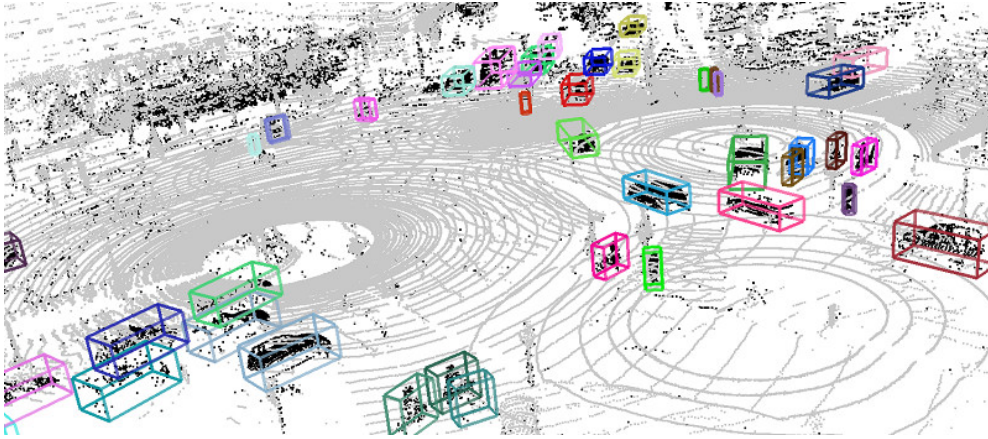


Figure 4.14: Ground truth labels over raw point clouds from (Busch et al. 2022).

To evaluate cooperative perception, classical tools such as those introduced in Section 3.4.2 are generally used (Seeliger et al. 2014). To our knowledge, an evaluation methodology that accounts for the cooperative nature of perception has not yet been proposed.

4.3 Fusion of Multiple Points of View

The problem of fusing cooperative sensors can be handled as a traditional sensor fusion process. By considering peers as remote sensors, a generic architecture can be derived to work with both standalone and cooperative information. In this section, we introduce our architecture. It is based on several novel concepts: an evidential representation of source detectability, an object-similarity metric and an improved object-existence estimator.

4.3.1 Generic Fusion Architecture

As reviewed in Section 4.2.2, a classical approach is to separate locally and cooperatively perceived object lists for security reasons. However, the question of object or track level fusion is still up to debate. In this work, the chosen solution is a track-level fusion because some sensors internally track or filter their raw data (e.g: smart cameras or GNSS receivers). Moreover, tracks are by definition predictable over time and can be communicated at lower rates than raw objects. Additionally, cooperative perception can be computationally intensive as it requires the fusion of many objects perceived by many sensors attached to many cooperative vehicles. A layered approach, where raw information is progressively tracked toward higher and more complete track lists can be used to alleviate this issue, as illustrated in Figure 4.15. The architecture we propose is thus a compromise between computation requirements and data completeness.

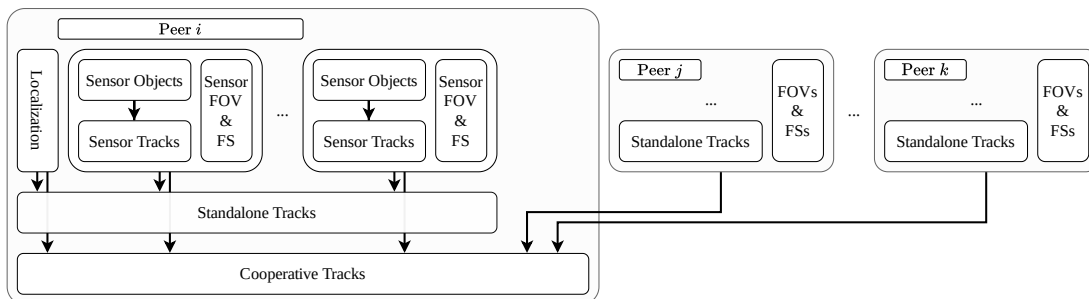


Figure 4.15: Multi-Vehicle (or *peer*) architecture used in this manuscript. Sensors are all tracked independently before being fused in both a standalone and cooperative tracker. Another peer sends its standalone tracks which are fused in the cooperative tracker.

Having sensor trackers allows sensor-specific behaviors and heuristics close to sensors themselves, isolating these specificities from the rest of the system. The standalone and cooperative trackers can thus do generic Track-To-Track Fusion (T2TF) without distinction between remote and local sources. In particular, peers can be considered as *deported* sensors for these trackers and received information can be fused without additional precautions. The term *source* will thus be used in the rest of this manuscript when the distinction between local sensor and remote peer is not relevant.

Sensor trackers have been described in Section 3.5.2.1 and the standalone and cooperative tracker will be described in Section 4.3.4. They are all based on the Split Covariance Intersection Filter (SCIF) that, as has been studied in Section 2.4.4, is able to manage unknown levels of correlation while providing good filtering performance.

4.3.2 Managing the Detectability of Multiple Sources

When multiple points of view are considered, it is useful to account for the area covered by each source. This idea has been introduced in (Aeberhard et al. 2011)

where a persistence probability (illustrated in Figure 4.16a) is derived from the field of view (range and openings) of each sensor. This probability is used to discount an existence-nonexistence belief function while fusing multiple sensors. The idea has then been extended in (Allig, Leinmüller, et al. 2019) to also account for obstructions caused by objects and used to detect discrepancies between overlapping areas. Both approaches can be summarized as applying a 2D probability function that decreases with distance and proximity to a polygon boundaries.

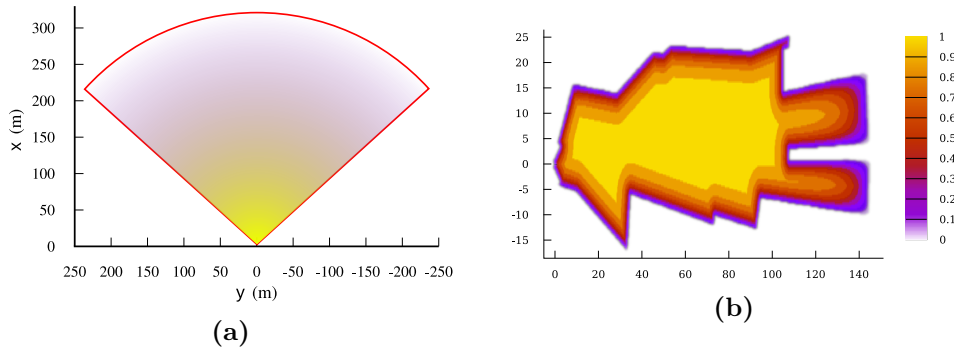


Figure 4.16: Persistence or detectability probabilities based on sensor range and opening (a, from (Aeberhard et al. 2011)) and obstacles (b, from (Allig, Leinmüller, et al. 2019)).

In this section, we introduce the concept of evidential detectability. Using belief functions, the idea of managing a detectability probability is extended to also express impossibility to detect an object, typically using a free-space measurement. After describing how it is computed for a given sensor, the combination of multiple sources is studied. Finally, we present how detectability is used to track objects across fields of view.

4.3.2.1 Definition of Detectability

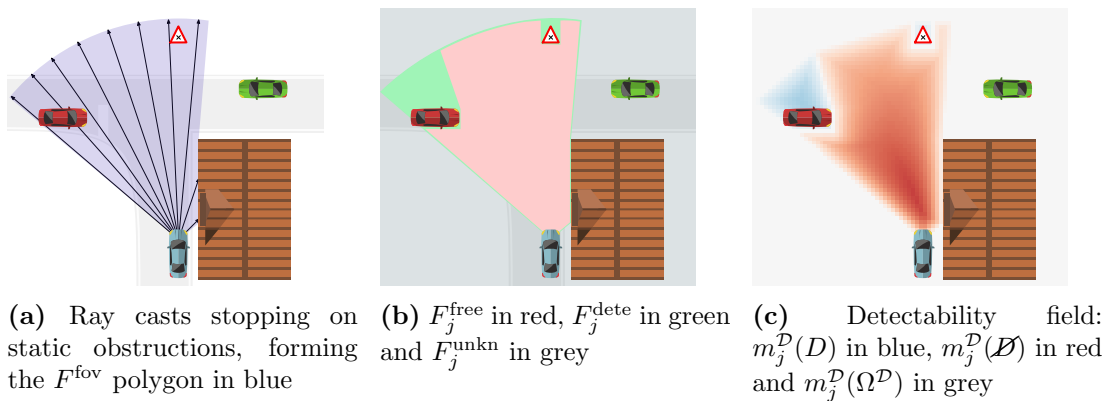


Figure 4.17: Steps to build the detectability of a sensor j .

Similar to (Aeberhard et al. 2011; Allig, Leinmüller, et al. 2019), the process starts by deriving the detection polygon from sensor characteristics (range and openings) and obstructions. A theoretical FOV polygon is built from characteristics and gets limited by obstructions by casting 2D virtual rays originating from

the sensor, as in Figure 4.17a. That way, an area hidden behind an obstruction will not be considered detectable. However, in our case we consider that only certain static objects (e.g: buildings, poles) occult view. Such objects are considered mapped and can be retrieved either from a standard or high definition (SD, HD) map such as Open Street Map (OSM). Moving objects are intentionally ignored because the shadow they cast cannot be properly modeled. As they are described in two dimensions as seen from above, it is not possible to capture that sensors placed high above the ground can see behind a small object. In addition, the size and position of perceived and tracked objects are highly uncertain in practice (due to the sparsity of LiDARs, projection errors of cameras or simply tracking errors). This effect cannot be ignored as it would mean that detectability would be inconsistent between points of view and even successive time steps. Until these issues can be properly modeled (with a three dimensional representation of uncertain obstructions), the best course of action is to ignore what causes them. The result of this process is thus a polygon F_j^{fov} for each sensor j .

In parallel, sensors measure areas that are explicitly free as described in Section 3.2.4. This produces a polygon F_j^{free} for each sensor j that should be contained entirely within F_j^{fov} . A third polygon $F_j^{\text{dete}} = F_j^{\text{fov}} \setminus F_j^{\text{free}}$ defines where objects can be detected, as they are in view but not where space is measured as free. Finally, everything outside of F_j^{fov} is considered unknown F_j^{unkn} as summarized in Figure 4.17b.

The detectability of a sensor j is a belief function

$${}^j m_p^{\mathcal{D}} : 2^{j\Omega^{\mathcal{D}}} \rightarrow [0, 1], \quad {}^j \Omega^{\mathcal{D}} = \{^j D, \emptyset\} \quad (4.4)$$

that expresses the evidence that for a given point in space $p = [x, y]$, sensor j is able to detect an object (${}^j D$) or that nobody should be able to (\emptyset).

It should be noted that detectability is defined on a frame of discernment that depends on the source. This is because detectability is *subjective*: it does not describe a physical reality but rather a potential that depends on the point of view. Undetectability on the other hand is an objective physical reality, as it describes space explicitly measured as free.

Although this difference in nature better captures the reality of sensor detectability, it requires special care when combining points of view, as it will be detailed in Section 4.3.2.3.

4.3.2.2 Computation of Detectability

Using the polygons described above, detectability is derived by following the idea that evidence decreases with distance from the sensor and near polygon borders.

For this, a distance function is first defined as

$${}^j d_{\iota, \kappa}^{\text{fov}}(p, F) := \left(1 - e^{-\|p, F\| \frac{\ln 2}{\iota}}\right) e^{-\|p - {}^j p_0\| \frac{\ln 2}{\kappa}} \quad (4.5)$$

where p and F are the point and the polygon at hand, ${}^j p_0$ is the origin of sensor j . ι and κ are two parameters describing how fast evidence should fade with respect

to boundary proximity and distance from the sensor respectively. Exponential functions are used to generate smooth boundaries, as illustrated in Figure 4.18.

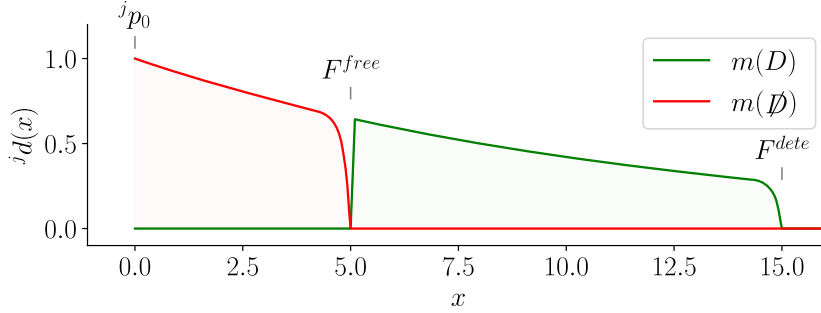


Figure 4.18: Example of detectability in one dimension with $j p_0 = 0$, $F^{\text{free}} = 5$, $F^{\text{dete}} = 15$, $\iota = 0.1$ and $\kappa = 8$.

A detectability mass is constructed by applying Equation (4.5) with respect to either the FS or FoV polygon. If the point is outside of either polygon, a fully unknown mass is attributed to it. This can be summarized with:

$$j m_p^{\mathcal{D}} = \begin{array}{c} \left[\begin{array}{cccc} \emptyset & \{D\} & \{\emptyset\} & \{D, \emptyset\} \\ \hline 0 & 0 & j d_{\iota, \kappa}^{\text{fov}}(p, F_j^{\text{free}}) & 1 - j d_{\iota, \kappa}^{\text{fov}}(p, F_j^{\text{free}}) \\ 0 & j d_{\iota, \kappa}^{\text{fov}}(p, F_j^{\text{dete}}) & 0 & 1 - j d_{\iota, \kappa}^{\text{fov}}(p, F_j^{\text{dete}}) \\ 0 & 0 & 0 & 1 \end{array} \right] \begin{array}{l} p \in F^{\text{free}} \\ p \in F^{\text{dete}} \\ \text{otherwise} \end{array} \end{array} \quad (4.6)$$

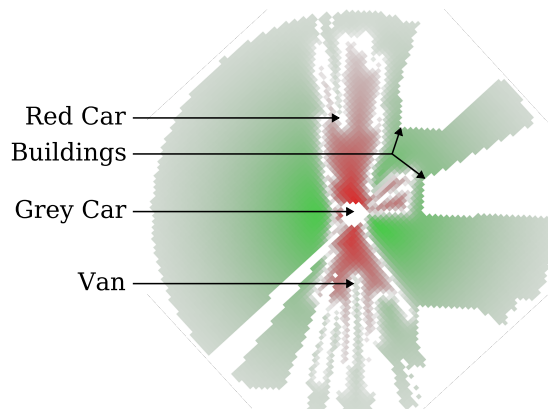
Finally, detectability is discretized in the form of a grid to save computations downstream and to help with visualizing multiple sources. To do this, Equation (4.6) is uniformly sampled across the ground plane such that all the j -FoV is covered. Letting p be the center of each cell falling in the FoV, the detectability grid of j is called $j m^{\mathcal{D}}$ and is defined as:

$$j m^{\mathcal{D}} = \{j m_p^{\mathcal{D}}\}_{p \in F_j^{\text{fov}}} \quad (4.7)$$

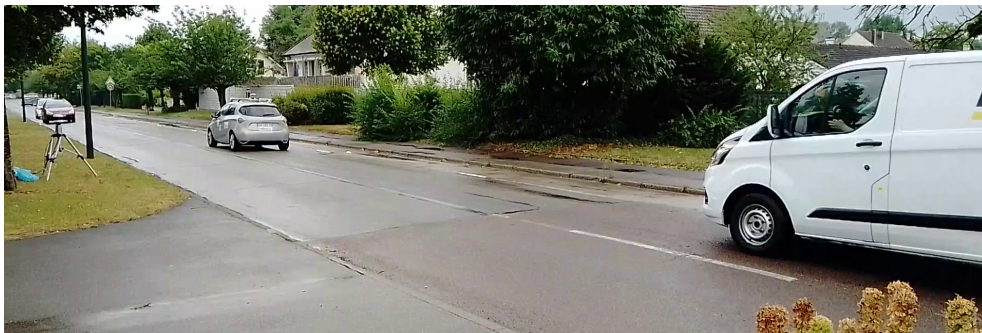
A grid generated by this process is illustrated in Figure 4.19. It is computed using real LiDAR data. Free space is measured following the method described in the previous chapter, and the LiDAR FoV is limited by buildings retrieved from OpenStreetMaps.

4.3.2.3 Fusion of Detectability Grids

In the next sections, detectability grids will be used to estimate the existence of objects and evaluate the coherency between objects and free space. When multiple field of view of sensors or peers overlap, performing these actions iteratively for each source can be impractical. Instead, one can combine detectability grids in advance and use the combined results. We consider that in the standalone case, each sensor computes its own detectability grid, leaving only the



(a) Top view of the detectability grid.



(b) Contextual view. A red car is visible on the left side, a grey car in the middle and a van on the right.

Figure 4.19: Real driving situation and example of detectability grid computed using LiDAR data. The LiDAR is attached to the grey car in the center of both images. Red cells correspond to space measured as free by the sensor, green cells to its field of view and white cells to unobserved or ambiguous areas. Notice that the red area is naturally stopped by dynamic objects (a red car in front of the grey car and a van behind it) and that the green area is limited by static objects (buildings on the grey car right side). In addition, notice how the intensity of green and red cells fades with distance and near the free space border.

task of combining them. However, in the cooperative case, exchanging grids can be unfeasible due to bandwidth limitations. Instead, poses, fields of view and measured free space can be exchanged, meaning that detectability grids have to be reconstructed and combined on the receiving side.

Because detectability is subjective, its combination requires special care. As illustrated in Example 4.4, a naive combination such as the conjunctive rule generates conflict when two sources disagree. In the case of detectability, disagreement is not a sign of conflict but a natural consequence of source subjectivity.

Example 4.4:

Consider the situation depicted in Figure 4.20. Two vehicles v_1 and v_2 have

their free space stopped by a third vehicle v_3 . Because v_3 is a dynamic object, it does not stop their FoV but does stop their free space, producing a detectable zone in its shadow (green polygons in Figure 4.20a and Figure 4.20b respectively). However, due to the different points of view, the detectability of one is the free space of the other, which results in conflict (blue polygon in Figure 4.20c). The real expected state is represented in Figure 4.20d.

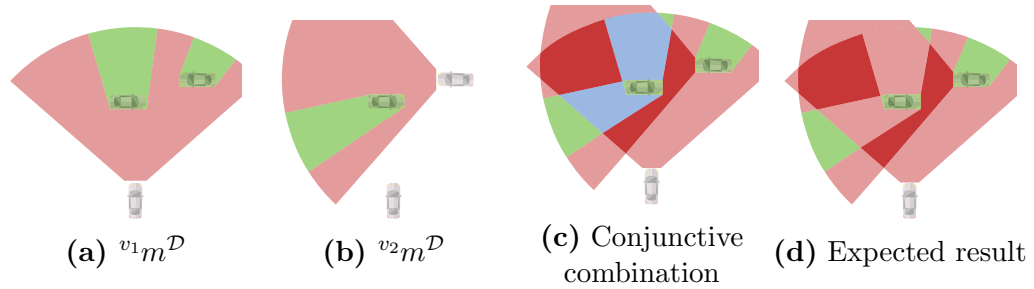


Figure 4.20: Example of naive detectability grid combination. Green is detectability, red is non-detectability (free) and blue conflict.

One method to prevent this undue conflict is to use the method described in Section 2.2.2.4 (page 28) to combine partially-overlapping frames. Indeed, with our formulation, undetectability is common to all frames of discernment, meaning that they intersect in at least one element. Let p the point or cell of interest, and J the set of sensors to combine. Our proposition works in two steps.

In the first step, Smets rule (\oplus) is used to combine masses on the union of all frames $\Omega_J^{\mathcal{D}} = \bigoplus_{j \in J} {}^j\Omega^{\mathcal{D}}$ as

$${}^Jm_p^{\mathcal{D}} = \bigoplus_{j \in J} {}^jm_p^{\mathcal{D}} \quad (4.8)$$

In the second step, the combined mass ${}^Jm_p^{\mathcal{D}}$ is projected onto $\Omega^{\mathcal{D}} = \{D, \emptyset\}$. D is defined as the objective detectability. It is approached by combining subjective detectabilities ${}^JD = \{{}^jD\}_{j \in J}$. On the other hand, because undetectability comes from direct free space measurements, it is not approached, and as such it will take precedence in the combination.

These operations can be summarized as

$$m_p^{\mathcal{D}} = \left[\begin{array}{cccc} \emptyset & \{D\} & \{\emptyset\} & \{D, \emptyset\} \\ 0 & {}^JBel_p^{\mathcal{D}}({}^JD) & {}^JPl_p^{\mathcal{D}}(\emptyset) - {}^Jm_p^{\mathcal{D}}(\Omega_J^{\mathcal{D}}) & {}^Jm_p^{\mathcal{D}}(\Omega_J^{\mathcal{D}}) \end{array} \right] \quad (4.9)$$

Example 4.5:

Consider a given point p seen by two sensors $J = \{i, j\}$. The global frame of

discernment is therefore:

$$\Omega_j^{\mathcal{D}} = \{^iD, ^jD, \emptyset\}$$

If both sensor agree that p is detectable, the results of the conjunctive and Smets combinations are illustrated in the following example:

$$\begin{array}{l} {}^i m_p^{\mathcal{D}} = \\ {}^j m_p^{\mathcal{D}} = \\ \textcircled{\oplus} m_p^{\mathcal{D}} = \\ {}^J m_p^{\mathcal{D}} = \end{array} \left[\begin{array}{c|cccccccc} \emptyset & \{D^i\} & \{D^j\} & \{D^i, D^j\} & \{\emptyset\} & \{D^i, \emptyset\} & \{D^j, \emptyset\} & \{D^i, D^j, \emptyset\} \\ \hline 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0.75 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.75 & 0 & 0 & 0 & 0.25 \end{array} \right]$$

If the sensors disagree (here i considers it is detectable and j undetectable), the results are:

$$\begin{array}{l} {}^i m_p^{\mathcal{D}} = \\ {}^j m_p^{\mathcal{D}} = \\ \textcircled{\oplus} m_p^{\mathcal{D}} = \\ {}^J m_p^{\mathcal{D}} = \end{array} \left[\begin{array}{c|cccccccc} \emptyset & \{D^i\} & \{D^j\} & \{D^i, D^j\} & \{\emptyset\} & \{D^i, \emptyset\} & \{D^j, \emptyset\} & \{D^i, D^j, \emptyset\} \\ \hline 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{array} \right]$$

Notice how the conjunctive combination produces conflict in both cases. Afterwards, the combined masses are projected onto $\Omega^{\mathcal{D}}$ following Equation (4.9). In the first case, this gives

$$\left[\begin{array}{c|cccc} \emptyset & \{D\} & \{\emptyset\} & \{D, \emptyset\} \\ \hline 0 & 0.75 & 0 & 0.25 \end{array} \right]$$

and in the second

$$\left[\begin{array}{c|cccc} \emptyset & \{D\} & \{\emptyset\} & \{D, \emptyset\} \\ \hline 0 & 0 & 0.5 & 0.5 \end{array} \right]$$

Notice how in the first case, the mass of D is increased because both sources agree, while in the second, the mass of \emptyset does not change as nothing confirms it.

Assuming that grids are in the same frame of reference (i.e. transformed in a global reference frame) and aligned, they can be combined by applying Equation (4.9) over every cell independently. Asynchronicity is managed by discounting cells of older grid. Because detectability grids convey dynamic information

that cannot properly be predicted (a common issue of using grids), it is as if they were losing information as time passes and so a discounting is applied:

$${}^j m^{\mathcal{D}}(t|t-1) = \Lambda(\Delta t, t_{1/2}^{\mathcal{D}}) {}^j m^{\mathcal{D}} \quad (4.10)$$

4.3.2.4 Object Detectability

Finally, as the goal of detectability is to identify detectable or non-detectable objects, it is defined for an object as $m_o^{\mathcal{D}}$ being the maximum detectability across that object. For this, the object 2D bounding box is computed from its position and size as defined in Equation (3.33). Key points c_o of the bounding box are taken to approximate its shape, such as its center and corners:

$$m_o^{\mathcal{D}} = \max_{\langle x,y \rangle \in c_o} m_{x,y}^{\mathcal{D}} \quad (4.11)$$

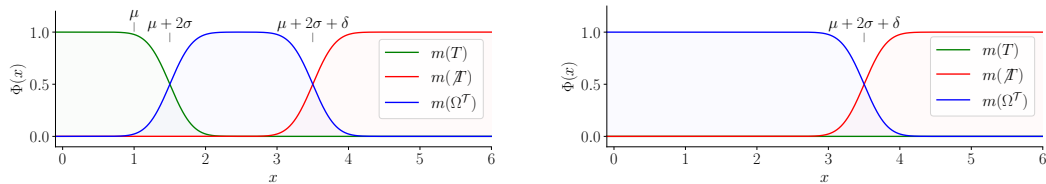
and maximum detectability is defined based on the following relation order:

$$m_1^{\mathcal{D}} > m_2^{\mathcal{D}} \Leftrightarrow m_1^{\mathcal{D}}(D) > m_2^{\mathcal{D}}(D) \quad (4.12)$$

4.3.3 Similarity between Objects

As reviewed in Section 3.3.1 (page 60), the distance between objects or tracks is generally used to denote the difference between objects. This implicitly assumes that close objects are the same which can sometimes lead to mis-associations. It can thus be useful to add comparisons on other characteristics, such as what (N. Zoghby et al. 2013) proposed. In this work belief functions are used to combine the similarity of characteristics such as position, kinematics and class. What is particularly interesting is the distinction that some characteristics can provide similarity (i.e. position) while other can only provide dissimilarity (i.e. kinematics, class).

In this section, similarity is extended to other characteristics and adapted to use a sigmoid function Φ . Φ constructs a mass function on $\Omega = \{H, \mathcal{H}\}$ from a scalar value to model that as the value increases, it supports an arbitrary hypothesis H then nothing then the opposite hypothesis \mathcal{H} , as depicted in Figure 4.21.



(a) Φ with parameters chosen to model support, unknown then disapproval

(b) Φ with parameters chosen to model unknown then disapproval

Figure 4.21: Sigmoids of two different mass functions.

It is defined as

$$\Phi^H(x, \mu, \sigma, \delta) := \left[\begin{array}{cccc} \emptyset & \{H\} & \{\mathcal{H}\} & \{H, \mathcal{H}\} \\ 0 & L\left(-\frac{2(x-2\sigma-\mu)}{\sigma}\right) & L\left(\frac{2(x-2\sigma-\mu-\delta)}{\sigma}\right) & 1 - m(\{H\}) - m(\{\mathcal{H}\}) \end{array} \right] \quad (4.13)$$

where L is the logistic function, μ , σ and δ are three scalar parameters respectively describing the positive tipping point (i.e. where H starts losing evidence), the rate of change and the negative tipping point (i.e. where \mathcal{H} start gaining evidence).

The similarity between two objects o and \mathbf{o} is defined as a belief function $m_{o,\mathbf{o}}^S$ defined on frame $\Omega^S = \{S, \mathcal{S}\}$ expressing that o and \mathbf{o} either correspond to the same relevant physical object or not. It is constructed as a combination of sub-similarities:

$$m^S = m^{\text{pos}} \odot m^{\text{kine}} \odot m^{\text{size}} \odot m^{\text{class}} \odot m^{\text{chara}} \quad (4.14)$$

That compare several aspects of objects. As a reminder, objects have been defined in Equation (3.33) as a state $\mathbf{x} = \langle x, y, \theta, v, \dot{\theta} \rangle$ (composed of a x and y position, heading θ , velocity v and rotational velocity $\dot{\theta}$), a covariance \mathbf{P} , a size $\mathcal{I} = \langle l, w, h \rangle$ (composed of a length l , width w and height h) a class c and a set of characteristics Z . As such, similarity of two objects i and j is defined by:

- m^{pos} the position similarity. It can be computed using the Mahalanobis distance (or Normalized Estimation Error Squared (NEES)) between the two objects. Different poses imply different objects and similar poses imply similar objects. As such, letting μ^{pos} , σ^{pos} and δ^{pos} describe the shape of the pose sigmoid, pose similarity is defined as:

$$\varepsilon^{\text{pos}} = \begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix} \cdot (\mathbf{P}_i + \mathbf{P}_j)^{-1} \cdot \begin{bmatrix} x_i - x_j & y_i - y_j \end{bmatrix} \quad (4.15)$$

$$m^{\text{pos}} = \Phi(\varepsilon^{\text{pos}}, \mu^{\text{pos}}, \sigma^{\text{pos}}, \delta^{\text{pos}})$$

- m^{kine} the kinematic vectors similarity. It can be computed using the Mahalanobis distance between the two vectors. Two similar kinematics do not mean similar objects, but different kinematics means different objects. As such, letting μ^{kine} , σ^{kine} and δ^{kine} describe the shape of the kinematic sigmoid, kinematic similarity is defined as:

$$\varepsilon^{\text{kine}} = \begin{bmatrix} \theta_i - \theta_j \\ v_i - v_j \\ \omega_i - \omega_j \end{bmatrix} \cdot (\mathbf{P}_i + \mathbf{P}_j)^{-1} \cdot \begin{bmatrix} \theta_i - \theta_j & v_i - v_j & \omega_i - \omega_j \end{bmatrix} \quad (4.16)$$

$$m^{\text{kine}} = \Phi(\varepsilon^{\text{kine}}, \mu^{\text{kine}}, \sigma^{\text{kine}}, \delta^{\text{kine}})$$

- m^{size} the size similarity. It can be computed using the difference of volume of both objects. Two similar sizes do not imply similar objects, but different sizes imply different objects. As such, letting μ^{size_l} , σ^{size_l} , δ^{size_l} , μ^{size_w} , σ^{size_w} ,

$\delta^{\text{size}w}$, $\mu^{\text{size}h}$, $\sigma^{\text{size}h}$ and $\delta^{\text{size}h}$ describe the shape of the length, width and height sigmoids, size similarity is defined as:

$$\begin{aligned} m^{\text{size}} &= \Phi(|l_i - l_j|, \mu^{\text{size}l}, \sigma^{\text{size}l}, \delta^{\text{size}l}) \\ &\oplus \Phi(|w_i - w_j|, \mu^{\text{size}w}, \sigma^{\text{size}w}, \delta^{\text{size}w}) \\ &\oplus \Phi(|h_i - h_j|, \mu^{\text{size}h}, \sigma^{\text{size}h}, \delta^{\text{size}h}) \end{aligned} \quad (4.17)$$

These parameters can be fixed by studying the capacity of sensor detectors at providing accurate sizes. This study has been conducted in the previous chapter and has shown that our detectors were not accurate. As such, size parameters are fixed to large values so that it is not too discriminating.

- m^{class} the class similarity. It can be computed using a confusion matrix D^2 . Two identical classes do not imply identical objects, but different classes imply different objects:

$$m^{\text{class}} = \{0, 0, D(c_i, c_j), 1 - D(c_i, c_j)\} \quad (4.18)$$

- m^{chara} the characteristics similarity. This can be for example plate numbers or traffic light color. Some similar characteristics will imply similar objects but not all.

Table 4.1 summarizes the different tuning parameters of similarity:

Table 4.1: Summary of similarity parameters.

Notation	Domain	Description
μ^{pos}	R^+	Position NEES from which two objects stop being similar
σ^{pos}	R^+	How quickly a rising NEES decreases similarity
δ^{pos}	R^+	NEES from which two objects are considered dissimilar
μ^{kine}	R^-	Kinematic NEES from which two objects stop being similar
σ^{kine}	R^+	How quickly a rising NEES decreases similarity
δ^{kine}	R^+	NEES from which two objects are considered dissimilar
μ^{size}	R^-	Volume from which two objects stop being similar
σ^{size}	R^+	How quickly a diverging volume decreases similarity
δ^{size}	R^+	Volume from which two objects are considered dissimilar
D	R^+	Cost of confusing a class for another

²Matrix introduced in Equation (3.7) describing the cost of associating two classes. Small for semantically close classes (e.g. stop and yield sign) and large for semantically far classes (e.g. pedestrian and truck).

4.3.4 Estimating the Existence of Tracked Objects

Tracks are managed using a belief function on their existence. The estimation of track existence at sensor-level (detailed in Section 3.5.2.2 on page 77) is extended here to multiple points of view with the idea that the estimation of object existences should be limited by the detectability of the source at hand. For example, a front-facing camera should not interfere with the existence of rear-facing objects since their fields of view do not overlap.

Birth and prediction are still defined based on two parameters m^{birth} and $t_{1/2}^{\exists}$ summarized in Table 4.2 on page 107, but update and deletion are modified to account for detectabilities and similarities.

For this, a tracker i computes the detectability of a source ${}^j m^{\mathcal{D}}$ upon reception and maintains a *consensus* grid ${}^J m^{\mathcal{D}}$ that combines the detectability of all recently received sources $J = \{j, \dots\}$. The order of operations is thus:

1. Compute the detectability grid of j ${}^j m^{\mathcal{D}}$ with Equation (4.7)
2. Predict the previous consensus grid ${}^J m^{\mathcal{D}}$ with Equation (4.10)
3. Predict the existence of tracks ${}^i \mathcal{O}$ with Equation (3.43)
4. Update the consensus grid ${}^J m^{\mathcal{D}}$ with ${}^j m^{\mathcal{D}}$ and Equation (4.9)
5. Update the existence of tracks ${}^i \mathcal{O}$ with Equation (4.19) or Equation (4.21)
6. Manage tracks based on their existence with Equation (4.22).

While in earlier sensor-level existence estimation, existence increased with a constant value, it is now dependent on the existence of observation \mathfrak{o} . However, several constraints have to be added to prevent mis-associations from biasing the existence estimation and to account for the potentially differing points of view between the sensor and consensus:

- The similarity of the associated track and observation $m_{\mathfrak{o},\mathfrak{o}}^{\mathcal{S}}$ ensures that both tracks correspond to the same physical object. The value of this term is close to 1 when detectable and 0 otherwise;
- The consensus plausibility of detection ${}^J Pl_{\mathfrak{o}}^{\mathcal{D}}(D)$ ensures that all sources agree that the object could be visible (i.e: not explicitly in their free-space). It is close to 1 in such cases and drops to 0 when more and more sources disagree. This forbids objects from being estimated in free-space;
- The source belief in detection ${}^j Bel_{\mathfrak{o}}^{\mathcal{D}}(D)$ ensures that the updating source can detect its track and is not extrapolating it. Here *Bel* is used instead of *Pl* as the nature of this constraint is different. While the previous term expressed a non-impossibility to detect objects, this one expresses the possibility of detecting one.

Update in the case of association The product of these three terms is a discounting factor applied to the observation existence $m_o^\exists(t)$. It can then be combined with the predicted track existence $m_o^\exists(t|t-1)$ using Dempster's rule:

$$\begin{aligned}\alpha^{\text{updt}} &= m_{o,o}^S(S) \cdot {}^j\text{Bel}_o^D(D) \cdot {}^J\text{Pl}_o^D(D) \\ m_o^\exists(t|t) &= m_o^\exists(t|t-1) \oplus_{\alpha^{\text{updt}}} m_o^\exists(t)\end{aligned}\quad (4.19)$$

Update in the absence of association Conversely when tracks are not associated (i.e. not observed), detectability can be used to estimate non-existence. The underlying idea is that detectable tracks should be observed again. As such, unobserved tracks are likely to be clutter. This is expressed similarly to Equation (4.19) by combining the estimated track existence with another discounted existence. Here, the other existence m^\nexists is constant and based on a parameter E^{updt} that controls how much unobserved tracks are considered clutter:

$$m^\nexists = \left[\begin{array}{cccc} \emptyset & \{\exists\} & \{\nexists\} & \{\exists, \nexists\} \\ 0 & 0 & E^{\text{updt}} & 1 - E^{\text{updt}} \end{array} \right] \quad (4.20)$$

As out-of-range tracks should not be penalized for not being detected, the detectability of the updating source ${}^j\text{Bel}_o^D(D)$ is used to discount the penalty m^\nexists :

$$\begin{aligned}\alpha^{\text{updt}} &= {}^j\text{Bel}_o^D(D) \\ m_o^\exists(t|t) &= m_o^\exists(t|t-1) \oplus_{\alpha^{\text{updt}}} m^\nexists\end{aligned}\quad (4.21)$$

Track management By introducing non-detectability in the estimation, a second condition can be added to delete tracks if too much evidence points to the track being clutter:

$$m_o^\exists(\Omega^\exists) > E^{\text{forget}} \wedge m_o^\exists(\mathcal{E}) > E^{\text{clutter}} \quad (4.22)$$

Compared to existing methods (Aeberhard et al. 2011) that would decay track existence based on their detectability, this explicitly models that detectable objects should be detected. This model adds two parameters to those summarized in Table 3.4 at page 77. The complete summary of existence parameters is thus given in Table 4.2.

Table 4.2: Summary of existence tracking parameters.

Notation	Domain	Description
E^{birth}	$[0, 1]$	Birth evidence of existence
E^{forget}	$[0, 1]$	Delete tracks whose information is below this threshold
$t_{1/2}^\exists$	R^{+*}	Time to losing half a track existence information
E^{updt}	$[0, 1]$	How much unassociated tracks are considered clutter
E^{clutter}	$[0, 1]$	Delete tracks whose clutter evidence is above this

4.4 Evaluation of Cooperative Perception

4.4.1 Evaluation Methodology

Evaluating perception with real data is a difficult task, as mentioned in Section 3.4. We explain here the methodology we followed.

4.4.1.1 Global and Local Evaluations

With single PoV datasets, any annotated ground-truth object should be detected by the perception system, and any missed or additional object is an error. However, when multiple PoVs are considered, objects perceived by one PoV can be hidden or out of reach to another, while still functioning nominally. In single-PoV such as (Ettinger et al. 2021), tracks disappear and reappear when they get in and out of Field of View (FoV). In multi-PoV situations, the real visibility of all peers in the scene should be modeled and labeled, which is hard and sometimes impossible.

To address this issue, we have identified two evaluation methods:

- **Global evaluation:** Ground-truth contains all the information held by all peers at all times. It represents the maximal amount of information in the considered system. This evaluation method is useful for making comparisons between perception systems as the global information is the same for all systems. It is however not adapted to evaluate the integrity of each perception system in the sense given in the general introduction. Indeed, the global ground-truth does not account for the task being realized by every vehicle. As such it will provide pessimistic evaluations that do not represent the actual integrity level of each perception system, especially in terms of miss-detections.
- **Local evaluation:** Here the ground-truth information is only composed of the information relevant to the task at hand for a given perception system. The electronic horizon of a navigation system can for example be used to limit the ground-truth in areas that will interact with the vehicle path. This method is useful to evaluate the integrity of a particular perception system but cannot be used to compare several systems. Indeed, because the task at hand changes from one vehicle to another, local ground-truths thus have to be different.

In the following studies, we will compare several systems. The global method will thus be used, but note that vehicle-specific integrity evaluation should use the local method.

4.4.1.2 Datasets with Ground-Truth

To evaluate cooperative perception systems, a ground-truth is necessary. Unfortunately, a ground-truth is rarely provided with cooperative datasets. The

existing datasets are either not based on real data or not based on moving vehicles, which is why we recorded our own. Several scenarios have been recorded, but as only one of them has been used for evaluation in this manuscript, they are described in Appendix B.

The dataset used for comparison is based on a complex overtaking scenario. As understanding the sequence of events will be important to interpret results, it is first introduced in the following.

As illustrated in Figure 4.22, the sequence is composed of five vehicles: three are perceiving vehicles (v_1 , v_2 and v_3) and two are controlled but not perceiving vehicles (v_4 and v_5). The dataset has been recorded on open roads, and other road users were also present. In particular, there was a vehicle in front of v_1 and behind v_2 .

The sequence can be split into five sub-sequences:

1. At first, between $t = 0$ and 10 (time is expressed in seconds), two groups of two (v_3 and v_5) and three (v_1 , v_2 and v_4) vehicles start on opposite sides of the road. Only v_1 , v_2 and v_3 are equipped with sensors and communication. v_3 perceives v_5 for the whole sequence while v_1 and v_2 can both see v_4 but not each other.
2. Secondly, between $t = 10$ and $t = 20$, all vehicles start moving. During this period v_1 and v_2 can see each other.
3. Between $t = 20$ and $t = 36$, both groups approach each other on a straight line.
4. Between $t = 36$ and $t = 44$, v_4 stops, forcing v_1 to also stop.
5. Finally, between $t = 44$ and $t = 55$, v_1 overtakes v_4 .

A ground truth has been manually labeled for this dataset. Though a method has been proposed to help with the labeling process, it has not been used for producing the results of this work because this method was not finalized in time. It is thus described in Appendix C.

The following describes the manual labeling process. Point clouds from all vehicles are projected in a common 2D plane, as illustrated in Figure 4.23a. Helped by context images, bounding boxes are drawn by a human annotator for key time stamps. Bounding boxes are then manually linked between time stamps to provide interpolable *tracks*. The annotator guarantees that all objects in certain areas (i.e. the road) are annotated. Outside of that area, some can be missed. This is used to form the free space grid as the absence of objects in that area as illustrated in Figure 4.23b.

4.4.1.3 Evaluation Metrics

To evaluate and compare perception systems studied in this manuscript, a common set of metrics has been defined. The three main characteristics of a perception system are evaluated. They are described hereafter.

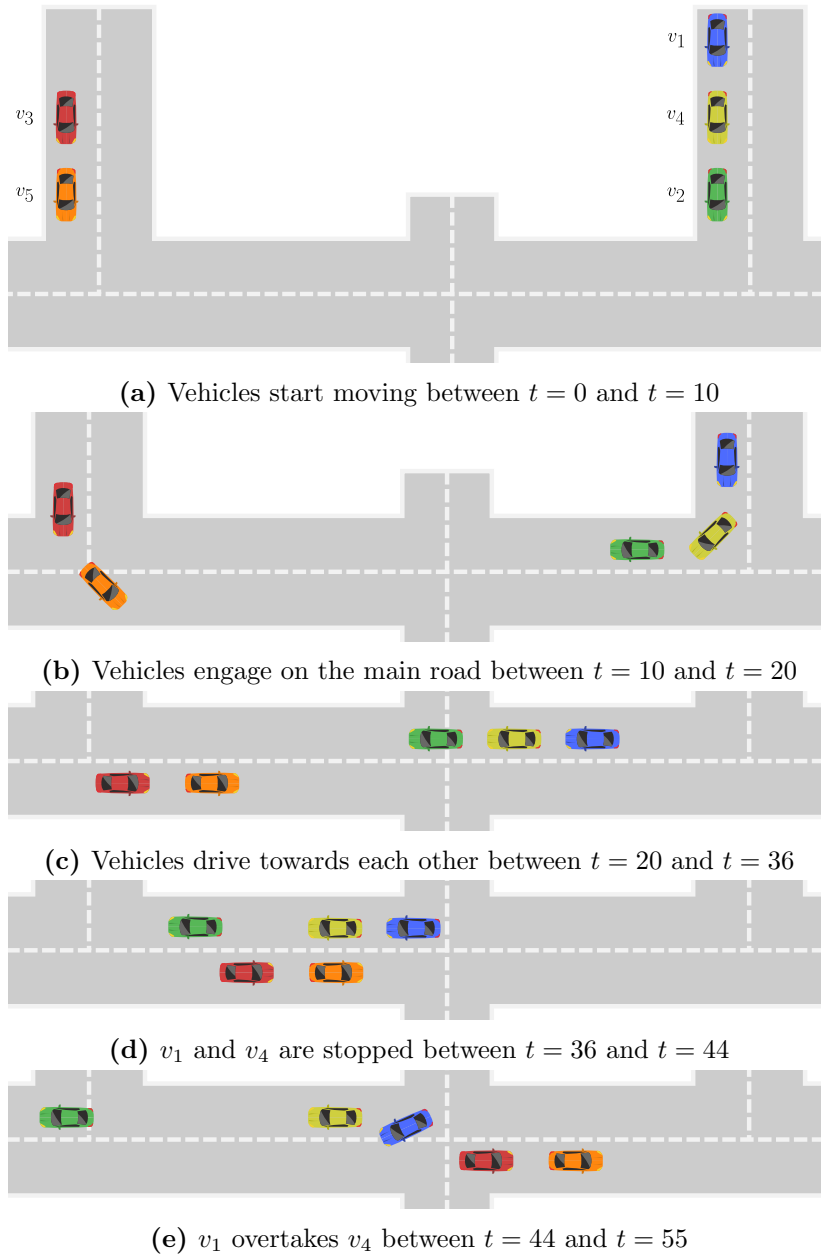


Figure 4.22: Schematic representation of the sequence of events of the overtaking scenario.

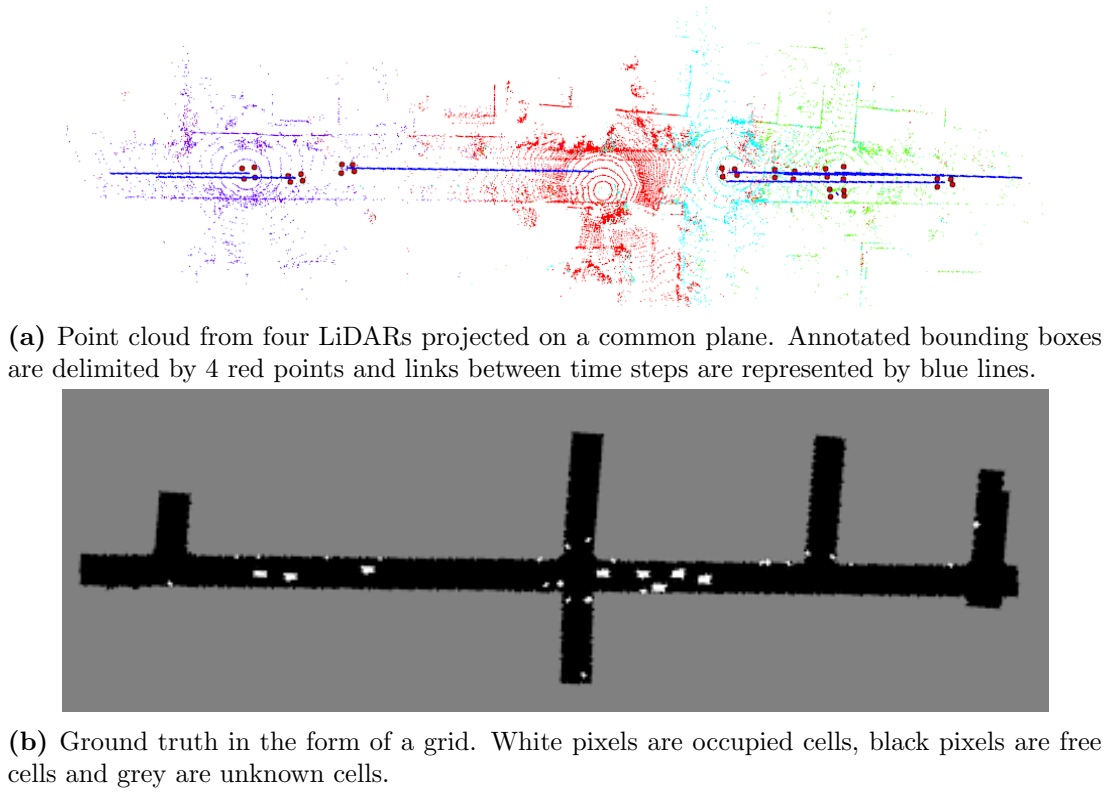


Figure 4.23: Example of ground truth generation for a given time step.

a) Object Existence To evaluate the object detection rates, ground truth objects \mathbf{G} are retrieved and synchronized with time-stamps of the perception system.

The objects to be evaluated are denoted \mathcal{O} . They are separated for each time $t \in \mathbb{T}$ in several object lists whose existence is above given thresholds $\tau \in \mathcal{T}$:

$$\mathcal{O}(t, \tau) = \{o\}_{o \in \mathcal{O}(t), \text{Bel}_o^{\exists}(E) > \tau} \quad (4.23)$$

They are then associated to ground truth objects \mathbf{G} using a Global Nearest Neighbors (GNN) algorithm with Mahalanobis distances. This gives sets of associated objects $\psi(t, \tau)$. Associations are then used to form True Positives (TPs), False Positives (FPs) and False Negatives (FNs) as:

$$\begin{aligned} TP^{\mathcal{O}}(\tau) &= \{\mathcal{O}_i(t, \tau)\}_{(i,j) \in \psi(t, \tau), t \in \mathbb{T}} \\ TP^{\mathbf{G}}(\tau) &= \{\mathbf{G}_j(t)\}_{(i,j) \in \psi(t, \tau), t \in \mathbb{T}} \\ FP(\tau) &= \mathcal{O}(t, \tau) \setminus TP^{\mathcal{O}}(\tau) \\ FN(\tau) &= \mathbf{G}(t) \setminus TP^{\mathbf{G}}(\tau) \end{aligned} \quad (4.24)$$

As illustrated in Figure 4.24, $TP^{\mathcal{O}}(\tau)$ is the set of perceived objects that have been associated ground truth. $TP^{\mathbf{G}}(\tau)$ is the set of ground truth objects that have been associated with perceived objects. $FP(\tau)$ is the set of perceived objects that have not been associated, corresponding to *ghost* objects that do not actually

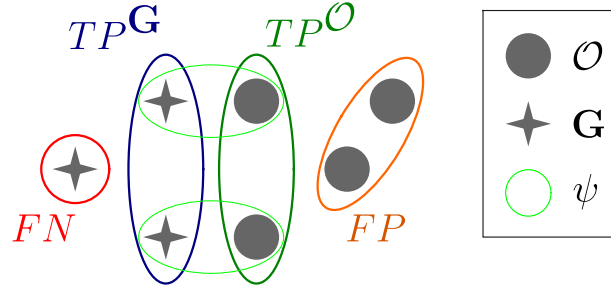


Figure 4.24: Sets used in the association process of object evaluation.

exist. $FN(\tau)$ is the set of ground truth objects that have not been associated, corresponding to missed detection of real objects.

Note that True Negatives (TNs) cannot be computed because enumerating everywhere objects are not does not make sense in a world that is infinite. Evaluation metrics relevant to this problem are therefore the precision, recall and F1-score, whose definitions have been given in Table 3.2, at page 66. A precision-recall curve can be constructed by plotting precision against recall for various thresholds τ .

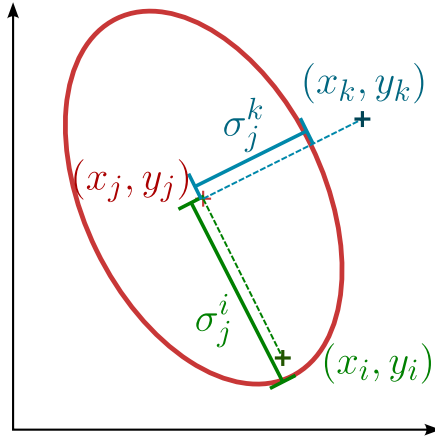


Figure 4.25: Error and uncertainty computation of two estimates i and k when compared to the ground truth j . Here, k and i are at the same distance of j , but only i is consistent (within σ_j^i). k is not because its error is higher than σ_j^k .

b) Object Accuracy and Consistency To evaluate the accuracy and consistency of tracked object state estimates, the association $\psi(t, \tau)$ is reused. The error vector \mathbf{e}_j^i is computed for each pair $\langle i, j \rangle \in \psi(t, \tau)$ of object indices. Then the absolute 2D error ϵ_j^i and its associated uncertainty σ_j^i are computed. As illustrated in Figure 4.25, σ_j^i corresponds to the uncertainty along the error vector, according to the method of (Tao et al. 2016):

$$\mathbf{e}_j^i = \begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix} \quad \epsilon_j^i = \|\mathbf{e}_j^i\| \quad \sigma_j^i = \sqrt{\frac{1}{\frac{\mathbf{e}_j^i T}{\epsilon_j^i} \cdot \mathbf{P}_i^{-1} \cdot \frac{\mathbf{e}_j^i}{\epsilon_j^i}}} \quad (4.25)$$

where \mathbf{P} is the covariance matrix.

Error plots or Stanford diagrams (see Section 1.2.2 at page 16) can then be constructed by plotting all ϵ_j^i against their associated estimated uncertainty σ_j^i for all times $t \in T$ and a given threshold τ . Note however that due to the large number of objects considered in this scenario, Stanford diagrams are too hard to interpret.

Instead, we plot the average RMSE and standard deviation across all objects for each time step. The mean RMSE $\hat{\epsilon}$ and the mean standard deviation $\hat{\sigma}$ are computed as:

$$\hat{\epsilon} = \frac{\sum_{t \in T} \sum_{\langle i,j \rangle \in \psi(t,\tau)} \epsilon_j^i}{\sum_{t \in T} |\psi(t,\tau)|}, \quad \hat{\sigma} = \frac{\sum_{t \in T} \sum_{\langle i,j \rangle \in \psi(t,\tau)} \sigma_j^i}{\sum_{t \in T} |\psi(t,\tau)|} \quad (4.26)$$

for a threshold τ fixed at 0.50.

An interesting metric to evaluate the consistency ζ is the percentage of times the error is included in a chosen confidence domain. In the following example, 3σ confidence domains have been chosen which corresponds to a consistency level of 99% for a χ^2 law with two degrees of freedom:

$$\zeta(\tau) = \frac{\sum_{t \in T} \sum_{\langle i,j \rangle \in \psi(t,\tau)} \begin{cases} 1 & \epsilon_j^i \leq 3 \cdot \sigma_j^i \\ 0 & \epsilon_j^i > 3 \cdot \sigma_j^i \end{cases}}{\sum_{t \in T} \sum_{\langle i,j \rangle \in \psi(t,\tau)} 1} \quad (4.27)$$

c) Free-Space To evaluate the quality of free space detection, ground-truth grids m^G are retrieved, synchronized (by finding their closest detectability grids in terms of time) and compared to detectability grids m^D elaborated by the perception system for each time steps t . Similarly to objects, cells of the detectability grids are separated using a threshold τ :

$$C^{\emptyset}(t, \tau) = \{c\}_{c \in m^D(t), Bel_c^D(\emptyset) > \tau} \quad (4.28)$$

However, because the opposite of "free" is "occupied" and not just "not free", areas where information is not available are ignored:

$$C^D(t, \tau) = \{c\}_{c \in m^D(t), m_c^D(\Omega^D) < 1, Bel_c^D(\emptyset) \leq \tau} \quad (4.29)$$

Then, assuming that both grids are temporally and spatially aligned, cells are compared one to one following Table 4.3.

Note that this formulation is not complete because detectability is a potential and not a description (i.e. detectable cells are not necessarily occupied, though occupancy should only appear in detectable cells). As such, free cells identified as detectable are not necessarily FNs but occupied cells identified as detectable are indeed TNs. This effect can falsely degrade results but is a first approximation. A better modeling would be to restrict detectability under perceived objects, but this begs for a joint estimation methodology between objects and free-space. This is a possible improvement.

Table 4.3: Cell-wise evaluation of perceived free space by ground truth grids.

	$m^{\mathbf{G}} = F$	$m^{\mathbf{G}} = O$
$C^{\mathcal{P}}(t, \tau)$	TP	FP
$C^D(t, \tau)$	FN	TN

4.4.2 Study of the Added-Value of Cooperative Perception

In this first evaluation, the points of view of three vehicles are tracked cooperatively according to Section 4.3.1. The outputs of standalone and cooperative trackers are compared for the three vehicles with metrics introduced in the previous section. Those are the mean RMSE, standard deviation and consistency of object positions, maximal F1-score and PR-Area Under Curve (AUC) for object detection and ROC-AUC for free space. They are given in Tables 4.4 to 4.6 and illustrated in Figures 4.26 to 4.28.

One can see that cooperative perception significantly improves object detection for all vehicles compared to standalone perception, with a max F1-score and AUC improved between 0.10 and 0.15. This is also visible in Figure 4.26 for example with the cooperative (orange) curve going twice as far as the standalone (blue) one. It is here doubled as v_1 receives information from v_3 which is at the other side of the road for a majority of the dataset.

The contribution is not so clear for other aspects. For example, detection accuracy is sometimes better, sometimes worse depending on the car. This can be explained by the metric used (the mean RMSE of all objects) meaning that inaccurate objects from one vehicle are propagated to all, worsening their mean. This is particularly visible in Figure 4.28a. Between $t = 00$ and $t = 10$, only two well observed objects were visible in standalone. In cooperative, objects from the other side of the scene get visible but they are not as well observed and thus the mean RMSE increases.

Similarly for free space detection, the contribution of cooperation is not clear. It is however clear that including more points of view should improve the detection of free space, leading us to believe that the evaluation methodology is the culprit. Possible causes can be cells mis-alignment or Equation (4.29) that does not produce the expected result. Nevertheless, a slight improvement in TPR can be observed, at the expense of a shorter range of FPR, which is still an improvement.

Table 4.4: Comparison of standalone and cooperative perception for v_1 .

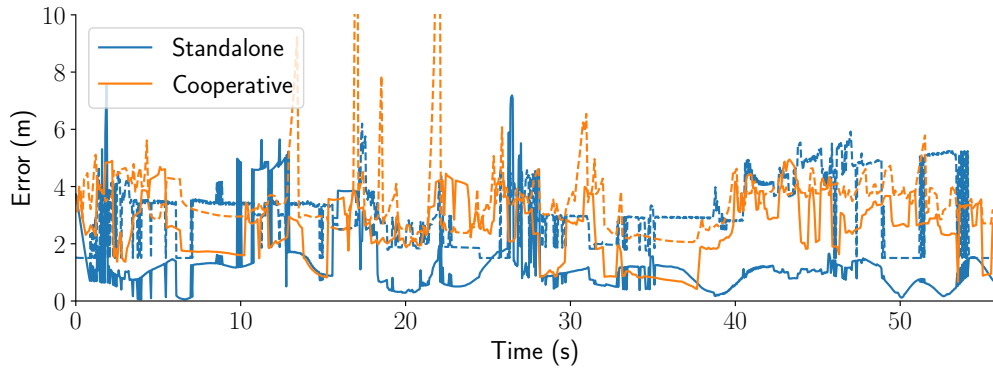
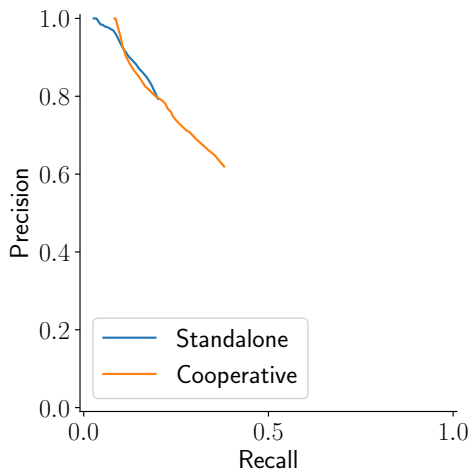
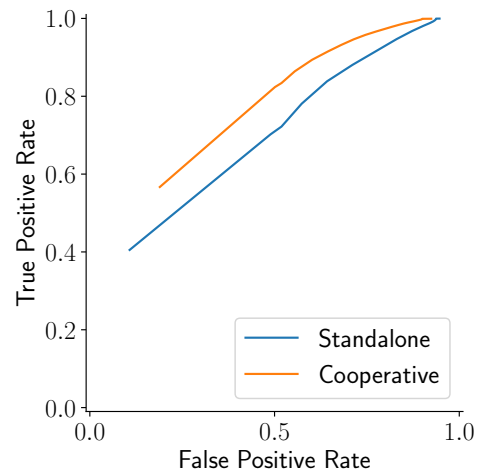
	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Standalone	1.28	1.02	0.88	0.32	0.16	0.61
Cooperative	1.52	0.86	0.90	0.47	0.23	0.61

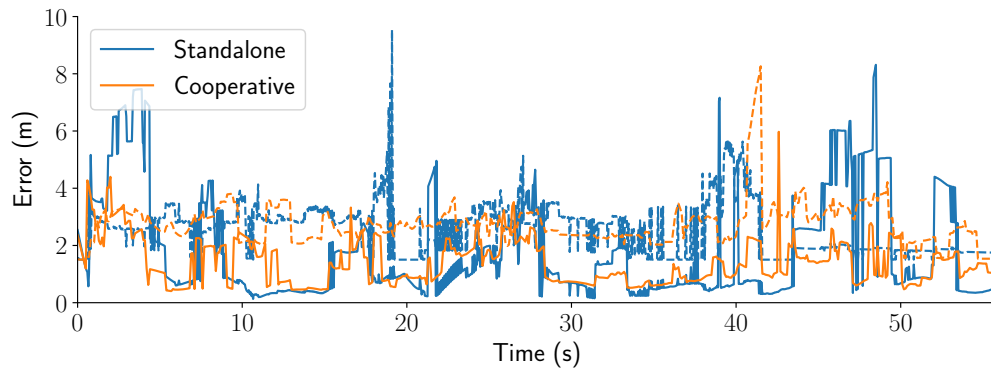
Table 4.5: Comparison of standalone and cooperative perception for v_2 .

	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Standalone	1.82	0.84	0.72	0.31	0.14	0.67
Cooperative	1.40	0.92	0.93	0.47	0.23	0.61

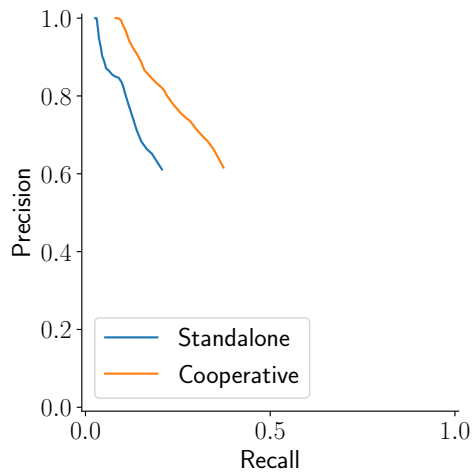
Table 4.6: Comparison of standalone and cooperative perception for v_3 .

	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Standalone	0.80	1.03	0.93	0.23	0.10	0.57
Cooperative	1.53	0.89	0.91	0.47	0.23	0.61

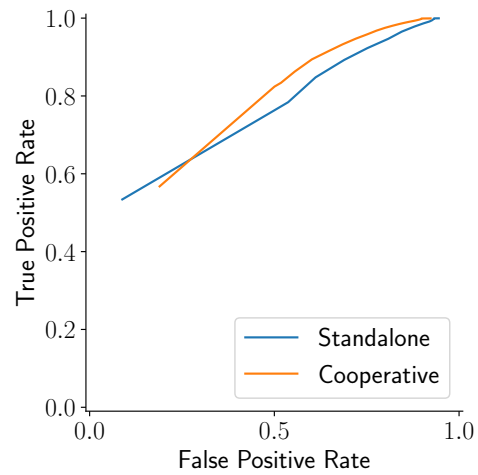
**(a)** Object 2D errors (continuous) and 3σ confidence bounds (dashed)**(b)** Object existence PR Curves**(c)** Free-Space ROC Curves**Figure 4.26:** Comparison of standalone and cooperative perception for v_1 . Blue curves are standalone perception and orange cooperative.



(a) Object 2D errors (continuous lines) and covariances (dashed lines)

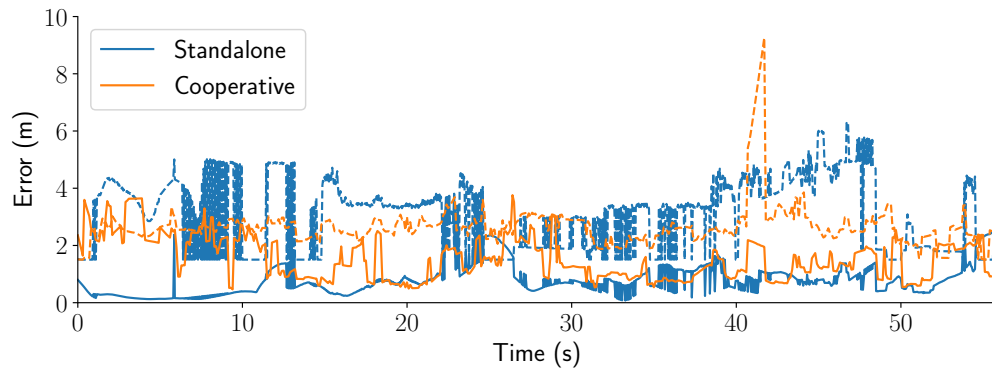


(b) Object existence PR Curves

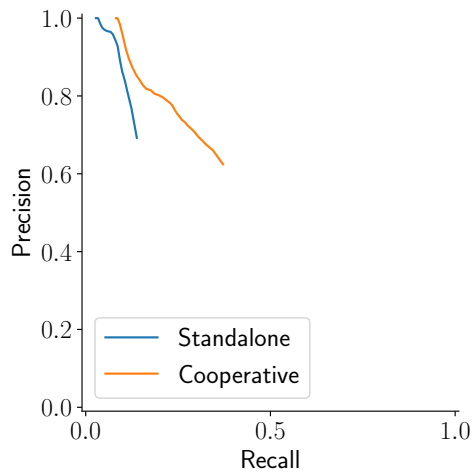


(c) Free-Space ROC Curves

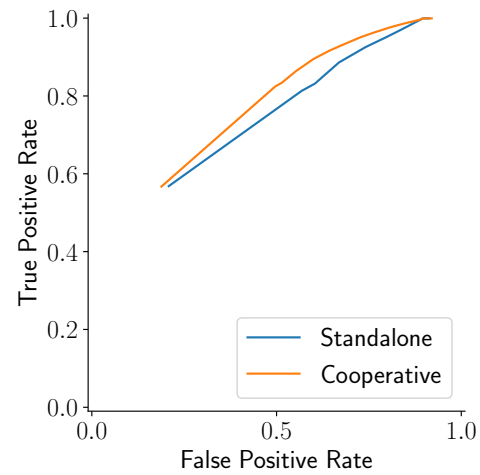
Figure 4.27: Comparison of standalone and cooperative perception for v_2 . Blue curves are standalone perception and orange cooperative.



(a) Object 2D errors (continuous lines) and covariances (dashed lines)



(b) Object existence PR Curves



(c) Free-Space ROC Curves

Figure 4.28: Comparison of standalone and cooperative perception for v_3 . Blue curves are standalone perception and orange cooperative.

4.4.3 Study of the Contribution of Detectability for Cooperative Perception

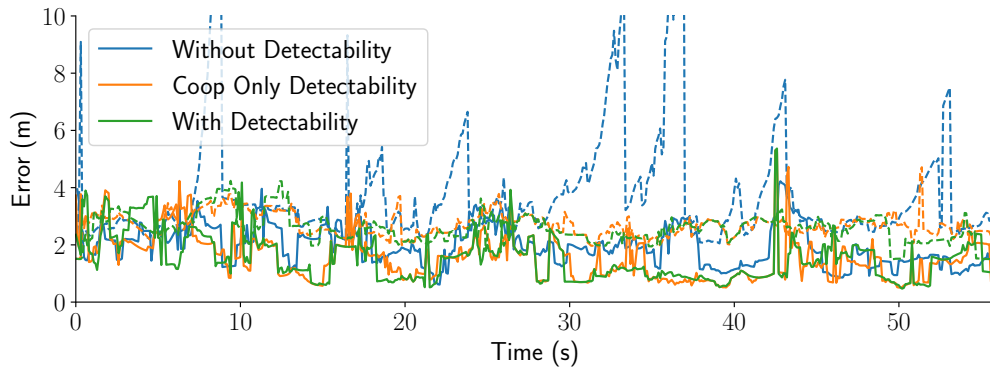
The second evaluation is about the contribution of detectability to the estimation of object existence for a cooperative perception system. The same methodology as before is used but only one vehicle will be shown for the sake of clarity. For this, three variants have been run on the three vehicles:

1. Without detectability (i.e. all objects are considered fully detectable when estimating their existence);
2. With detectability used only in the cooperative tracker;
3. With detectability in both standalone and cooperative trackers.

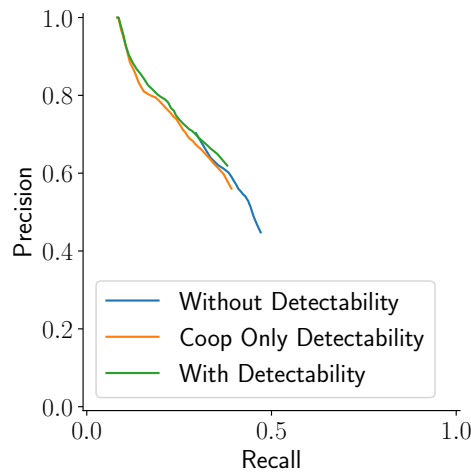
The evaluation of the cooperative perception for these three variants is provided in Table 4.7 and illustrated in Figure 4.29. In these, the contribution of detectability is clear. Objects are significantly better detected when passing from a system without detectability to one that uses it. A slight improvement can be observed when passing from a system where detectability is used only when cooperating to a system where it is used everywhere. This is expected as our vehicles only possess one sensor, so the standalone detectability does not have a significant impact. Detectability thus improves the capacity of the tracker to estimate the existence of objects. Correct objects are maintained for longer periods of time in the case of multiple PoVs, and erroneous objects are removed sooner thanks to the inclusion of undetectability. This is visible for example on Figure 4.29a at $t = 32$ where tracks are erroneously duplicated, leading to large covariances being estimated. Without detectability, this effect spans several seconds until the erroneous tracks are removed due to not being observed for a long time. With detectability, duplicates diverge quickly into free space and are thus removed sooner. This reduced the mean error on the whole sequence, with for example the RMSE and variance that get improved as well. Free space detection on the other hand is not impacted by this comparison and thus provides the same results in all variants.

Table 4.7: Comparison of no detectability, cooperative detectability and full detectability.

	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Without Detectability	1.97	1.27	0.91	0.48	0.10	0.61
Coop Only Detectability	1.55	0.90	0.92	0.46	0.23	0.61
With Detectability	1.52	0.86	0.90	0.47	0.23	0.61



(a) Object 2D errors (continuous lines) and covariances (dashed lines)



(b) Object existence PR Curves

Figure 4.29: Comparison of no detectability (blue curves), cooperative detectability (orange curves) and full detectability (green curves) for an arbitrary vehicle v_1 .

4.5 Conclusion

In this chapter, we have introduced a generic architecture to combine the perception of several points of view. This is proposed to solve the problem of data fusion in cooperative perception by seeing other vehicles as additional points of view. Note that we believe this architecture to be beneficial to multi-sensor perception for standalone perception as well. This architecture is based on a generic interface composed of a list of objects, free space polygons and fields of view. Three aspects have been studied: Track-To-Track Fusion (T2TF), the representation of source Points of View (PoVs) and the estimation of object existence. T2TF is realized with the SCIF and has shown to be an effective tool for the task. Representation of a source point of view is realized with detectability, a concept that has been introduced to combine FoV and free space in a joint and discretized representation. It is complex to manipulate as detectability is a potential while undetectability is a description of physical reality, meaning that its fusion and evaluation are not straightforward and must be managed with care.

We have proposed solutions to solve these problems. Despite these limitations, it has been shown to be effective in helping cooperative trackers to improve their object detection. We believe that this representation should be useful to scene understanding modules as it gives a cooperative estimation of areas that have been covered. Finally, for the estimation of object existence, we have introduced a formulation that includes the similarity between track and observation and both of their detectabilities.

Together, these concepts and methods provide an improvement in the detection of object and free space but evidence is lacking regarding the improvement of object accuracy. Further analysis is required on this aspect, for example by computing errors object-wise and on track predictions to verify their predictability. Other analyses could also be carried out further, such as accounting for the fact that certain areas of the road are more important to properly perceive than others (and how this impacts the availability of the perception module). Accounting for dynamic objects in the definition of detectability is also another interesting point to study further, in particular to represent situations where a vehicle hides perception behind it. Finally, an important work that remains is to derive a joint evaluation metric for the three aspects of perception (accuracy, object and free space detection). They have been studied separately but are all tied (i.e. object accuracies depends on their detection, free space and object detection are two sides of the same coin). Double counting might happen in this case, making the tuning process harder to interpret.

Chapter 5

Estimation of Trust in Cooperative Peers

Contents

5.1	Introduction	121
5.2	Review of Trust in Intelligent Vehicles	121
5.3	Trust Estimation and Use for Data Fusion	124
5.4	Experimental Evaluation	135
5.5	Conclusion	145

5.1 Introduction

Up to this chapter, peers were considered trustworthy and reliable. Each agent receiving information from the others integrated and merged it with confidence to improve and expand its perception of the environment. In practice, this assumption cannot be guaranteed on open roads as erroneous or even malicious peers can be present in the cooperative network.

In this chapter, we introduce the concept of trust, a quantity estimated over time that represents how much a peer (and thus the information it sends) can be trusted. After reviewing how this aspect is considered in the literature in Section 5.2, we formulate our model of trust, its computation and its use in the data fusion process in Section 5.3. The approach is then evaluated on hybrid real and simulated data in Section 5.4.

5.2 Review of Trust in Intelligent Vehicles

According to (van der Heijden et al. 2019), the attacker model in Cooperative Intelligent Transportation Systems (cITS) is a peer sending erroneous information. (Ansari et al. 2021) details several attacks and how they can be mitigated using information contained in CPMs. The distinction between malicious or not

is irrelevant as the attacker has limited presence in the network due to its limited range of communication. The assumption of honest majority is credible due to security measures introduced in Section 4.2.1.4.

To do this, most approaches estimate a quantity between 0 and 1 representing the trustworthiness of received data. For example in (Ambrosin, Yang, et al. 2019), it is the probability of a peer being trustworthy that is estimated, while in (Allig, Leinmüller, et al. 2019) it is the confidence in correctness of received data. In other approaches, un-trustworthiness is explicitly estimated as well, with for example (Hurl et al. 2020) extending the estimate range to $[0, 1] \cup \{-1\}$ and (Schmidt et al. 2008) to $[-2, 1]$, from untrustworthy through unknown to trustworthy.

5.2.1 Misbehavior Detection

Trust estimation relies on misbehavior detectors, functions that compare what is received with other sources of information (prior, ego or cooperative information for example). These are extensively reviewed in (van der Heijden et al. 2019) where the authors differentiate three types of detectors: node-centric, plausibility checks and consistency checks depending on which type of information is compared.

Node-centric checks focus on the information outside of exchanged messages such as a peer’s respect of protocol and timings. Plausibility checks compare received and ego information. For example, in (Obst et al. 2014) a received position is compared to the receiver own perception to detect ghost objects. In (Bißmeyer et al. 2013), the position of a peer is checked against sudden jumps by predicting previously received positions. Similarly, in (Yavvari et al. 2017), the velocity of a peer is compared to its previously received velocities and constrained in curvature base on the peer size. Finally, in (Ambrosin, Yang, et al. 2019), authors define four levels of anomalies that summarize previous detectors:

- Message level: peers are in communication range, their objects are reported once, do not suddenly appear and their characteristics are within predefined ranges;
- Model level: received objects follow a plausible trajectory;
- Perception level: received objects overlap with the receiver perception;
- System level: received objects overlap with the perception of others.

The last level (system level) is a coherency check as it uses external information. This idea is also developed in (Allig, Leinmüller, et al. 2019) with two situations depicted in Figure 5.1: a peer can be validated by observing common objects or by observing another trusted peer. In this approach, consistency of perceived objects is checked for pairs of peers and fused using Bayesian filtering.

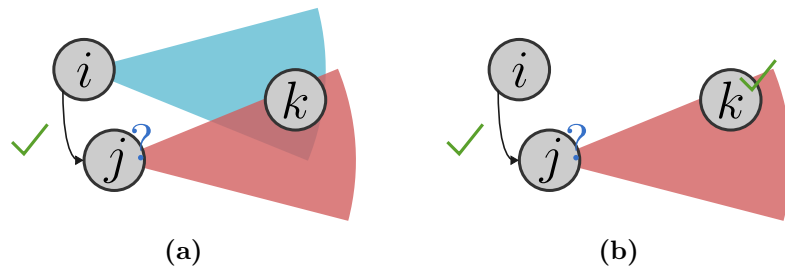


Figure 5.1: Validation situations in (Allig, Leinmüller, et al. 2019). In (a), i trusts j because they each observe a third object k . In (b), i trusts j because it observed a verified object k .

In (Hurl et al. 2020) trust is defined for each object o as its average observability from all peers $j \in V$ weighted by the actual detection:

$$\mathcal{T}(o) = \frac{\sum_{j \in V} \Phi^j(o) \cdot e^j(o)}{\sum_{j \in V} \Phi^j(o)} \quad (5.1)$$

where $e^j(o)$ is the overlapping score based on bounding box Intersection over Union (IoU) and $\Phi(o)$ the observability score based on LiDAR projections. Finally, in (Liu et al. 2021) objects are a joint object-grid consistency check is proposed.

5.2.2 Aggregation

Once trusts are computed, how they are used is conditioned to whether they are exchanged or not. Several approaches produce reports about a peer when it is deemed untrustworthy. Reports can be aggregated in a global or decentralized manner to ignore or evict untrusted peers. To avoid reporting abuse, (Zhuo et al. 2009) proposed a suicide-based reporting where the reporting node also evicts itself from the network. To help with the decision of evicting reported peers, (Ambrosin, Yang, et al. 2019) defines two notions: unobservability and undecidability. An area is unobservable by a particular peer if its sensors do not cover that area, and objects are undecidable by a particular peer if they are in an unobservable area or do not match with any of their objects. In (Schmidt et al. 2008) on the other hand, recommendations are sent about trustworthy peers instead of misbehavior reports. When the aggregation is decentralized, simple counting methods can be sufficient (Allig, Leinmüller, et al. 2019), but more advanced methods can be reused from other fields. For example, (Zacharia et al. 2000) proposes to represent reputation as a graph. Two algorithms can be used to compute trust between two agents: HISTOS, that finds a path between them by breadth-first search and computes their trust as the product of that path, and SPORAS that recursively applies a complex expression combining reputations and ratings. The first is particularly adapted to densely connected networks, while the second is more adapted to sparse networks.

5.3 Trust Estimation and Use for Data Fusion

5.3.1 Fusion Architecture with Trust Management

Based on the review of existing methods, it can be said that the general goal of trust is to prevent erroneous peers from propagating erroneous information to other vehicles.

Centralizing the information on a global server raises privacy issues as well as networking challenges, whereas a decentralized approach only suffers from a limited view. This is not necessarily a problem, as trusting other can be a cautious process especially when considering the problem of information integrity. As cooperative perception is mainly a way of extending the range of perception, it is preferable to underestimate trust than to overestimate it. Underestimating trust reduces the availability of downstream navigation systems (because the vehicle will have less information to anticipate) but overestimating it might allow misleading information to reach the navigation system and cause the vehicle to make hazardous decisions.

In the architecture of the previous chapter, peers were modeled as remote sensors. Following this logic, trust can be seen as a reliability check applied to remote sensors and used to bring the information quality on the same level as local sensors. As such, a trust module is added onto Figure 4.15 between remote peers and the cooperative tracking module in Figure 5.2.

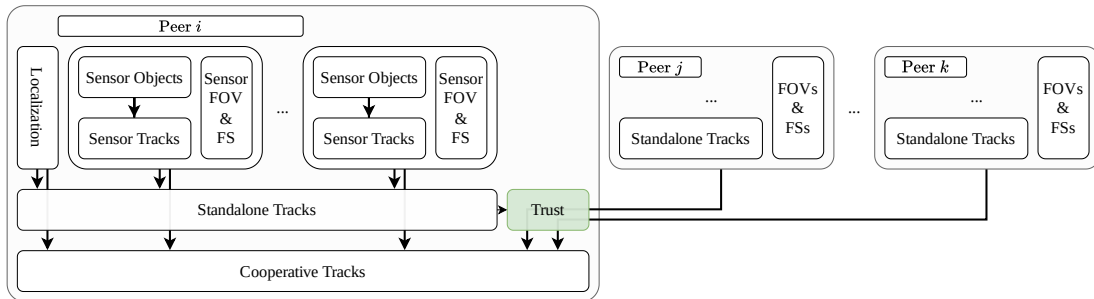


Figure 5.2: Trust augmented multi-peer data fusion architecture. Peer perception goes through a trust module before being fused in the cooperative tracking module.

In this formulation, trust between Humans is imitated. It is estimated independently about other peers and used to take information communicated by peers more or less cautiously. It starts with a mostly unknown prior, increases or decreases with new elements based on comparisons with a-priori, internal or external information.

5.3.2 Evidential Estimation of Trust

The process of estimating trust in others is based on negative and positive feedback that we will be presenting in the following. Note that this work has been published in (Lima, Cherfaoui, et al. 2022).

To convey both aspects under the same formalism, belief functions are used. They will also ultimately be useful to provide nuance when using trust. Trust is defined as a belief function $m_j^{\mathcal{T}}$ on $\Omega^{\mathcal{T}} = \{T, \mathcal{F}\}$ denoting that a peer is trustworthy or untrustworthy (i.e. its information can be taken without further considerations or should be considered with care). For the sake of clarity, we consider the point of view of i and will ignore the prefixed upper script in $m_j^{\mathcal{T}}$ when unambiguous.

It is estimated through time using a prediction-update estimator similar to state filtering:

$$m_j^{\mathcal{T}}(t|t) = \Lambda_{\Delta t} m_j^{\mathcal{T}}(t - \Delta t|t - \Delta t) \oplus m_j^{\mathcal{T}}(t) \quad (5.2)$$

where $m_j^{\mathcal{T}}(t - \Delta t|t - \Delta t)$ is the trust of previous time step and Δt the elapsed time between two steps. The prediction is implemented with the time-discounting operator $\Lambda_{\Delta t}$ presented in Equation (2.17) on page 29 and the update is done by fusing the predicted trust with an *observation* $m_j^{\mathcal{T}}(t)$ using Dempster's rule (\oplus). Revision operators (Ma et al. 2011) were considered but add too much complexity for little added benefit as the frame of discernment only contains two elements.

The trust *observation* is constructed by fusing several Basic Belief Assignments (BBAs) that represent basic checks. They are combined in the form of a tree, which is depicted in Figure 5.3. In that tree, vertices are conjunctive combinations. Edges are discounted by a fixed amount that is not noted for the sake of clarity.

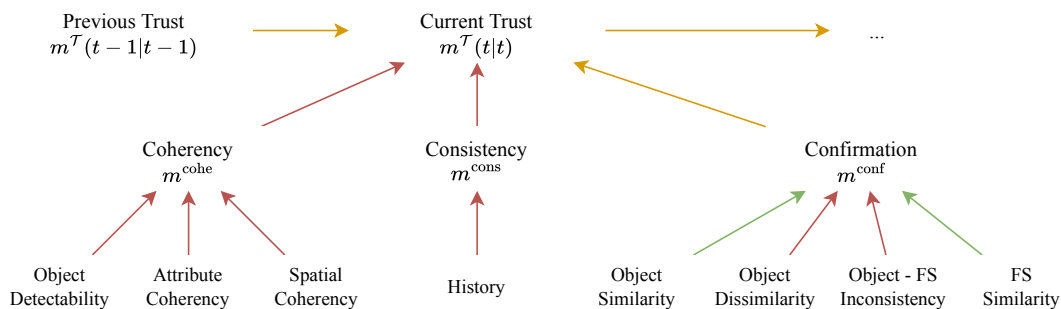


Figure 5.3: Evidential tree for trust estimation. Trust of BBAs on the bottom, their combination in the middle and the resulting filtering for time t on the top. Red arrows only convey untrustworthiness, green trustworthiness, and orange both.

BBAs express non-idiomatic and simple constraints on $\Omega^{\mathcal{T}}$. BBAs are first grouped thematically within three groups:

$$m_j^{\mathcal{T}}(t) = m_j^{\text{cohe}} \oplus m_j^{\text{cons}} \oplus m_j^{\text{conf}} \quad (5.3)$$

In the following, we describe how they are evaluated.

5.3.3 Coherency

m^{cohe} models that the information contained in a message has to be coherent within itself.

$$m_j^{\text{cohe}} = m_j^{\text{obd}} \oplus m_j^{\text{atc}} \oplus m_j^{\text{spc}} \quad (5.4)$$

5.3.3.1 Object Detectability

m_j^{obd} expresses that received objects should be in the detectability area of the emitting source. For this, the object detectability ${}^j m_o^{\mathcal{D}}$ from the point of view of j is used (see the previous chapter at page 103). A constant threshold D^{min} is used to detect objects with low detectabilities (mass on D^j) as this means that they are either unknown (${}^j \Omega^{\mathcal{D}}$) or even in free space (\emptyset).

An untrustworthiness mass parameterized with a constant β^{pen} is generated for each object whose detectability is too low:

$$m_j^{\text{obd}} = \bigoplus_{\substack{o \in O_j \\ {}^j m_o^{\mathcal{D}}(D) < D^{\text{min}}}} \left[\begin{array}{cccc} \emptyset & \{T\} & \{\mathcal{X}\} & \{T, \mathcal{X}\} \\ 0 & 0 & \beta^{\text{pen}} & 1 - \beta^{\text{pen}} \end{array} \right] \quad (5.5)$$

Example 5.1:

For example, received objects that are in the received FS are by definition undetectable and their detectability will be low. Similarly objects outside the FoV are unknown and will have a low detectability.

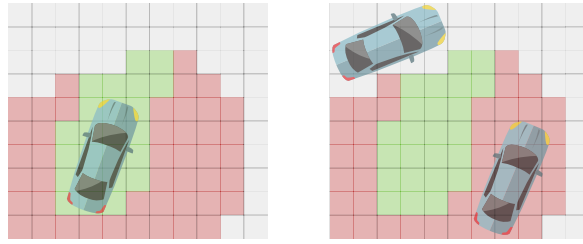


Figure 5.4: Received object and detectability grid. Green is detectable, red undetectable and grey unknown. On the left is a normal situation, a car is fully included in the detectable and nothing is outside. On the right is an abnormal situation with cars outside of the detectable areas, which can be a track spoofing.

5.3.3.2 Attribute Coherency

m_j^{atc} expresses that object attributes Z_o (e.g. velocity, size or covariance) have to be likely with predefined rules.

For each attribute z_o in Z_o , m_j^{atc} is defined as

$$m_j^{\text{atc}} = \bigoplus_{o \in O_j} \bigoplus_{z \in Z} \Phi(z_o, -\kappa, \sigma^z, \delta^z + \kappa) \quad (5.6)$$

with Φ the scalar sigmoid function introduced in Section 4.3.3. Here κ is a large value (e.g. 1000) fixed to ensure that Φ can only provide negative trust, while reducing the number of parameters to two (σ and δ).

Example 5.2:

For example, a car should have a length l within typical lengths range:

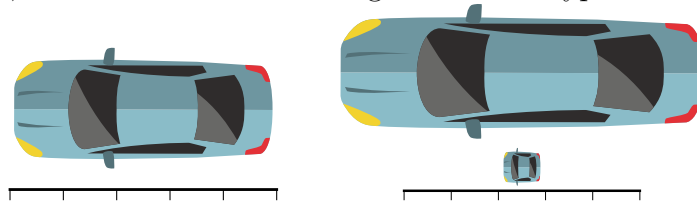


Figure 5.5: Received car. On the left its length is normal (within predefined ranges) while on the right its length is abnormal (too small or large). This can be track spoofing.

This can be expressed with $\sigma^l = 0.5$ and $\delta^l = 5$ which gives Figure 5.6.

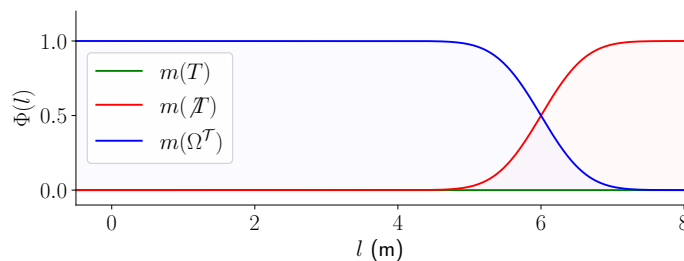


Figure 5.6: Sigmoid used for the detection of car length faults. $\sigma^l = 0.5$ and $\delta^l = 5$.

Another example is that an object velocity v has to be coherent with a normal behaviour. This can be done by reflecting speed limits with $\sigma^v = 1$ and $\delta^v = 15$ which gives Figure 5.7.

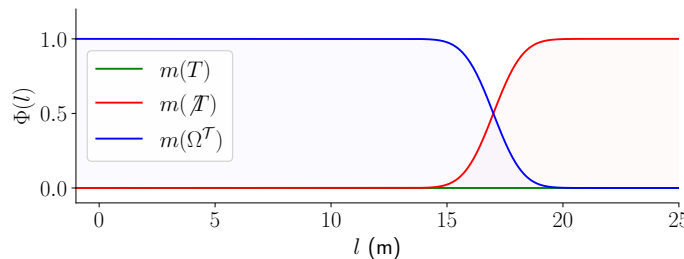


Figure 5.7: Sigmoid used for speed fault detection. $\sigma^v = 1$ and $\delta^v = 15$.

5.3.3.3 Spatial Coherency

m_j^{spc} expresses that objects have to be spatially coherent, that is be distinct from one another and be where they are expected. These constraints can be expressed as sigmoids Φ on distances D between objects and areas.

$$m_j^{\text{spc}} = \bigoplus_{d \in D} \Phi(\delta^d - d, -\kappa, \sigma^d, \delta^d - \kappa) \quad (5.7)$$

Example 5.3:

D can for example be composed of:

- Inter-object distances, computed for all pairs of objects $\langle o_1, o_2 \rangle$ in the received object list ${}^j\mathcal{O}$ as $\|o_1, o_2\|$;
- Car-road distances, computed for all cars $o \in {}^j\mathcal{O}$ as $\|R, o\|$ with R a polygon describing the shape of the road.

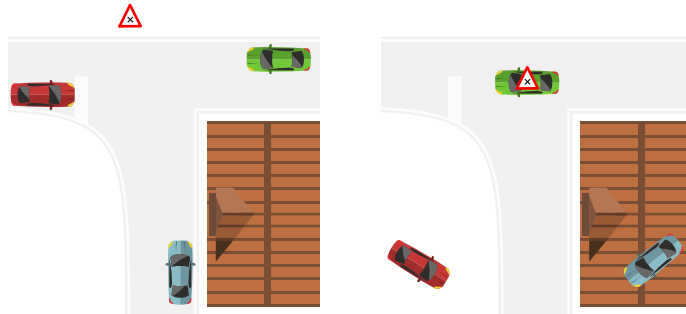


Figure 5.8: Received objects. On the left is a normal spacial distribution (objects are well separated and placed logically). On the right is an abnormal spacial distribution (objects are on top of each other and cars are inside buildings or outside the road), which can be a sign of track spoofing.

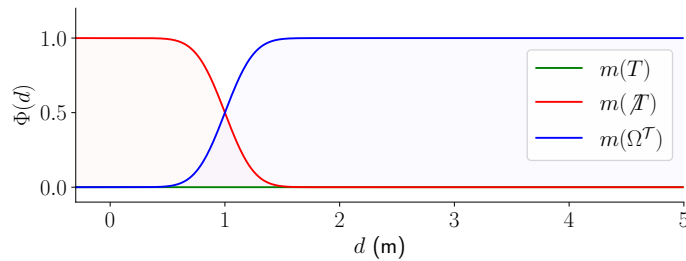


Figure 5.9: Sigmoid used for the detection of abnormal distances. $\sigma^d = 4$ and $\delta^d = 20$.

5.3.4 Consistency

The second group of checks is *consistency*, or the temporal coherency of objects received from a given peer. m^{cons} models that objects must follow coherent trajectories in time and not change their dynamics in unpredictable ways

Example 5.4:

For example, a track should not jump position between two messages as in Figure 5.10.

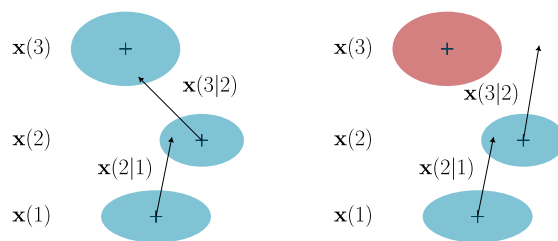


Figure 5.10: Received track across three time steps. On the left it follows a predictable trajectory. On the right it move outside of what the model can predict, which can be a track duplication for example.

For this, received objects are stored. Upon reception of a new object list, previous objects are predicted $O_j(t|t-1)$ and associated with new objects $O_j(t)$ using an assignment function noted \mathcal{A} as in Equation (3.40). The similarity mass function m^S from Section 4.3.3 is used to compare objects. It is then projected onto trust such that consistent objects are silent but those that do not match with their past generate untrustworthiness.

$$\begin{aligned} \psi &= \mathcal{A}(O_j(t|t-1), O_j(t)) \\ m_{O_j}^S &= \bigoplus_{o_1, o_2 \in \psi} m_{o_1, o_2}^S \\ m_j^{\text{cons}} &= \left[\begin{array}{cccc} \emptyset & \{T\} & \{\mathcal{I}\} & \{T, \mathcal{I}\} \\ 0 & 0 & m_{O_j}^S(\mathcal{I}) & 1 - m_{O_j}^S(\mathcal{I}) \end{array} \right] \end{aligned} \quad (5.8)$$

5.3.5 Confirmation

Previous sections described how errors can be detected. However this can only increase distrust and another mechanism must be derived in order to also increase trust. The proposed mechanism called *confirmation* compares received objects and free space against a personal *consensus*. Here, consensus refers to the fusion of all the information trusted by the ego vehicle. This can be for example the fusion objects and detectability grids from all sources trusted by the ego-vehicle, including cooperative peers. This aspect will be studied further in Appendix E. The general idea of confirmation is to generate trustworthiness when objects or free space are commonly perceived as it means that both understanding of the scene supports one another. On the other hand, when they mis-match, untrustworthiness can also be generated. Doing this, detectability of the receiving and sending peers are used to represent that comparisons cannot be made on non-overlapping areas. This implements the *unobservability* and *undecidability* concepts introduced in (Ambrosin, Yang, et al. 2019).

m_j^{conf} is defined as:

$$m_j^{\text{conf}} = m_j^{\text{osi}} \oplus m_j^{\text{odi}} \oplus m_j^{\text{ofi}} \oplus m_j^{\text{fsi}} \quad (5.9)$$

5.3.5.1 Object Similarity

First, m_j^{osi} models that trustworthy information should match with the local one. Received objects O_j are associated with consensus objects, resulting in $\psi = \{\langle o, \mathbf{o} \rangle, \dots\}$ the set of associated objects. Associated objects are compared using the similarity function m^S of Section 4.3.3. The plausibility of detecting of object \mathbf{o} , introduced in Section 4.3.2.4, is used to discount non-detectable object.

The confirmation BBA can be written as:

$$m_{O_j}^{\text{osi}} = \bigoplus_{\langle o, \mathbf{o} \rangle \in \psi} [Pl_o^{\mathcal{D}}(\emptyset)] m_{o, \mathbf{o}}^S$$

$$m_j^{\text{osi}} = \left[\begin{array}{cccc} \emptyset & \{T\} & \{\mathcal{X}\} & \{T, \mathcal{X}\} \\ 0 & m_{O_j}^{\text{osi}}(S) & 0 & 1 - m_{O_j}^{\text{osi}}(S) \end{array} \right] \quad (5.10)$$

This is done to only compare objects that are detectable by the consensus, as illustrated in Figure 5.11.

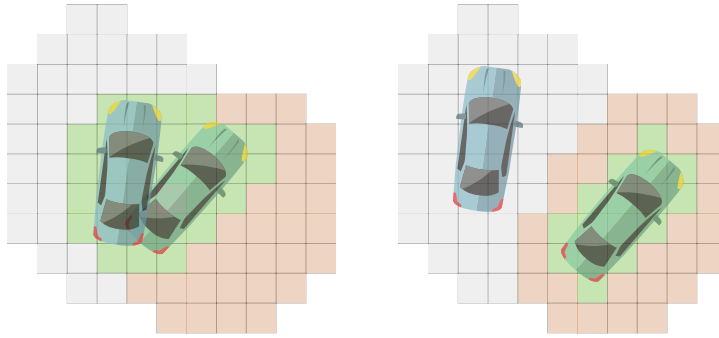


Figure 5.11: Comparison of a consensus object (green car) and received object (blue car). On the left, both objects match and are detectable (green cells) so the objects are similar and trust will be increased. On the right, the received object is not detectable (unknown grey cells or orange undetectable cells), so trust will not increase.

5.3.5.2 Object Dissimilarity

Conversely, m_j^{odi} models that received objects must not mis-match local ones. For this, objects are also associated, resulting in ψ the set of assignments. As illustrated in Figure 5.12, unassigned objects create non-trust depending on how detectable they are, modeling that they should have been. For this, the object detectability from the point of view of the consensus ($m_o^{\mathcal{D}}$) and from the point of view of j (${}^j m_o^{\mathcal{D}}$) is used as:

$$m_{O_j}^{\text{odi}} = \bigoplus_{o \in O_j, \mathbf{o} \notin \psi} [Pl_o^{\mathcal{D}}(\emptyset)] {}^j m_o^{\mathcal{D}}$$

$$m_j^{\text{odi}} = \left[\begin{array}{cccc} \emptyset & \{T\} & \{\mathcal{X}\} & \{T, \mathcal{X}\} \\ 0 & 0 & m_{O_j}^{\text{odi}}(D) & 1 - m_{O_j}^{\text{odi}}(D) \end{array} \right] \quad (5.11)$$

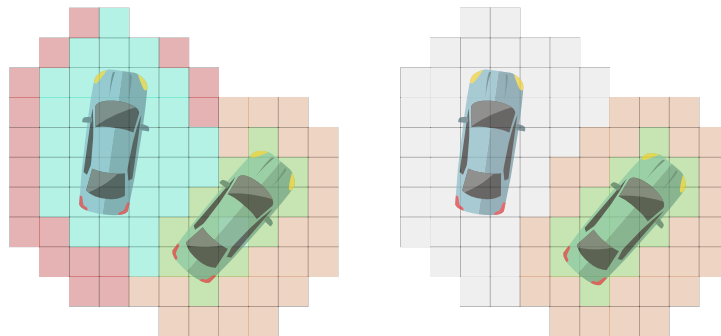


Figure 5.12: Comparison of a consensus object (green car) and received object (blue car). On the left is the same neutral situation as object similarity: objects do not match but due to the lack of detectability, non-trust is not increased. On the right, objects do not match both are both detectable by their sources (green and cyan cells for consensus and received respectively). Objects are dissimilar and detectable so non-trust is increased.

5.3.5.3 Object-Free-Space Inconsistency

m_j^{ofi} models that received objects must not be inconsistent with the consensus detectability. For this, the object detectability from the point of view of the consensus $m_o^{\mathcal{D}}$ is used. Objects falling in undetectable areas (i.e free space) thus produce distrust:

$$m_j^{\text{ofi}} = \bigoplus_{o \in j\mathcal{O}} \left[\frac{\emptyset \quad \{T\} \quad \{\mathcal{X}\} \quad \{T, \mathcal{X}\}}{0 \quad 0 \quad m_o^{\mathcal{D}}(\emptyset) \quad 1 - m_o^{\mathcal{D}}(\emptyset)} \right] \quad (5.12)$$

5.3.5.4 Free-Space Similarity

m_j^{fsi} models that received Free Space (FS) must match with the consensus. For this, the detectability over all cells p from the point of view of j (${}^j m_p^{\mathcal{D}}$) is compared to the consensus ($m_p^{\mathcal{D}}$). First, the number of cells where the consensus *supports* the free space (undetectability) measured by j is computed.

Support can be defined using metrics such as Jousselme Distance or Song-Deng divergence (Jousselme et al. 2001; Yutong Song et al. 2019). However, these metrics consider consonant belief (i.e. same focal sets but different amounts of information in each) to be divergent. In addition, they do not account for the total amount of information. This can be seen in Figure 5.13 with the fact that undetectable cells (blue pixels) are considered divergent to unknown cells (white pixels).

To address these issues, we propose the following support function. For a given cell p , it computes the distance between $m_p^{\mathcal{D}}$ and ${}^j m_p^{\mathcal{D}}$, weighted by the quantity of information contained in $m_p^{\mathcal{D}}$. In other words, cells where the reference is uncertain will not discriminate received cells even if their value differ significantly. Let $\sigma^{\mathcal{D}}$ be a tuning parameter describing how quickly support increases, support

is defined as:

$$D^{\text{support}}(m_p^{\mathcal{D}}, j m_p^{\mathcal{D}}) := \text{erf} \left(\frac{j m_p^{\mathcal{D}}(\emptyset) - m_p^{\mathcal{D}}(\emptyset) + 2\sigma^{\mathcal{D}}}{(1 - m_p^{\mathcal{D}}(\emptyset))\sigma^{\mathcal{D}}\sqrt{2}} \right) m_p^{\mathcal{D}}(\emptyset) \quad (5.13)$$

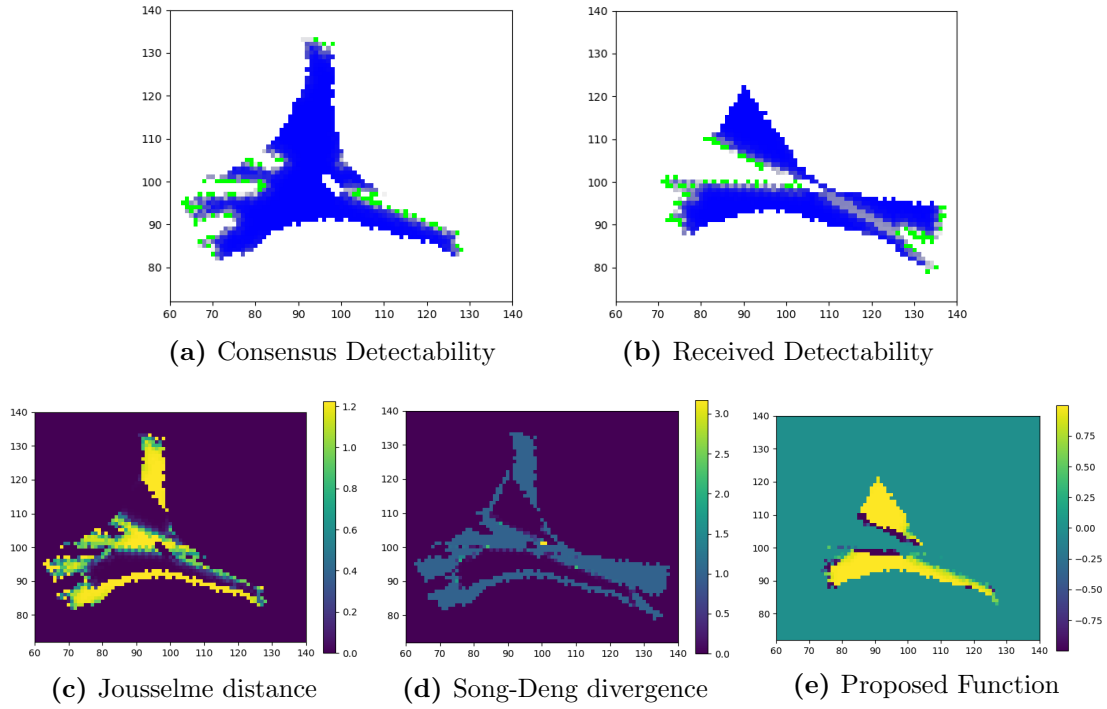


Figure 5.13: Result from several support functions. Top row are the consensus and reference detectability grids, with white pixels being unknown $\Omega^{\mathcal{D}}$, blue pixels being undetectable \emptyset and green pixels being detectable \mathcal{D} . Bottom row are the cell-wise results of applying the Jousselle distance, Song-Deng divergence and Equation (5.13) respectively.

The theoretical output of this function is illustrated in Figure 5.14 over all possible inputs. It is also illustrated in Figure 5.13e on a real case, where it can be seen that:

- Overlapping areas of same class generate positive support;
- Overlapping areas of opposite classes generate negative support;
- Non-overlapping areas do not generate either.

However, in the current definition of detectability, two sources can have opposite detectabilities and yet both be correct (see Section 4.3.2.3). For this reason, only positive support is considered. Cell-wise support is then turned into a single BBA by integrating positive support over all cells p as:

$$\tau_j^{\mathcal{D}} = \int_c D^{\Sigma}(m_p^{\mathcal{D}}, j m_p^{\mathcal{D}}) \quad (5.14)$$

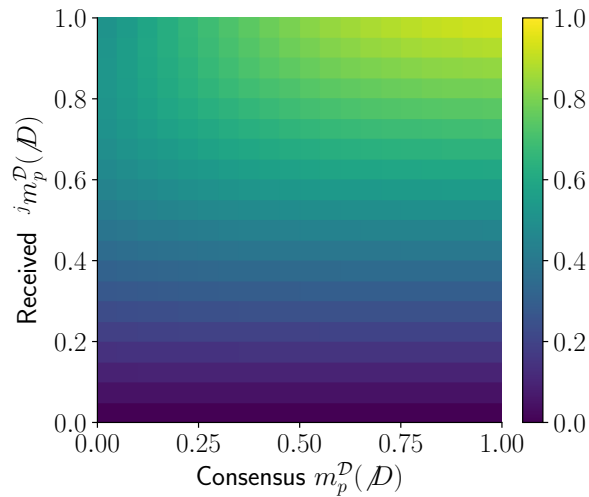


Figure 5.14: Output of the support function D^{support} for all input values and $\sigma^{\mathcal{D}} = 200$.

and using the sigmoid function Φ in reverse such that only positive trust can be estimated:

$$m_j^{\text{fsi}} = \Phi \left(-\tau_j^{\mathcal{D}}, -\delta^{\text{fsi}} - 4\sigma^{\text{fsi}}, \sigma^{\text{fsi}}, \kappa \right) \quad (5.15)$$

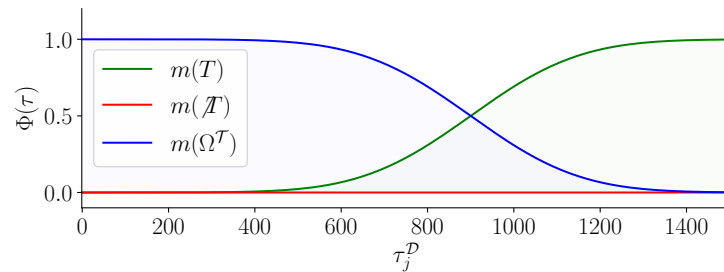


Figure 5.15: Sigmoid function used in free-space confirmation with $\delta^{\text{fsi}} = 500$ cells and $\sigma^{\text{fsi}} = 200$.

δ^{fsi} controls how much supported area is needed to confirm j and σ^{fsi} how fast to confirm, as depicted in Figure 5.15.

5.3.6 Summary of Trust Parameters

The aforementioned method to estimate trust in others is composed of many parameters. Most of them reflect a physical reality but tuning this system remains a complex task. Table 5.1 summarizes parameters used to estimate trust with their interpretation.

Table 5.1: Summary of Trust Parameters.

Notation	Domain	Description
$\Lambda_{\Delta t}$	R^{+*}	Time it takes to lose half information about trust
$1-\alpha^*$	$[0, 1]$	How much BBAs $*$ should impact upper levels
D^{\min}	$[0, 1]$	Minimal object detectability to consider objects incoherent
β^{pen}	$[0, 1]$	How much to penalize an incoherent detectability
δ^z	R^{+*}	Value at which z_o starts becoming incoherent
σ^z	R^{+*}	How quickly values of z_o above δ^z produce non-trust
δ^d	R^{+*}	Distance under which object are incoherently close
σ^d	R^{+*}	How quickly values of z_o above δ^z produce non-trust
δ^{fsi}	R^{+*}	Minimal support area to confirm detectability
σ^{fsi}	R^{+*}	How quickly areas above δ^{fsi} produce trust
ι_x	R^{+*}	x coordinates dilution of untrustworthy objects
ι_y	R^{+*}	y coordinates dilution of untrustworthy objects

5.3.7 Trust-Aware Tracking

The goal of trust is to ignore the information of untrustworthy peers in the cooperative tracker. This can be done by setting a trust threshold under which information is simply ignored. However, in some contexts, having ambiguous information is better than not having information at all, such as hidden areas where only non-trusted peers have seen. Due to this, fixing a constant threshold might be complex.

To account for this fact and in order to smooth transitions, we propose to *dilute* objects and FS based on how untrustworthy their source is, as in Figure 5.16.

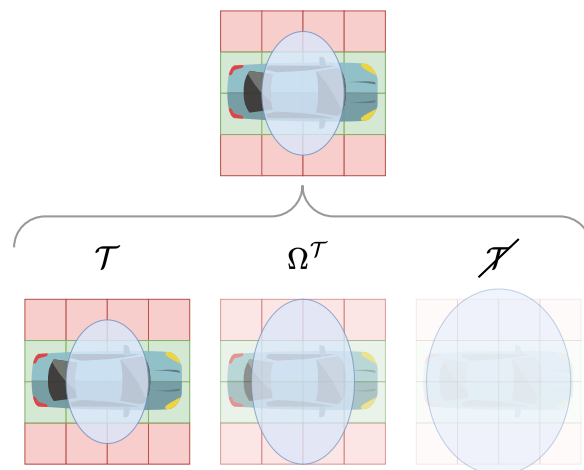


Figure 5.16: Example of perception dilution. The more untrustworthy a source is, the more transparent and enlarged objects become.

Let $\alpha \in [0, 1]$ be a scalar representing how much to dilute the perception infor-

mation received from a peer based on how trusted the peer is. Dilution is defined as discounting detectability as ${}_{\alpha}m^{\mathcal{D}}$, discounting object existences and increasing object covariances as:

$${}_{\alpha}o = \langle \mathbf{x}, \mathbf{P} + \mathbf{P}_{\alpha}, {}_{\alpha}m^{\exists}, \dots \rangle \quad (5.16)$$

where \mathbf{P}_{α} is a penalty matrix function of α and maximal penalties $\iota_x, \iota_y \dots$ of the form

$$\mathbf{P}_{\alpha} = \begin{bmatrix} \alpha\iota_x & & \dots \\ & \alpha\iota_y & \\ \vdots & & \ddots \end{bmatrix} \quad (5.17)$$

That way, untrustworthy sources have less impact on the final fusion result. Discounting the existence based on a *trust* parameter has already been proposed in several works such as (Aeberhard 2017) but that parameter was considered constant. On the other hand, increasing covariance have not yet been proposed, to our knowledge. This is done because our tracking system (GNN association then SCI update) does not account for the object existence while updating states. This means that even low-existence objects can significantly impact tracks, and to reduce that effect, object states are also *diluted*.

In the results shown afterwards, this diluting factor α is defined as the plausibility of a source being untrustworthy ($Pl^{\mathcal{T}}(\mathcal{Z})$). Other metrics could also be adapted, such as the pignistic.

5.4 Experimental Evaluation

In this section, the impact of trust and several parameters used for its estimation are analyzed. The methodology described in Section 4.4, based on an overtaking scenario, is applied on the output of v_1 's cooperative tracker. In addition, trust estimated at each time step is also recorded and is plotted as summarized in Figure 5.17. With this display, non-trust ($m^{\mathcal{T}}(\mathcal{Z})$) is plotted accumulated on top of trust ($m^{\mathcal{T}}(T)$), showing how information is distributed between 0 and 1.

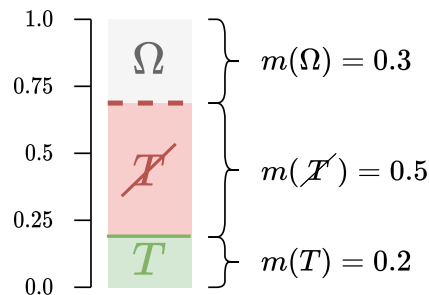


Figure 5.17: Example of trust plotting used in the following curves. Green area and curve is $m^{\mathcal{T}}(T)$, in red $m^{\mathcal{T}}(\mathcal{Z})$ and grey $m^{\mathcal{T}}(\Omega^{\mathcal{T}})$.

Although no dedicated study has been conducted, a step of trust estimation takes in the order of 1 s to complete, which is slower than what the cooperative perception standard specifies (exchanges between 1 and 10 Hz). While curves shown

afterwards were computed off-line with 10 Hz data, trust estimation qualitatively runs at a maximum of 1 Hz on current computers. This is not necessarily an issue though, considering that trust can be estimated asynchronously from data exchange and object tracking.

In the following evaluation all vehicles start with an initial value of trust of 0.8 to ensure quick convergence.

5.4.1 Added Value of Trust in Nominal Cases

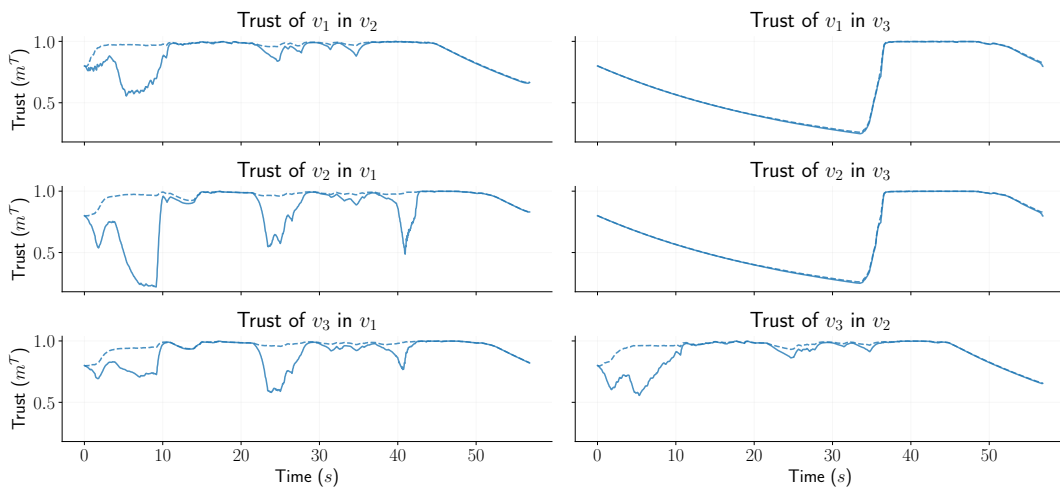
In this first analysis, the behavior of trust and its impact on perception is studied in the nominal case, that is to say without the addition of faulty objects.

In Table 5.2 and Figure 5.18, one can see an improvement in the detection of objects and free space, with a gain in Area Under Curves (AUCs) of 0.03 and 0.05 respectively. This is also visible with the orange curve being slightly over the blue one in Figure 5.18c and going slightly further on the FPR axis in Figure 5.18d. The accuracy and consistency on the other hand are degraded, possibly due to the averaging effect described in Section 4.4.2 or to the dilution process described in Section 5.3.7 that can lead objects to be less accurate.

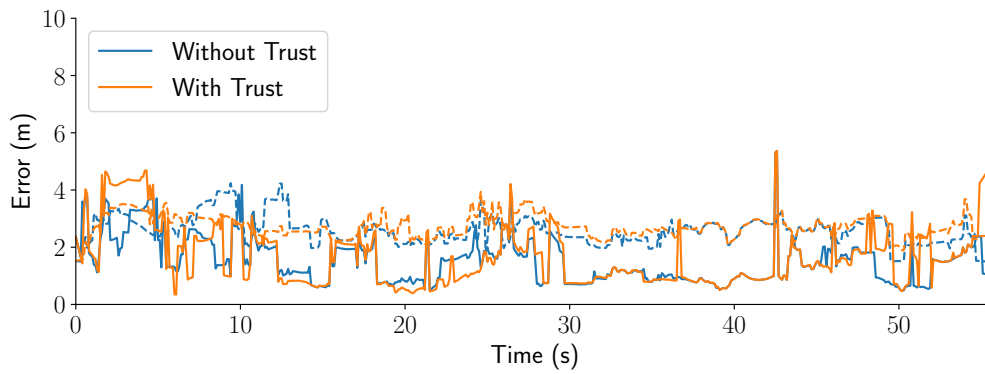
In addition, one can see in Figure 5.18a that trust is relatively constant over time for v_1 and v_2 that partially share FoV. Three negative spikes can nevertheless be seen at $t = 7$, $t = 22$ and $t = 40$. They are due to naturally occurring tracking errors that lead false objects to be created on v_1 . Trust thus effectively detected these faults and reduced their impacts. v_3 , which is on the other side of the road between $t = 2$ and $t = 30$ can trust v_1 and v_2 without sharing FoV with them as they both start with a high trust and confirm each other in the eyes of v_3 afterwards. On the other hand, v_1 and v_2 cannot trust v_3 and progressively lose information until they start sharing FoV at $t = 30$. Note though that estimating distrust does not requires sharing points of view, as can be seen in the estimation of v_3 about v_1 that still contains the negative spikes.

Table 5.2: Comparison with and without trust on a nominal situation for vehicle v_1 .

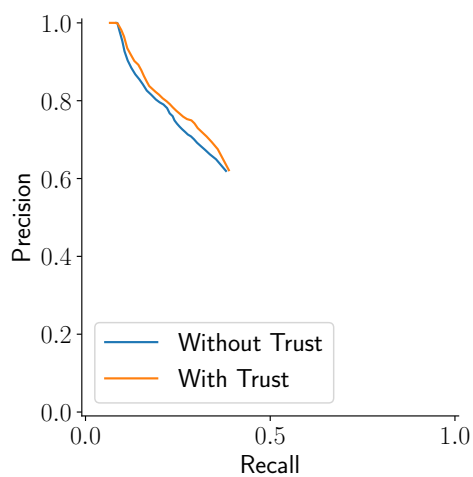
	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Without Trust	1.52	0.86	0.90	0.47	0.23	0.61
With Trust	1.66	0.91	0.83	0.48	0.26	0.66



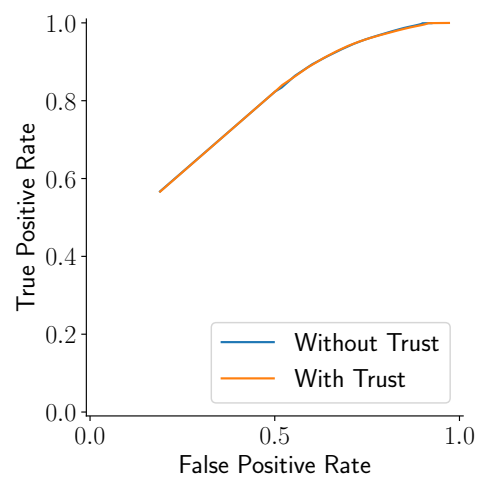
(a) Evolution of trust (continuous lines) and distrust (dashed lines) over time.



(b) Object 2D errors (continuous lines) and covariances (dashed lines)



(c) Object existence PR Curves



(d) Free-Space ROC Curves

Figure 5.18: Comparison without trust (blue curves) and with trust (orange curves) on a nominal situation for vehicle v_1 .

5.4.2 Trust Estimation and Perception Performance in Case of Faults

In this analysis, the behavior of trust and its impact on perception is studied in the case of faulty information. For this, hybrid data combining real perception data and simulated objects and errors have been used. More specifically, faults have been artificially added to the output of v_2 standalone tracker by

- Removing detected objects on certain parts of the road;
- Adding ghosts objects that follow the road with constant speed where there are no real objects;
- Adding white noise to some tracked objects poses and sizes.

This is indicated at various points in time denoted by grey bars in the following curves. v_2 is thus not impacted by these faults, only v_1 and v_3 are.

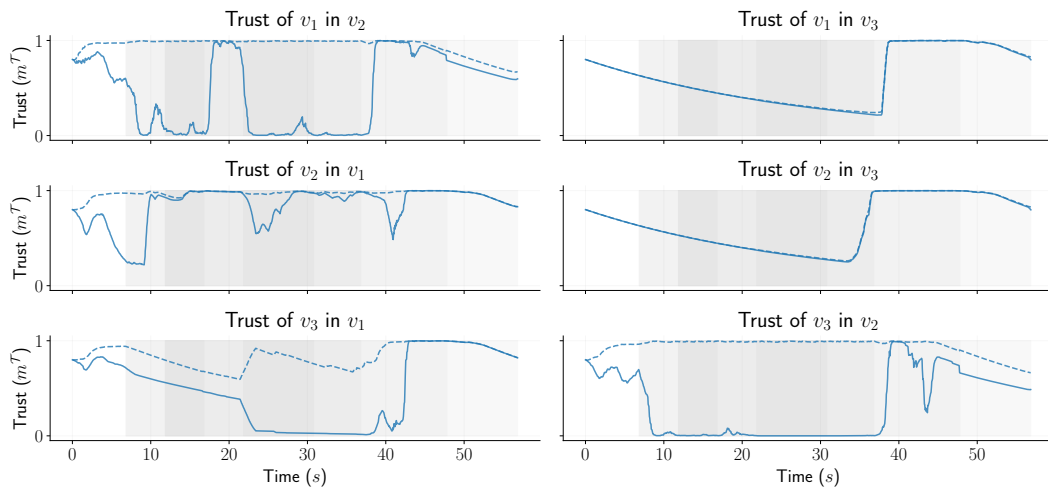
Similarly to the previous study, an improvement in object and free space detection can be observed in Table 5.3 and Figure 5.19, with a gain in Area Under Curves (AUCs) of 0.03 and 0.04 respectively. Object accuracy and consistency also seems degraded.

However, looking at Figure 5.19a, one can see untrustworthiness is correctly estimated when errors are added. In particular, v_3 distrusts v_2 until they start sharing FoV and faults get lighter. The same pattern can be observed in the distrust of v_1 in v_2 that is strong when faults are numerous at $t = 10$ but decreases when faults are scarcer at $t = 20$.

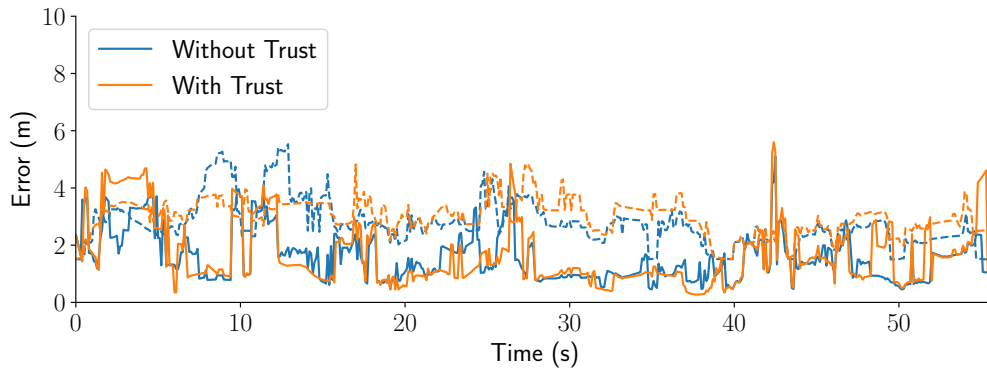
Table 5.3: Comparison with and without trust on a faulty situation on vehicle v_1 .

	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Without Trust	1.57	0.91	0.87	0.42	0.20	0.61
With Trust	1.67	0.99	0.83	0.43	0.23	0.65

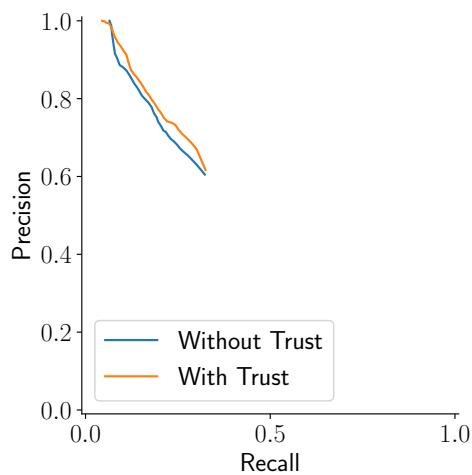
Another interesting property to notice is the impact of one's fault on trust estimated about others. To study this effect, trusts in the presence and absence of faults are conjointly plotted in Figure 5.20a. In this plot, one can see that trusts is similar until faults are added at $t = 7$, at which point the trust of v_1 and v_3 about v_2 starts diverging, while trusts of v_2 are not impacted in this study. However, notice how the trust of v_1 and v_3 about each other change, due to their consensus being different. Indeed, because v_1 stops listening to v_2 , it cannot trust v_3 before seeing it itself. This effect is even more visible in the trust v_3 estimates about v_1 , where it initially distrusts v_1 because v_1 's perception does not match its consensus, solely composed of v_2 's perception at that point in space. v_3 rapidly understands that v_2 is the one at fault and stops listening to it at $t = 10$. From that point and until $t = 40$, v_3 cannot confirm v_1 because there is no other redundant and trusted source.



(a) Evolution of trust (continuous lines) and distrust (dashed lines) over time. Grey bars are when faults are added. The more opaque, the more faults were added at the same time.



(b) Object 2D errors (continuous lines) and covariances (dashed lines)



(c) Object existence PR Curves



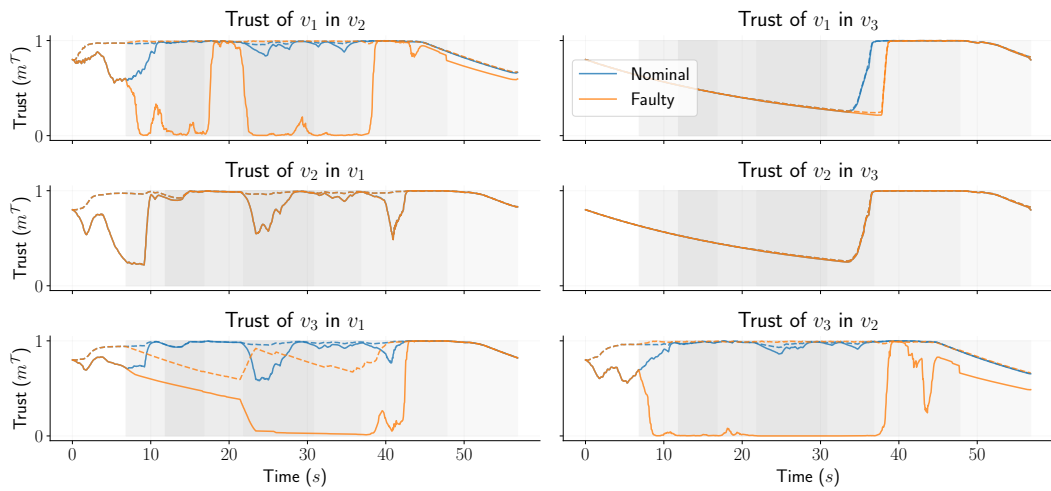
(d) Free-Space ROC Curves

Figure 5.19: Comparison without trust (blue curves) and with trust (orange curves) on a faulty situation for vehicle v_1 .

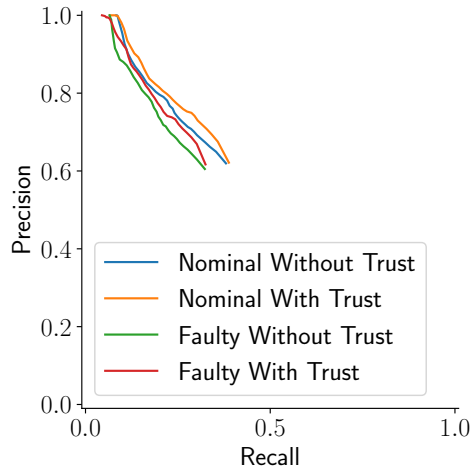
Additionally, Figure 5.20b and Table 5.4 illustrate that in terms of object and free space detection, trust is efficient at bringing faulty cooperation closer to the nominal situation. This is visible in particular with the *nominal without trust* and *faulty with trust* variants that share the same object-AUC.

Table 5.4: Comparison with and without trust on faulty and nominal situations for vehicle v_1 .

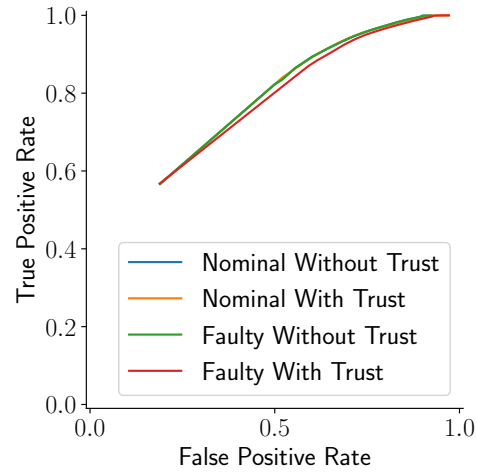
	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Nominal w/o Trust	1.52	0.86	0.90	0.47	0.23	0.61
Nominal w/ Trust	1.66	0.91	0.83	0.48	0.26	0.66
Faulty w/o Trust	1.57	0.91	0.87	0.42	0.20	0.61
Faulty w/ Trust	1.67	0.99	0.83	0.43	0.23	0.65



(a) Evolution of trust (continuous lines) and distrust (dashed lines) over time. Grey bars are when faults are added. The more opaque, the more faults were added at the same time.



(b) Object existence PR Curves



(c) Free-Space ROC Curves

Figure 5.20: Comparison with and without trust on faulty and nominal situations for vehicle v_1 .

5.4.3 Impact of Trust Parameters

Finally, in order to evaluate the impact of several parameters used in the estimation of trust, the fact that trust is expressed in the form of a tree can be used. Figure 5.21 is a tree in the same form as Figure 5.3 but provides the values at each nodes over time. Hence, the contribution of each node to the final estimate can be analyzed and explained separately.

In Figure 5.21, one can see that the main sources of distrust from v_1 towards v_2 is due to consistency (`cons`), attribute coherency (`cohe-atc`) and dissimilarities between consensus and received objects (`conf-odi-rc`). Distrust is thus estimated despite the creation of trust in the object similarity node (`conf-osi`). On the other hand, valleys in trust that were visible in Figure 5.18a can be explained by looking at the bottom Basic Belief Assignments (BBAs) of Figure 5.22 that

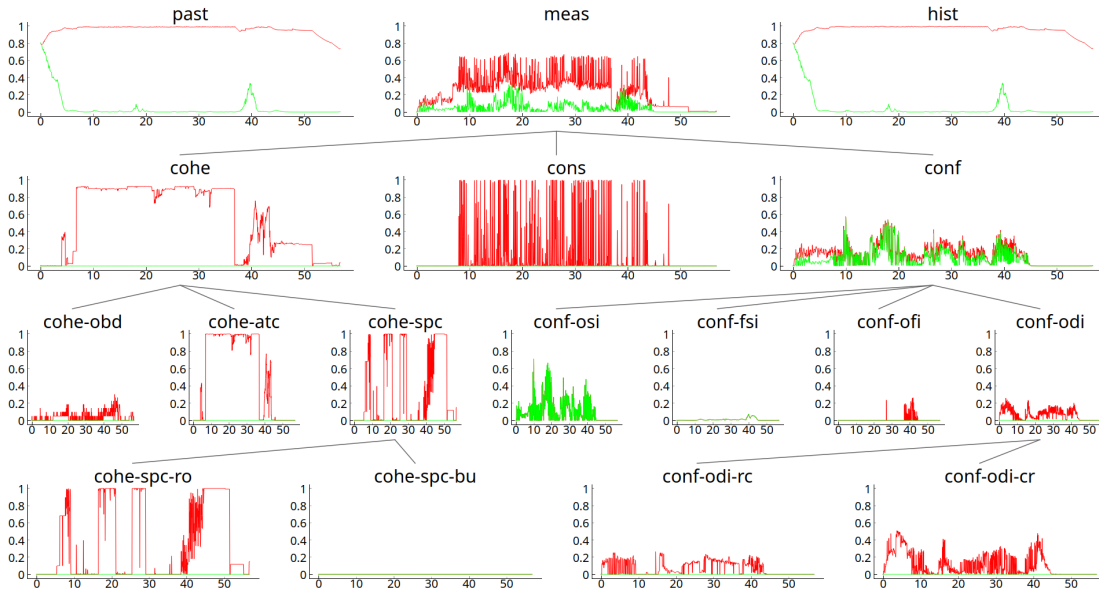


Figure 5.21: Evolution of BBAs in the estimation of trust over time in the form of a tree. Trust estimated by v_1 on v_2 in the case where v_2 is adding faults. Despite trust (green curves) being created in *conf-osi* and *conf-fsi*, non-trust (red curves accumulated on top of trust) takes precedence.

generate strong distrust at those moments ($t = 7$, $t = 20$ and $t = 40$).

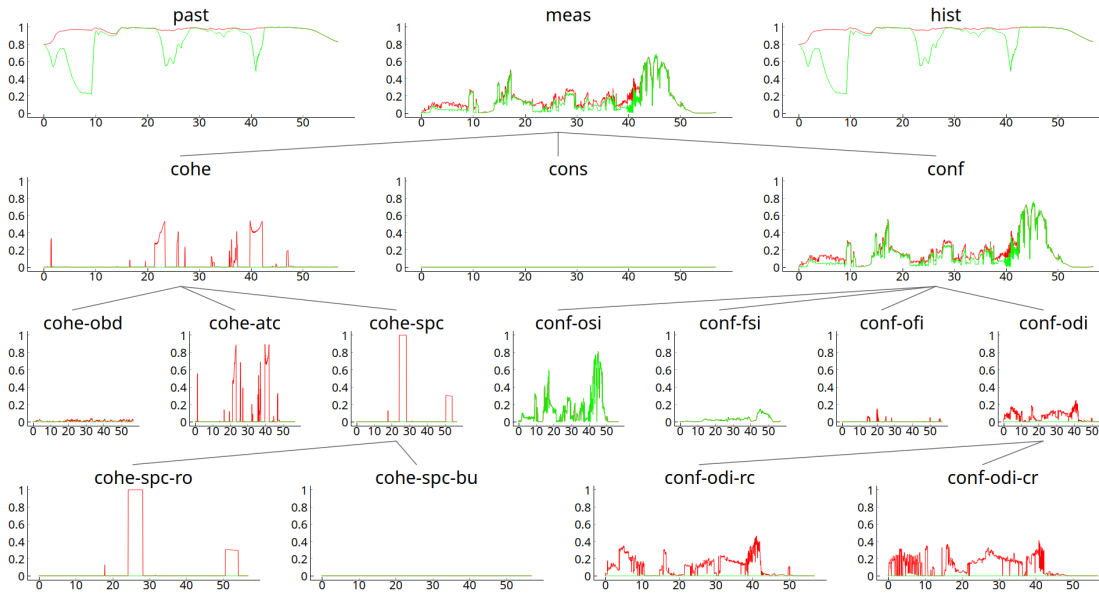


Figure 5.22: Evolution of BBAs in the estimation of trust over time in the form of a tree. Trust estimated by v_2 on v_1 in the the nominal case. Notice how distrust (red curves) estimated in leaves (*cohe-spc-ro* and *conf-odi-rc*) rise to the top to create valleys in trust (green curves) in *hist*.

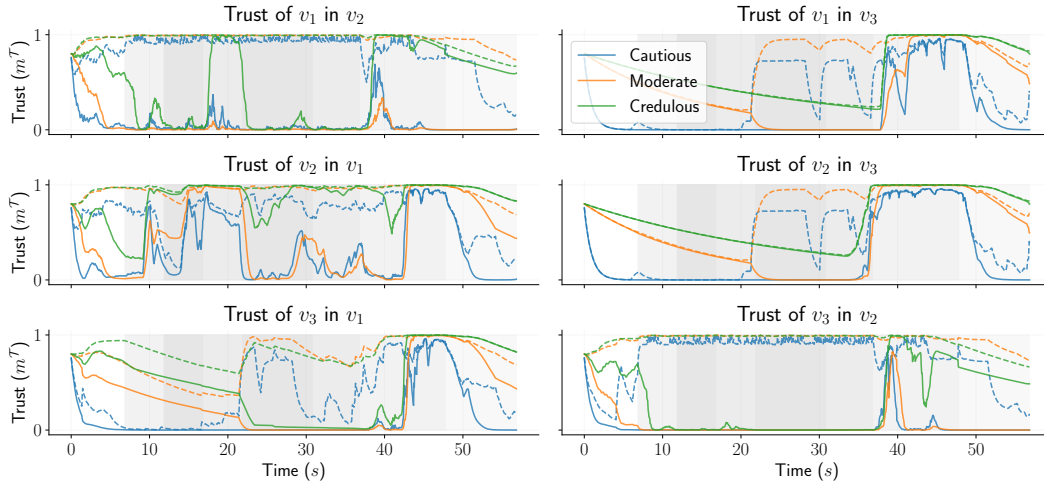
With this in mind, three parameter sets have been tested and are compared to further illustrate the behavior of trust estimation. Those sets are given in Table 5.5 with respective goals behind them:

- *Cautious*, with high negative weights, low positive weights and strong forgetting factors. This variant illustrates cautious trust systems that prefer ignoring others rather than risking the integration of erroneous information;
- *Credulous*, with low negative weights, high positive weights and weak forgetting factors. This variant illustrates confident trust systems that are quick to trust others at the expense of a reduced safety margin;
- *Moderate*, with middle-range weights for everything.

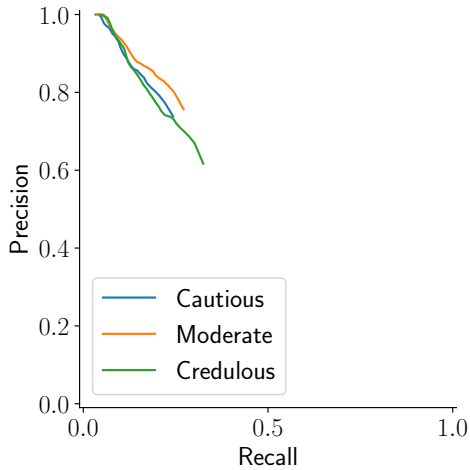
Table 5.5: Sets of parameters used in the following comparison.

Parameter	Cautious	Moderate	Credulous
α^{past}	0.95	1.0	1.0
α^{cohe}	0.35	0.25	0.15
$\alpha^{\text{cohe,atc}}$	0.9	0.9	0.6
$\alpha^{\text{cohe,obd}}$	0.2	0.1	0.1
$\alpha^{\text{cohe,spc}}$	0.35	0.25	0
$\alpha^{\text{cohe,spc,bu}}$	1.0	1.0	0
$\alpha^{\text{cohe,spc,ro}}$	1.0	1.0	0
α^{cons}	0.75	0.5	0.4
α^{conf}	0.9	0.75	0.9
$\alpha^{\text{conf,osi}}$	0.8	0.8	0.9
$\alpha^{\text{conf,fsi}}$	0.8	0.8	0.9
$\alpha^{\text{conf,ofi}}$	0.2	0	0
$\alpha^{\text{conf,odi}}$	0.9	0.8	0.6
$\alpha^{\text{conf,odi,rc}}$	0.6	0.5	0.4
$\alpha^{\text{conf,odi,cr}}$	0.9	0.3	0.2
$\Lambda_{\Delta t}$	3	5	10
β^{pen}	0.1	0.05	0.01

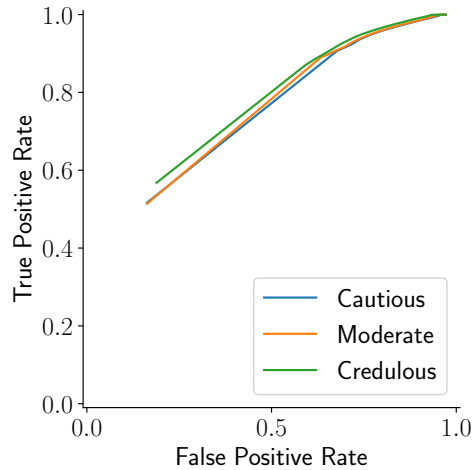
In Figure 5.23 and Table 5.6, one can see that the credulous tuning improves its object detectability by more easily trusting others. It however comes at the expense of degrading object accuracies as more low-quality objects are integrated. In Figure 5.23a, the cause is this effect is clearly visible. As the credulous estimate is slow to lose information and fast to gain positive information, it is often more optimistic than the other two variants, even when not relevant such as at time $t = 19$. On the other hand, the cautious estimate is almost always distrustful, which might lead to pessimism during cooperation. Indeed, if the only moments where sources are trusted is when a large FoV is shared with them, the benefit of cooperation is reduced.



(a) Evolution of trust (continuous lines) and distrust (dashed lines) over time.



(b) Object existence PR Curves



(c) Free-Space ROC Curves

Figure 5.23: Comparison of several parameters tuning under erroneous information on vehicle v_1 : cautious in blue, moderate in orange and credulous in green.

Table 5.6: Comparison of several parameters tuning under erroneous information for v_1 .

	Mean RMSE (m)	Mean Std (m)	Consistency (%)	Obj Max F1	Obj AUC	FS AUC
Cautious	1.34	0.92	0.88	0.37	0.18	0.65
Moderate	1.36	0.97	0.88	0.40	0.21	0.65
Credulous	1.67	0.99	0.83	0.43	0.23	0.65

Other parameters such as the combination rule or consensus building mode have been studied. They are presented in Appendices D and E as they did not show significant differences with results presented previously.

5.5 Conclusion

In this chapter, we have introduced and studied the concept of *trust* in an open network where vehicles interact for a few moments without knowing a priori if they can trust others. The method we have proposed allows, from a certain point of view, to combine fault detection and confirmation methods in a novel tree representation using belief functions. This allows for a finer representation of uncertainty, leaving more information to downstream decision modules. Additionally, representing sources of trust or distrust as an evidential tree allows for an easier explainability.

Conducted studies illustrate that trust can be effective at removing information from erroneous peers while keeping information from valid peers. Detection, tracking or localization errors are thus naturally handled by our formulation. Although it is composed of numerous parameters, making it hard to tune, we have shown that parameters mostly follow physical or logical rules that makes them interpretable.

The main limitation of trust is that peers are trusted based on how similar their perception is to the truster's. This assumes that the truster's perception is faithful to the real world, hence any matching perception must be faithful as well. However, in practice an attacker could simply repeat the perception it received and be trusted. This issue might be a physical limitation but several solutions can be implemented to temper it. For example, imitating how map or dictionary makers detect plagiarism, faults could intentionally be added to verify if they are repeated.

Other aspects could also be studied further. In particular, the initial trust is currently set to the same value for all peers, it might be interesting to vary it based on the peer type (i.e. trust infrastructure or public services more at first) or even based on reputation. Considering that peers also communicate their trust, a combination of received trust could be used as a prior. In addition, trust could be used to *dilute* objects in a more advanced manner. The current method is relatively naive but using trust in conjunction with a JPDA would be more adapted to smooth transitions between trusted or distrusted in trackers.

Finally, the application of trust to local sensors could be studied further. As illustrated in the nominal situation, trust improves perception even without the addition of faults and works as a sort of error detector for local sensors. Trust could probably be used as a *reliability* factor even on standalone-multi-PoV systems.

Chapter 6

General Conclusion

6.1 Conclusion

In this manuscript, we have studied cooperative perception in the context of intelligent vehicles navigating on open roads. By exchanging perception, intelligent vehicles can better understand their surrounding, especially beyond the range of their on-board sensors, and make safer and more efficient decisions.

As discussed, a decentralized approach in which each vehicle takes care of building its perception information on its own is the most relevant approach. It favors the scalability of this technology while ensuring that privacy is preserved. However, the decentralized nature of these exchanges raises several problems, one of the most delicate being the management of trust in other peers that transmit their perception information. As a consequence, controlling the information integrity (either for autonomous navigation or driver assistance) is a more complex task.

Representing cooperative peers as remote sensors is an effective solution to the decentralized cooperative perception problem, provided that exchanged information is well defined and that resilient data fusion tools are used. The basic idea when approaching this question is to exchange perceived objects after having localized them in a global frame of reference thanks to localization information but, very quickly, the exchange of free spaces appears as an essential information. In this work, we have extended this concept with evidential detectability grids that manage the field of view and quantify the information in these areas.

In this work, much effort has been made to conduct real experiments with equipped vehicles on open roads. Although additional experiments should be conducted to confirm them, this allows us to draw some interesting conclusions.

Firstly, the fusion of multiple cooperative points of view has a significant positive impact on object detection rates, which clearly shows that cooperative perception allows vehicles to perceive the environment further than they can with their own sensors. We believe the same applies to free space detection but our results showed marginal differences that we believe are due to limitations in the evaluation methodology. In order to obtain good results, the field of view and the measured free space of each source must be clearly represented.

Secondly, the added-value of cooperation for increasing the quality of object states estimation (in terms of position and speed for instance) is not clear. Due to the combined effect of perception errors, localization errors, out-of-sequence observations caused by communication latency, objects estimated by external sources do not improve the overall accuracy of objects perceived locally.

For the reasons cited above, we can conclude that the integrity of perception can be affected both positively and negatively by cooperative perception. Our results show that cooperation improves perception completeness. However, regarding correctness (i.e. accuracy and consistency) fusing information from remote sources in a decentralized manner has almost no positive impact. This is because decentralized data fusion methods have to be resilient and cautious, which reduces the contribution of received information compared to on-board sensor for dynamic objects state estimation. Finally, the integrity of cooperative perception can be significantly perturbed by data from erroneous peers, whether intentional or not. The trust management method that we have proposed is able to quickly detect and ignore such peers, which is essential to deal with highly dynamic information.

6.2 Contributions

Decentralized data fusion for tracking and state estimation introduces information loops in classical data fusion schemes. To address this issue, we have studied Kalman-based estimation and covariance intersection methods. The use of the *split covariance intersection filter* in the context of cooperative perception is particularly relevant. We have derived interpretations and proposed tuning methodologies for the filter.

The data fusion of tracking systems across different points of view requires a particular care. We have proposed *evidential detectability grids* as a tool to combine multiple free space measurements and to track objects across fields of view. The proposed architecture can fuse perceptions from on-board sensors and cooperative peer in a generic manner. This allows to manage the existence of tracks with belief functions.

Finally, as peers are outside of our sphere of control, we have proposed a system to estimate the *trust* about other peers. It is computed through belief functions and an evidential network that performs fault detection and confirmation through redundancy. Thanks to this estimation of trust, unreliable peers are ignored in the perception task.

This work has been evaluated with real data. Datasets were recorded and all the necessary perception algorithms were implemented in the context of this PhD. The developed perception algorithms have also been used in other works, such as the Tornado demonstration in July 2021 and other research work. Finally, we have proposed a global evaluation methodology using ground truth that was carried out by hand.

6.3 Perspectives

Several aspects of the proposed approaches remain to study.

A global evaluation metric summarizing the three aspects of perception (object and free space detection rates and object state accuracy and consistency) could be derived. It would be interesting to compare several perception systems and different strategies for tuning parameters in a consistent and global manner.

Other object tracking methods could also be studied. A joint state formulation could also improve the state estimation performance. Corner-based tracking might also provide better performance than the current bounding-box-based tracking. In addition, trust could be integrated in state filtering more tightly, for example by modifying the state update equations to add a weighting factor on the residual based on how trusted its source is.

It could also be interesting to study the use of cooperative perception to reduce the number of sensors necessary to ensure the integrity of intelligent navigation.

The methods described in this manuscript use significant amounts of parameters to work. This has been done to avoid having *hidden parameters*, and to clearly expose all the key parameters. While this makes the overall system difficult to tune and understand for a novice or non-expert, this opens the possibility to apply semi-automatic tuning methods. Gradient descent or similar methods could be used in conjunction with the global metric introduced in the previous paragraph to automatically find the best parameter sets across various experimental situations.

The trust estimation scheme proposed in this work has only been applied to remote peers. It could be interesting, for the sake of generality, to apply it the output of on-board sensors. Based on the added value of trust seen in the nominal case, we believe that applying trust to on-board sensors could also improve the standalone perception capabilities of a vehicle. This could also be used to detect faulty sensors in trustworthy network sensors. However, more research is needed to study the capacity of trust to detect and isolate sources that have minor errors. Indeed, tuning the sensibility of trust to detect slight errors would certainly result in pessimistic estimates. While pessimistic trust estimation is not an issue in cooperative perception (as vehicle are independently capable of navigating), it might be in more critical systems.

In this work, we showed that trust is able to detect errors in both nominal and malicious situations. However, certifying that it will always be able to do so is not necessarily possible. New types of faults could be found, subtle enough to remain hidden but impactful to navigation systems.

Finally, a proper evaluation of the integrity remains to be done. In this work, we illustrated the relative gain in terms of performance of our methods. We have not shown the absolute capacities of our perception system to provide non-misleading, complete and available information to navigation modules. For this, several navigation use cases should be derived in order to evaluate perception against task-specific ground truths.

As a more general note to these perspectives, we believe that more research needs to be conducted on the joint representation of dense and predictable information for the integrity of perception. In this work, we used separate representations for objects and free space then linked the two afterwards (e.g. tracked objects stop existing inside free space, coherency checks of the two). A proper joint representation would simplify these aspects and make its evaluation simpler.

List of Figures

1.1	Architecture used at Ulm University, adapted from (Taş et al. 2016). Modules are regrouped in three main categories: input (sensors), processing (perception, scene understanding) and output (navigation, control). In the perception modules, note that objects and space are both estimated.	12
1.2	Architecture used at Renault during the Tornado project, adapted from (Milanés et al. 2022). The same main categories are present, but note the presence of communication with connected infrastructure and cloud connection.	13
1.3	Scene composed of two vehicles and a pedestrian. Perceived objects and space considered free by the blue vehicle at the bottom are depicted in red and green respectively. This example also illustrates that there are always hidden areas in a field of perception.	14
1.4	Common structures for communication and computation between multiple peers. Small blue nodes are sources of information. Orange nodes are where computations are realized. Small orange nodes are both information sources and computers.	14
1.5	Lateral position error and bounds of a localization system through time. Taken from (Lima, Welte, et al. 2020).	16
1.6	Simplified Stanford diagram and example of usage for evaluating a localization system taken from (Gottschalg et al. 2020). Dotted lines represent the Alert Limit.	17
1.7	Misleading perception cases in the same situation as Figure 1.3. The road, real objects and their motion are displayed in the background. Localization error and estimated motion in red. The space perceived as free is in green.	18
2.1	Information loop introduced between three vehicles. Follow the blue arrow from the top that represents the initial piece of information sent by the blue vehicle. It is augmented by the information of the red and green vehicle then comes back to the blue vehicle. This means that the blue vehicle will receive its own information, potentially thinking that the information is new.	22
2.2	Sets composing $\Omega = \{\omega_1, \omega_2, \omega_3\}$ and subset $A = \{\omega_2, \omega_3\}$	23
2.3	Subsets used when computing $Bel(\{\omega_2, \omega_3\})$ in (a), $Pl(\{\omega_2\})$ in (b) and $BetP(\{\omega_2\})$ in (c).	25
2.4	Gaussian probability distribution and 3σ domain of a one-dimensional object with a mean $s = 3$ and standard deviation $\sigma_s = 0.5$	30

2.5	2D Gaussian probability distribution and 3σ bound of a two dimensional object with $x = 1$, $y = 5$, $\sigma_x = 2$, $\sigma_s = 3$ and $\eta = 0.4$	31
2.6	2D Object with its current and predicted state. Covariance is not represented.	33
2.7	Update of a 2D object. Blue is the predicted state, green the observation and red the updated state. Covariance is not represented. Observation error exaggerated for clarity.	34
2.8	Covariance ellipsoid resulting from a CI fusion for multiple values of ω	37
2.9	Comparison of Kalman update with CI over three situations: a baseline in which inputs are orthogonal and centred in (a), an edge case in which inputs are orthogonal but non centred in (b) and an edge case in which inputs are aligned and non centred in (c).	37
2.10	Comparison of CI, IFCI, ICI and ITCI over the three previous situations.	39
2.11	Situation composed of three vehicles in one-dimension.	40
2.12	Estimation errors and $\pm 3\sigma$ uncertainty bounds. Estimate of the KF in blue, CIF in red and K-CIF in green. In (a) and (c), estimates of the KF and K-CIF are the same.	42
2.13	Illustration of CI in a simple one dimension case using state interval representation of Figure 2.4. A blue predicted state is fused with an orange observation, resulting in the green fusion. It is different from the intuition one might have about the "intersection" of intervals represented in purple.	44
2.14	Curve of the fused covariance matrix determinant for ω between from 0 to 1 in the case of a partial measurement. The region between 0.25 and 1 is zoomed in.	45
2.15	Fusion of a state (blue) and partial measurement (orange) for different values of ω . The optimal ω in the sense of Equation (2.30) is denoted <i>auto</i> and equals 0.99.	45
2.16	Estimation errors and $\pm 3\sigma$ confidence domains. KF+CIF in green and SCIF in red. Continuous lines are the total confidence domains and dashed the independent part. In (a), curves are superimposed.	46
3.1	Several successive poses of a LiDAR head as it rotates while the vehicle moves.	52
3.2	point-clouds accumulated over 5 seconds of moving in a roundabout. The red point-cloud shows the distortion caused by the vehicle motion, and the blue point-cloud are their undistorted counterpart. Note the car and sign duplication caused by the varying point of view and the fuzziness of the left and right fences.	53
3.3	Reference point-cloud in red. $P^{\setminus rz}$ filtered point-cloud in blue. Points are cut above the sensor, below the ground and further than 150 m to limit the area of focus. Both are accumulated over a second for the sake of clarity.	54
3.4	Original point-cloud in red and intensity filtered point-cloud in blue. Notice how signs on the right are highlighted while those on the left are not. This is because only one side of road signs is reflective.	54

3.5	Reference point-cloud in red, ground point-cloud filtered with the method of (Jiménez et al. 2021) in blue.	55
3.6	Reference point-cloud in red and clustered point on top, where different colors mean different clusters.	56
3.7	Accumulated reference point-cloud in red. Snapshot bounding boxes are fitted over each cluster using Algorithm 1 . Only objects close to the road are shown here. One can notice the bad alignment of the green bounding box here because the vehicle is partially seen from the side.	57
3.8	Bird eye view of a particular cluster point-cloud, heading and bounding boxes. The ideal heading and bounding boxes differs from what the PCA yields due to the principal component being across the car width and length. This issue is worse when a car is only one side is seen.	57
3.9	Image with bounding boxes classified by YOLOv5. Results obtained within the Tornado project taking place in the city of Rambouillet.	58
3.10	Accumulated points classified by Cylinder3D. In pink are point classified as cars, road in cyan, signs in red, vegetation in orange and infrastructure in green. Results obtained in the city of Compiègne.	59
3.11	Point-cloud and measured free space polygon. As the polygon is computed from the vehicle position, the vehicle is depicted in grey.	59
3.12	Object existence stages in track management (from (Aeberhard 2017)). Four zones are defined: scheduled for deletion in red, unconfirmed in orange and confirmed in green. The yellow zone is an hysteresis to let object that fell below confirmed to still be maintained.	64
3.13	Types of information usually found in perception ground truth images. Taken from (Lin et al. 2014).	65
3.14	Illustration of terms associated to the evaluation of binary classifiers.	65
3.15	Comparison of four models using ROC and PR curves, with the best model being dark blue and worst being red.	67
3.16	Example of IoU.	67
3.17	Principle of HOTA $\mathcal{A}(c)$ calculation, taken from (Luiten et al. 2021). Here two tracks (grey and black) are associated for each time step with their ground truth (respectively light and dark blue). Starting from the point indicated by red and orange arrows, the number of correct or incorrect associations is accumulated.	69
3.18	The three Renault Zoe used at the Heudiasyc laboratory.	70
3.19	GNSS-based sensors used at the Heudiasyc laboratory.	70
3.20	Simplified view of a LiDAR firing a laser beam.	71
3.21	LiDAR sensors and cameras used in this research.	72
3.22	Example of point-clouds produced by previously mentioned LiDARs. Colors represent the material reflectivity from purple absorbing to red highly reflective. Notice how the point-cloud is composed of rings stretching outwards from the origin, how dense the Pandora is compared to the VLP32C and the VLS128 to the Pandora.	72
3.23	Images returned by the Hesai Pandora cameras.	73
3.24	LiDAR processing pipeline.	74

3.25	Trajectory and data used in sign detection. In blue are the road borders described in an HD map, in red are the ground-truth road-signs and in green the car trajectory.	79
3.26	Ego-aligned association error between LiDAR/Mobileye road-sign detection and ground-truth.	79
3.27	Scene and trajectories used in car detection. In grey are the road borders described in the HD map. The three vehicle trajectories are plotted in blue, green and red.	80
3.28	Ego-aligned association error between LiDAR/Mobileye car detection and ground-truth.	80
3.29	Size error between LiDAR/Mobileye car detection and ground-truth.	81
4.1	Summary of ETSI messages between two communicating cars.	85
4.2	Summary of fields in the Collective Perception Message. Adapted from (Ansari et al. 2021).	86
4.3	Generation rules defined by (Thandavarayan et al. 2019) for sending an object or not.	86
4.4	Example of Public Key Infrastructure (PKI). Adapted from (Danquah et al. 2020).	87
4.5	Illustration of early to late fusion in sensor processing.	88
4.6	Fusion architectures used in a) Rauch et al. 2012, b) (Allig and Wanielik 2019a) and c) (Günther et al. 2016).	89
4.7	Four main frames of reference used in terrestrial navigation. Arrows represent the transformation from one frame to the next. A detected point \mathbf{x} is referenced here in the sensor frame.	90
4.8	Simple asynchronous tracking with two sensors.	92
4.9	Example of out-of-sequence observations with two sensors	93
4.10	OOS integration using observation forward prediction.	93
4.11	OOS integration using reprocess	94
4.12	OOS integration using retrodiction	94
4.13	OOS integration using FPDF	95
4.14	Ground truth labels over raw point clouds from (Busch et al. 2022).	95
4.15	Multi-Vehicle (or <i>peer</i>) architecture used in this manuscript. Sensors are all tracked independently before being fused in both a standalone and cooperative tracker. Another peer sends its standalone tracks which are fused in the cooperative tracker.	96
4.16	Persistence or detectability probabilities based on sensor range and opening (a, from (Aeberhard et al. 2011)) and obstacles (b, from (Allig, Leinmüller, et al. 2019)).	97
4.17	Steps to build the detectability of a sensor j	97
4.18	Example of detectability in one dimension with ${}^j p_0 = 0$, $F^{\text{free}} = 5$, $F^{\text{dete}} = 15$, $\iota = 0.1$ and $\kappa = 8$	99

4.19	Real driving situation and example of detectability grid computed using LiDAR data. The LiDAR is attached to the grey car in the center of both images. Red cells correspond to space measured as free by the sensor, green cells to its field of view and white cells to unobserved or ambiguous areas. Notice that the red area is naturally stopped by dynamic objects (a red car in front of the grey car and a van behind it) and that the green area is limited by static objects (buildings on the grey car right side). In addition, notice how the intensity of green and red cells fades with distance and near the free space border.	100
4.20	Example of naive detectability grid combination. Green is detectability, red is non-detectability (free) and blue conflict.	101
4.21	Sigmoids of two different mass functions.	103
4.22	Schematic representation of the sequence of events of the overtaking scenario.	110
4.23	Example of ground truth generation for a given time step.	111
4.24	Sets used in the association process of object evaluation.	112
4.25	Error and uncertainty computation of two estimates i and k when compared to the ground truth j . Here, k and i are at the same distance of j , but only i is consistent (within σ_j^i). k is not because its error is higher than σ_j^k	112
4.26	Comparison of standalone and cooperative perception for v_1 . Blue curves are standalone perception and orange cooperative.	115
4.27	Comparison of standalone and cooperative perception for v_2 . Blue curves are standalone perception and orange cooperative.	116
4.28	Comparison of standalone and cooperative perception for v_3 . Blue curves are standalone perception and orange cooperative.	117
4.29	Comparison of no detectability (blue curves), cooperative detectability (orange curves) and full detectability (green curves) for an arbitrary vehicle v_1	119
5.1	Validation situations in (Allig, Leinmüller, et al. 2019). In (a), i trusts j because they each observe a third object k . In (b), i trusts j because it observed a verified object k	123
5.2	Trust augmented multi-peer data fusion architecture. Peer perception go through a trust module before being fused in the cooperative tracking module.	124
5.3	Evidential tree for trust estimation. Trust of BBAs on the bottom, their combination in the middle and the resulting filtering for time t on the top. Red arrows only convey untrustworthiness, green trustworthiness, and orange both.	125
5.4	Received object and detectability grid. Green is detectable, red undetectable and grey unknown. On the left is a normal situation, a car is fully included in the detectable and nothing is outside. On the right is an abnormal situation with cars outside of the detectable areas, which can be a track spoofing.	126

5.5	Received car. On the left its length is normal (within predefined ranges) while on the right its length is abnormal (too small or large). This can be track spoofing.	127
5.6	Sigmoid used for the detection of car length faults. $\sigma^l = 0.5$ and $\delta^l = 5$.	127
5.7	Sigmoid used for speed fault detection. $\sigma^v = 1$ and $\delta^v = 15$	127
5.8	Received objects. On the left is a normal spacial distribution (objects are well separated and placed logically. On the right is an abnormal spatial distribution (objects are on top of each other and cars are inside buildings or outside the road), which can be a sign of track spoofing.	128
5.9	Sigmoid used for the detection of abnormal distances. $\sigma^d = 4$ and $\delta^d = 20$	128
5.10	Received track across three time steps. On the left it follows a predictable trajectory. On the right it move outside of what the model can predict, which can be a track duplication for example.	129
5.11	Comparison of a consensus object (green car) and received object (blue car). On the left, both objects match and are detectable (green cells) so the objects are similar and trust will be increased. On the right, the received object is not detectable (unknown grey cells or orange undetectable cells), so trust will not increase.	130
5.12	Comparison of a consensus object (green car) and received object (blue car). On the left is the same neutral situation as object similarity: objects do not match but due to the lack of detectability, non-trust is not increased. On the right, objects do not match both are both detectable by their sources (green and cyan cells for consensus and received respectively). Objects are dissimilar and detectable so non-trust is increased.	131
5.13	Result from several support functions. Top row are the consensus and reference detectability grids, with white pixels being unknown Ω^D , blue pixels being undetectable $\not D$ and green pixels being detectable D . Bottom row are the cell-wise results of applying the Joussemme distance, Song-Deng divergence and Equation (5.13) respectively. . .	132
5.14	Output of the support function D^{support} for all input values and $\sigma^D = 200$	133
5.15	Sigmoid function used in free-space confirmation with $\delta^{\text{fsi}} = 500$ cells and $\sigma^{\text{fsi}} = 200$	133
5.16	Example of perception dilution. The more untrustworthy a source is, the more transparent and enlarged objects become.	134
5.17	Example of trust plotting used in the following curves. Green area and curve is $m^T(T)$, in red $m^T(\not T)$ and grey $m^T(\Omega^T)$	135
5.18	Comparison without trust (blue curves) and with trust (orange curves) on a nominal situation for vehicle v_1	137
5.19	Comparison without trust (blue curves) and with trust (orange curves) on a faulty situation for vehicle v_1	139
5.20	Comparison with and without trust on faulty and nominal situations for vehicle v_1	141

5.21	Evolution of BBAs in the estimation of trust over time in the form of a tree. Trust estimated by v_1 on v_2 in the case where v_2 is adding faults. Despite trust (green curves) being created in <code>conf-osi</code> and <code>conf-fsi</code> , non-trust (red curves accumulated on top of trust) takes precedence.	142
5.22	Evolution of BBAs in the estimation of trust over time in the form of a tree. Trust estimated by v_2 on v_1 in the the nominal case. Notice how distrust (red curves) estimated in leaves (<code>cohe-spc-ro</code> and <code>conf-odi-rc</code>) rise to the top to create valesys in trust (green curves) in <code>hist</code>	142
5.23	Comparison of several parameters tuning under erroneous information on vehicle v_1 : cautious in blue, moderate in orange and credulous in green.	144
B.1	Overview of the Roundabout dataset with the trajectory of the three vehicles. The red vehicle starts in the center, then joined by the green vehicle then joined by the blue vehicle.	167
B.2	Overview of the Intersection dataset with the trajectory of the three vehicles and the VLS-128 position (purple dot).	168
B.3	Overview of the Overtaking dataset with the trajectory of the five vehicles and the VLS-128 position (purple dot). The yellow vehicle stops as vehicles pass by each other and blue overtakes it.	168
C.1	Fusion of background grid map and snapshot observation grids of the three vehicles at a particular time of the overtaking dataset. Colors are white for unknown, orange for static S , blue for dynamic D , red for immobile I , green for free F and cyan for passable P	170
D.1	Trust estimated by three vehicles about each others using several combination rules. Continuous lines are trustworthiness $m^{\mathcal{T}}(T)$ and dashed lines are untrustworthiness $m^{\mathcal{T}}(\mathcal{I})$ stacked on top of trustworthiness. Blue curves use Dempster's rule and orange Yager's.	171
E.1	Trust estimated by three vehicles about each others using several consensus building methods. Continuous lines are trustworthiness $m^{\mathcal{T}}(T)$ and dashed lines are untrustworthiness $m^{\mathcal{T}}(\mathcal{I})$ stacked on top of trustworthiness. Blue is for the global, orange for fused for fused and green for sequential. Note that the because the three methods produce results close to one another, curves are superimposed.	174

List of Tables

1.1	6 Levels of Automation as defined by the SAE.	12
3.1	Terms associated to the evaluation of binary classifiers.	65
3.2	Ratios used in binary evaluation and their meanings.	66
3.3	ROS packages developed and contributed to in the context of this PhD. Some are private for intellectual property reasons.	73
3.4	Summary of state and existence tracking parameters	77
3.5	Evaluation of sign detection for a total of 560 signs on the trajectory	79
3.6	Evaluation of car detection	81
4.1	Summary of similarity parameters.	105
4.2	Summary of existence tracking parameters.	107
4.3	Cell-wise evaluation of perceived free space by ground truth grids. . .	114
4.4	Comparison of standalone and cooperative perception for v_1	114
4.5	Comparison of standalone and cooperative perception for v_2	115
4.6	Comparison of standalone and cooperative perception for v_3	115
4.7	Comparison of no detectability, cooperative detectability and full de- tectability.	118
5.1	Summary of Trust Parameters.	134
5.2	Comparison with and without trust on a nominal situation for vehicle v_1	136
5.3	Comparison with and without trust on a faulty situation on vehicle v_1 .	138
5.4	Comparison with and without trust on faulty and nominal situations for vehicle v_1	140
5.5	Sets of parameters used in the following comparison.	143
5.6	Comparison of several parameters tuning under erroneous information for v_1	144
C.1	Evaluation of the automatic ground-truth method in comparison with a manual ground-truth.	170

Acronyms

- ACC** Adaptive Cruise Control. 11
- ADAS** Advanced Driver-Assistance System. 11
- AL** Alert Limit. 16, 17, 151
- AUC** Area Under Curve. 114, 136, 138, 140
- BBA** Basic Belief Assignment. 24, 28, 125, 130, 132, 134, 141, 155
- BEV** Bird Eye View. 57, 58
- BSM** Basic Safety Message. 84, 85
- C-V2X** Cellular V2X. 84
- CAM** Cooperative Awareness Message. 85
- CI** Covariance Intersection. 36, 37, 39, 44, 88, 152
- CIF** Covariance Intersection Filter. 38, 41–43, 45, 49, 88, 152
- cITS** Cooperative Intelligent Transportation Systems. 121
- CNN** Convolutional Neural Network. 57, 58
- COLA** Cardinalized Optimal Linear Assignment. 68
- CP** Cooperative Perception. 13, 15, 85
- CPM** Collective Perception Message. 85, 86, 88, 121
- DBSCAN** Density-Based Spatial Clustering of Applications with Noise. 55, 74
- DENM** Decentralized Environment Notification Message. 85, 86
- DSRC** Dedicated Short Range Communication. 84
- DST** Dempster-Shafer Theory. 23
- ECEF** Earth-Centered Earth-Fixed. 89, 90
- EKF** Extended Kalman Filter. 35

- ENU** East-North-Up. 90
- ETSI** European Telecommunications Standards Institute. 85, 154
- FCI** Fast Covariance Intersection. 38
- FN** False Negative. 65, 66, 111, 113, 114
- FoV** Field of View. 71, 85, 99, 101, 108, 119, 126, 136, 138, 143, 164
- FP** False Positive. 65, 66, 111, 114
- FPFD** Forward-Prediction Fusion and Decorrelation. 94
- FPR** False Positive Rate. 114, 136
- FS** Free Space. 99, 126, 131, 134
- GNN** Global Nearest Neighbors. 61, 62, 77, 78, 80, 111, 135
- GNSS** Global Navigation Satellite System. 70, 71, 89
- HD** High Definition. 75
- HOTA** Higher Order Tracking Accuracy. 68, 69
- ICI** Inverse CI. 38, 39, 152
- IFCI** Improved Fast Covariance Intersection. 38, 39, 152
- IMF** Information Matrix Fusion. 88
- IMM** Interacting Multiple Model. 82
- IMU** Inertial Measurement Unit. 52, 70, 71
- IoU** Intersection over Union. 66, 123
- IPM** Inverse Perspective Mapping. 58
- ITCI** Information-Theoretic Covariance Intersection. 38, 39, 152
- ITS** Intelligent Transportation Systems. 84
- JIT** Just In Time. 165
- JPDA** Joint Probabilistic Data Association. 62, 145
- K-CIF** Kalman-Covariance Intersection Filter. 41, 42, 152
- KF** Kalman Filter. 32, 36, 41–43, 45, 46, 49, 152
- LDM** Local Dynamic Map. 60, 62

- LiDAR** Light Detection And Ranging. 41, 51, 52, 57, 69, 71–74, 92, 99, 100, 155, 169
- MAP** . 85
- MHT** Multiple Hypothesis Tracking. 62, 82
- MOTA** Multi-Object Tracking Accuracy. 67
- MOTP** Multi-Object Tracking Precision. 67
- NEES** Normalized Estimation Error Squared. 66, 104, 105
- NLLR** Negative Log Likelihood Ratio. 61
- OBU** On-Board Unit. 84
- ODD** Operational Design Domain. 17
- OOS** Out-Of-Sequence. 92–94
- OSM** Open Street Map. 98
- OSPA** Optimal Sub-Pattern Assignment. 68
- PCA** Principal Component Analysis. 56
- PHD** Probability Hypothesis Density. 89
- PKI** Public Key Infrastructure. 87, 154
- PoV** Point of View. 108, 118, 119, 145, 169
- PPK** Post-Processing Kinematics. 71, 169
- PPP** Precise Point Positioning. 70
- PR** Precision-Recall. 66, 69, 114
- RMSE** Root Mean Square Error. 67, 69, 78, 79, 81, 113, 114, 118
- ROC** Receiver Operating Characteristics. 66, 114
- ROS** Robotic Operating System. 73, 163–165, 167
- RSU** Road Side Unit. 84
- RTK** Real Time Kinematics. 70
- SAE** Society of American Engineer. 11, 12, 84, 85, 158
- SCI** Split CI. 39, 76, 135

- SCIF** Split Covariance Intersection Filter. 19, 39, 40, 46, 47, 49, 76, 96, 119, 152
- SIMD** Single Instruction Multiple Data. 163
- SLAM** Simultaneous Localization And Mapping. 48
- SLR** Scan Line Run. 55, 164
- SPaT** . 85
- T2TF** Track-To-Track Fusion. 93, 96, 119
- TBM** Transferable Belief Model. 23
- TIR** Target Integrity Risk. 16, 17
- TN** True Negative. 65, 112–114
- TP** True Positive. 65, 111, 114
- TPR** True Positive Rate. 114
- TTC** Time To Collision. 69
- UKF** Unscented Kalman Filter. 35
- USA** United States of America. 11, 70
- VRU** Vulnerable Road Users. 51
- WGS84** World Geodesic System 1984. 89, 90

Appendix A

Developments

This work relies on a significant amount of software development, the detail of which is given in this annex.

The Robotic Operating System (ROS) middle-ware¹ have been extensively used to facilitate communication between components and to provide a framework of tools and libraries. The version used in the thesis is ROS noetic. As a rapid reminder, ROS imposes that processing is done as simple tasks separated in *nodes*, which can communicate with each other using *topics*.

A.1 Datasets

Unless specified, all data in this thesis have been recorded in real life. To facilitate the process of recording, a series of Python scripts, configurations and tools for live-recording, live-monitoring, post-processing and organization have been developed². They aim at standardizing how datasets are recorded, stored and processed at Heudiasyc.

A.2 Perception

The perception stack that serves as the basis for cooperation is based on a LiDAR cars and sign detector developed jointly with other PhD students. It is written in C++ in the form of a LiDAR processing library *lidar_utils*³. Point-clouds are represented as $N \times M$ Eigen matrices⁴ and operations on them are done using parallel processing (Single Instruction Multiple Data) as much as possible. Real-time processing is easily attained on modern CPUs but a possible improvement could be to use GPU oriented libraries or study the Eigen-Cuda interfacing.

For example for car-detection, the order of operations is:

1. Untwist the point cloud

¹<https://www.ros.org/>

²https://gitlab.utc.fr/hds_vehint/datasets

³https://gitlab.utc.fr/hds_vehint/lidar_utils

⁴<https://eigen.tuxfamily.org>

2. Separate ground and non-ground points
3. Cluster non-ground points
4. Determine cluster 2D convex hull
5. Filter out clusters outside the road using polygon operations
6. Determine cluster bounding-box

Processing steps are chained together using ROS *nodelets*, lightweight nodes that limit communication overheads. To keep a common interface between students, these nodelets implement multiple methods (e.g. clustering can either be euclidean or Scan Line Run), and a particular method can be chosen at run-time using dynamic parameters⁵ along with method-specific parameters. This simplifies development and testing as it makes qualitative evaluation and tuning quicker.

To facilitate handling from other students, this package is automatically built when pushed on the git repository using Gitlab CI⁶. Code is checked for errors, tested, compiled and exported as an installable Debian package using a custom facilitator package⁷.

Another example of perception processing is how detectability grids are computed. They are represented with a community-developped library: `grid_map`⁸. They rely on Eigen matrices and provide higher-level functions to access cells. In particular, it is possible to access all cells within a polygon, which is used to project the free-space and FoV polygons in a grid.

A.3 Tracking

Higher level operations such as tracking or trust evaluation are implemented in Python⁹. Tracks are stored in dictionaries sorted by time step using the `sortedcontainers` library¹⁰ and belief functions using a modified version of `pyds`¹¹. Here again the specific methods to use (e.g. Kalman, CI or SCI) are specified and can be changed at runtime along with other parameters. To help understand and debug trackers, live display of their inner objects have been implemented using `pyqtgraph`¹² along with a debugging mode that outputs large amounts of information about operations that take place.

While Python is adapted to rapid prototyping, it lacks in performance. To keep the best of the two worlds, performance critical functions have been implemented using one of two techniques:

⁵http://wiki.ros.org/dynamic_reconfigure

⁶<https://docs.gitlab.com/ee/ci/>

⁷https://gitlab.utc.fr/hds_vehint/gitlab_ci

⁸https://github.com/ANYbotics/grid_map

⁹<https://gitlab.utc.fr/multiception/multiception>

¹⁰<https://github.com/grantjenks/python-sortedcontainers>

¹¹<https://github.com/reineking/pyds>

¹²<https://github.com/pyqtgraph/pyqtgraph>

- Compile Python code Just In Time (JIT) using `numba`¹³
- Implement in C++ and bind to Python using `pybind11`¹⁴. In particular, this approach have been applied to grid maps to link `numpy` arrays with `Eigen` matrices. This have been made available to the community through an open-source pull request¹⁵.

Tracking and trust building functions are interfaced with the rest of the system as ROS nodes that communicate using custom messages.

A.4 Display

To visualize the output of previously mentionned nodes, the integrated ROS visualizer `rviz` is used. Built-in types (e.g. point clouds, polygons) can natively be displayed in it and plugins can be implemented to display custom types. In our case, plugins for object and detectability grids have been implemented¹⁶.

A.5 Special Processing

To evaluate an algorithm, recorded data have to be played back in post-processing. ROS provides `bags` as a way to post-process data in an online fashion. However, because data originates from three vehicles, it is particularly heavy to post-process on a single desktop computer. After a successful but sub-optimal attempt at distributing computations amongst multiple desktop computers using SSH, the best solution have been found in offline post-processing. It required refactoring previous modules to accept either online messages or offline batches but yield the best performance and reproducibility. Using these properties, a docker image have been used to perform the computations on a 80 cores / 512Go memory server from Heudiasyc. This allowed for the concurrent evaluation of multiple parameter sets in a reasonable time.

¹³<https://github.com/numba/numba>

¹⁴<https://github.com/pybind/pybind11>

¹⁵https://github.com/ANYbotics/grid_map/pull/335

¹⁶https://gitlab.utc.fr/multiception/rviz_multiception

Appendix B

Cooperative Datasets

This work has been done in Rémy Huet master internship, that has been published in (Huet et al. 2023)..

Three data-sets have been recorded, using the experimental setup described in Section 3.5.1. They each cover a particular cooperative scenario and have been recorded on open and busy roads. Datasets are available on the Heudiasyc dataset portal¹. They are provided in ROS bag format or exported as videos, zip and csv files along with the ground truth outputted from the method detailed in Appendix C.

B.1 Roundabout

Three vehicles driving around a roundabout as depicted in Figure B.1. One starts inside the roundabout with the other two progressively entering, for 4 minutes.

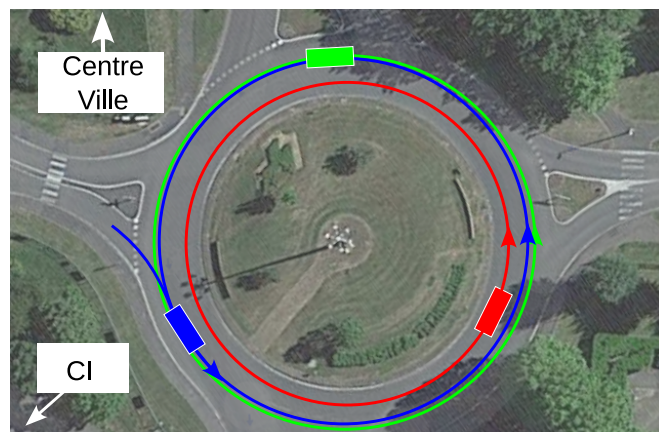


Figure B.1: Overview of the Roundabout dataset with the trajectory of the three vehicles. The red vehicle starts in the center, then joined by the green vehicle then joined by the blue vehicle.

¹<https://datasets.hds.utc.fr/project/12>

B.2 Intersection

Three vehicles driving near an intersection as depicted in Figure B.2. They pass in front of a VLS-128 stationed on the curb and cross each other paths regularly over a duration of 25 minutes.

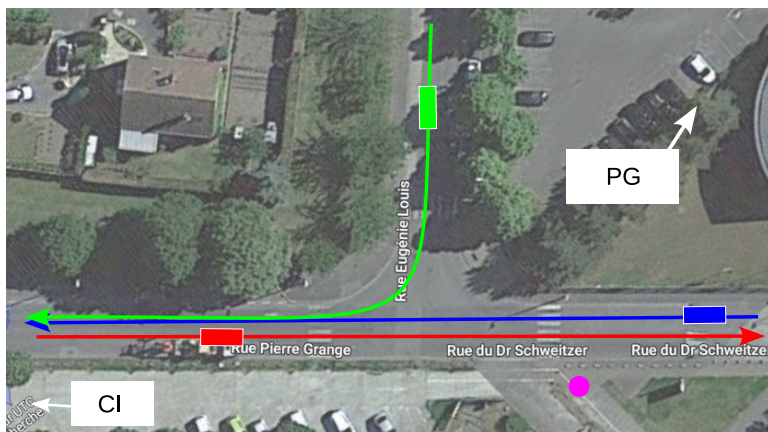


Figure B.2: Overview of the Intersection dataset with the trajectory of the three vehicles and the VLS-128 position (purple dot).

B.3 Overtaking

Five vehicles driving on a straight road as depicted in Figure B.3. Three are on the opposite lane from the other two while a VLS-128 is stationed on the curb. One vehicle stops as they pass by each other, then is overtaken by another vehicle in a scenario that lasts 1 minute.

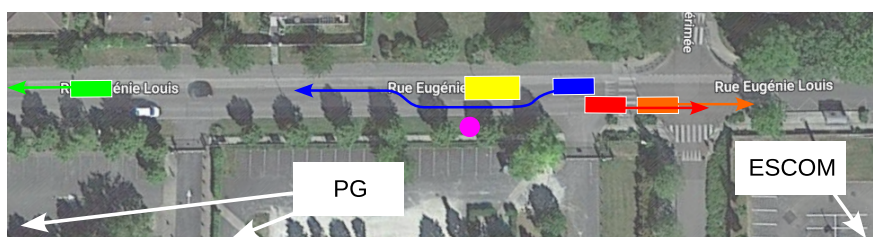


Figure B.3: Overview of the Overtaking dataset with the trajectory of the five vehicles and the VLS-128 position (purple dot). The yellow vehicle stops as vehicles pass by each other and blue overtakes it.

Appendix C

Cooperative Ground-Truth

To facilitate the process of labeling cooperative datasets, raw data (LiDAR point clouds and the PPK poses) are automatically processed to extract candidate labels using a method described in (Huet 2022; Huet et al. 2023). Offline post-processing allows for the use of heavy and non-causal processes. As such, because the accuracy of objects is not the focus but mainly their existence, grid-based approaches are a good solution to fuse multiple points of view. In particular, because PoVs are incomplete, an evidential representation has been used. The resulting grid contains information about free-space and objects. To potentially implement tracking, object information is split in three categories: immovable (buildings, road signs or vegetation), static (movable objects but stopped for the whole dataset such as parked cars) and dynamic (non-parked cars, pedestrians). As such, the following frame of discernment is defined:

$$\Theta = \{F, I, S, D\} \tag{C.1}$$

with F free-space, I immovable, S static object and D dynamic object. For convenience, three subsets are also defined: $M = \{S, D\}$ for unspecified movable objects, $O = \{I, S, D\}$ for unspecified occupancy and $P = \{F, D\}$ for *passable* areas that can either be free of dynamically occupied through time (Steyer et al. 2020).

In this approach, global measurement grids are generated from sensor measurements. First, a grid comprised of dynamic mass is generated below each vehicle using their PPK poses and sizes. LiDAR points are first classified using Cylinder3D (see Section 3.2.3) and then propagated to three subsets: free, immovable or unspecified movable. Using these measurement grids, a background mapping, depicted in Figure C.1 is first realized to highlight passable, immovable and static areas. It is used as a prior while fusing measurement grids and predicting (i.e. discounting) the fused grid.

A pass of clustering and filtering is also applied on the successive fused grids to generate tracks. When compared to a manual ground truth, the approach yields promising results that are described in Table C.1. Most of the error in these results can be explained by the naive nature of track generation. As such and due to lack of time, the manual ground truth is the one used for the rest of this

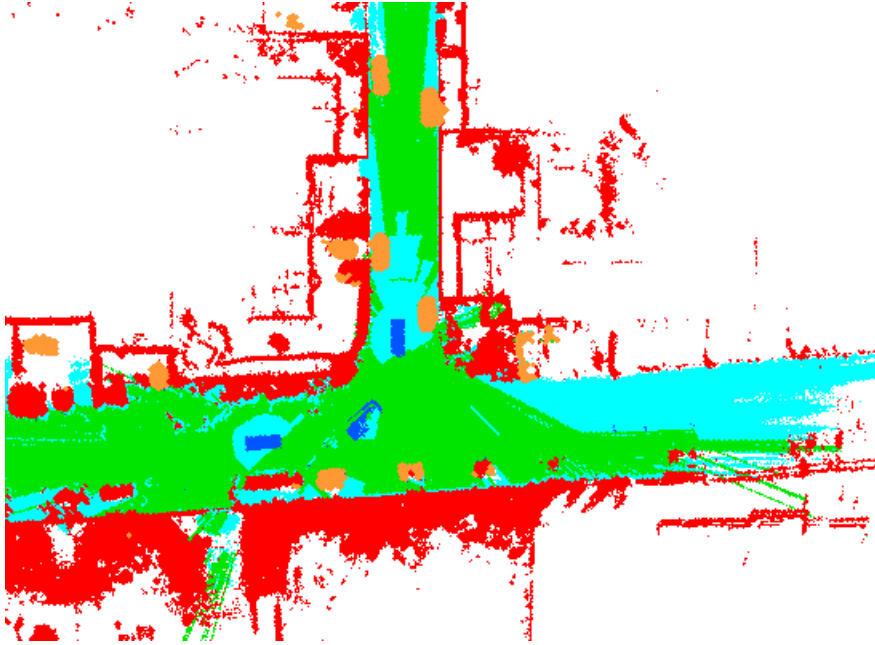


Figure C.1: Fusion of background grid map and snapshot observation grids of the three vehicles at a particular time of the overtaking dataset. Colors are white for unknown, orange for static S , blue for dynamic D , red for immobile I , green for free F and cyan for passable P .

manuscript. The semi-automatic method yields few erroneous objects but still requires a lengthy manual pass to filter them.

Table C.1: Evaluation of the automatic ground-truth method in comparison with a manual ground-truth.

Precision	Recall	F1-score
0.95	0.97	0.96

Appendix D

Study of the Combination Rule Used in Trust Estimation

When computing trust, Dempster’s rule is used to combine all nodes of the trust tree, which can raise several issues. For example, Dempster’s rule is known to yield counter-intuitive results in the presence of strong conflict (Zadeh 1979). A possible solution is to change how conflict is managed, such as with Yager’s rule that normalize conflict on the uncertainty. Because every term in trust estimation contains a degree of uncertainty (e.g. weights between nodes, sigmoid functions), counter-intuitive results should not appear. To confirm this hypothesis, the trust estimated using Dempster’s and Yager’s rule have been compared in Figure D.1. One can see that outside of short instances (around $t = 35$ for the trust v_1 estimates about v_2), the two methods produce exactly the same results. Dempster’s rule is thus not an issue with this formulation.

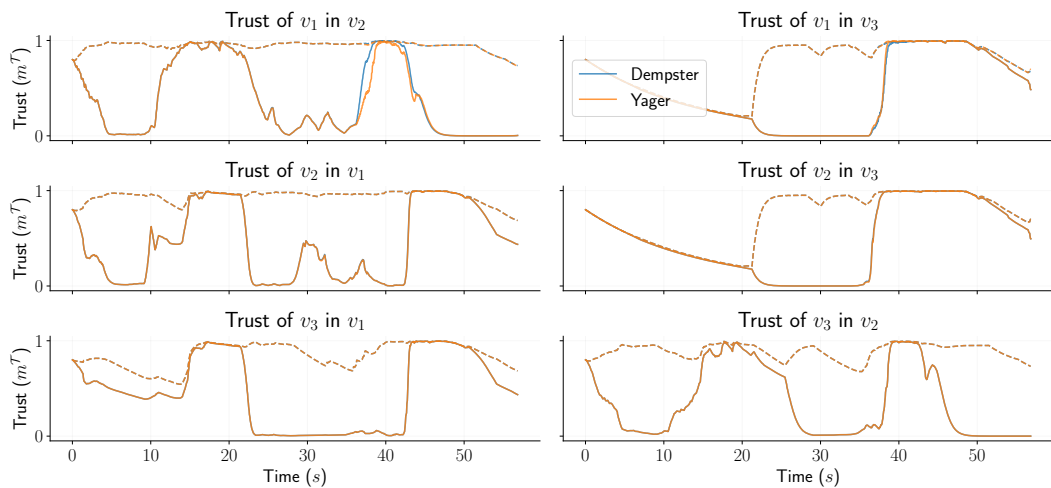


Figure D.1: Trust estimated by three vehicles about each others using several combination rules. Continuous lines are trustworthiness $m^T(T)$ and dashed lines are untrustworthiness $m^T(\mathcal{F})$ stacked on top of trustworthiness. Blue curves use Dempster’s rule and orange Yager’s.

Appendix E

Study of Consensus Building in Trust Estimation

When computing trust, the received perception is compared against a *consensus*. In this appendix, we study how it can be defined. There are three possible definitions.

The first, denoted *global* uses the output from the cooperative tracker. It naturally accounts for past trusts and contains the most complete information available, while not requiring additional computation. However, it can lead to self-confirmation when there is not enough redundancy. In areas that no other sources perceived, a source will confirm its own information, leading to over-confidence in trust. This could probably be prevented by using a cautious rule (Equation (2.12)) when confirming but due to its over-cautious nature, another solution is preferred.

The second possible definition, denoted *fused* consists in maintaining another tracker in parallel that does not include the source being confirmed. Its output can then be used as a consensus. Letting n be the number of sources to confirm, this means n trackers fusing $n - 1$ sources, which is computationally heavy but certifies that no trust-loops are introduced.

A third definition, denoted *sequential* reduces the amount of computation by confirming against each other sources separately. This means that Equation (5.9) is computed for $n - 1$ sources and their results are fused.

The main issue with the two last definitions is that in absence of redundancy, no confirmation can occur, leading to trust being lost over time. A mechanism to discount as a function of potential confirmation could be derived, though it is not included in this work.

In practice, the difference is not as significant as expected as illustrated in Figure E.1. Outside of several short instances where the global consensus differs from the other two (such as $t = 35$ in the trust v_1 estimates in v_2 or at $t = 40$ between v_3 and v_1), the estimated trusts are very similar. In particular, the sequential methods yields the same values as the fused one, for a portion of the computational cost, and as such is the preferred method.

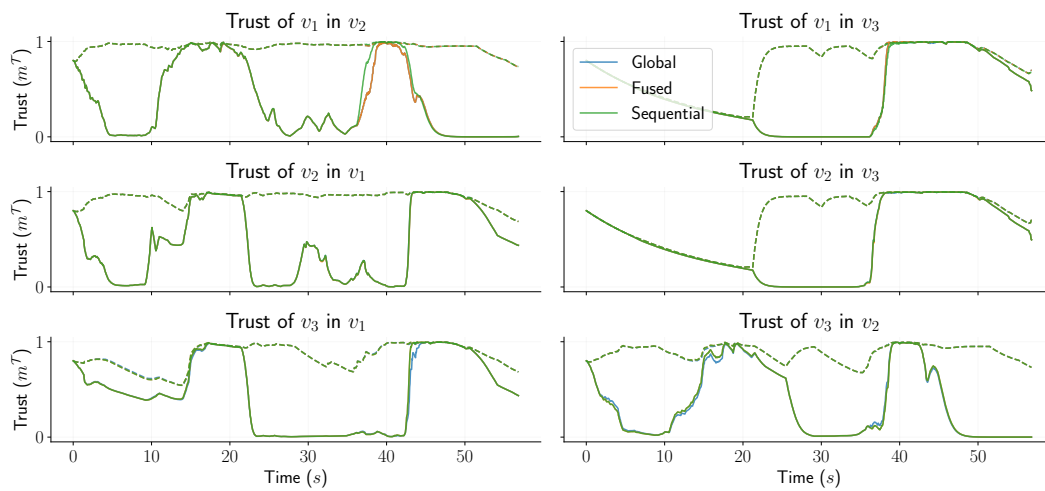


Figure E.1: Trust estimated by three vehicles about each others using several consensus building methods. Continuous lines are trustworthiness $m^T(T)$ and dashed lines are untrustworthiness $m^T(\mathcal{I})$ stacked on top of trustworthiness. Blue is for the global, orange for fused for fused and green for sequential. Note that the because the three methods produce results close to one another, curves are superimposed.

Bibliography

- Aeberhard, Michael (2017). “Object-Level Fusion for Surround Environment Perception in Automated Driving Applications”. Faculty of Electrical Engineering and Information Technology Of Dortmund (cit. on pp. 62, 64, 135).
- Aeberhard, Michael, Sascha Paul, Nico Kaempchen, and Torsten Bertram (June 2011). “Object Existence Probability Fusion Using Dempster-Shafer Theory in a High-Level Sensor Data Fusion Architecture”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2011.5940430 (cit. on pp. 62, 77, 96, 97, 107).
- Al Hage, Joelle, Stefano Mafrica, Maan El Badaoui El Najjar, and Franck Ruffier (Aug. 2019). “Informational Framework for Minimalistic Visual Odometry on Outdoor Robot”. In: *IEEE Transactions on Instrumentation and Measurement*. DOI: 10.1109/TIM.2018.2871228 (cit. on p. 36).
- Allig, Christoph, Tim Leinmüller, Prachi Mittal, and Gerd Wanielik (Dec. 2019). “Trustworthiness Estimation of Entities within Collective Perception”. In: *IEEE Vehicular Networking Conference*. DOI: 10.1109/VNC48660.2019.9062796 (cit. on pp. 97, 122, 123).
- Allig, Christoph and Gerd Wanielik (June 2019a). “Alignment of Perception Information for Cooperative Perception”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2019.8814108 (cit. on pp. 86, 88, 89, 93).
- Allig, Christoph and Gerd Wanielik (Oct. 2019b). “Dynamic Dissemination Method for Collective Perception”. In: *IEEE Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC.2019.8917266 (cit. on p. 88).
- Ambrosin, Moreno, Ignacio J Alvarez, Cornelius Buerkle, Lily L Yang, Fabian Oboril, Manoj R Sastry, and Kathiravetpillai Sivanesan (Oct. 2019). “Object-Level Perception Sharing Among Connected Vehicles”. In: *IEEE Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC.2019.8916837 (cit. on pp. 67, 88).
- Ambrosin, Moreno, Lily L Yang, Xiruo Liu, Manoj R Sastry, and Ignacio J Alvarez (Oct. 2019). “Design of a Misbehavior Detection System for Objects Based Shared Perception V2X Applications”. In: *IEEE Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC.2019.8917066 (cit. on pp. 122, 123, 129).
- Ansari, Mohammad Raashid, Jean-Philippe Monteuis, Jonathan Petit, and Cong Chen (Dec. 2021). “V2X Misbehavior and Collective Perception Service: Considerations for Standardization”. In: *IEEE Conference on Standards for Communications and Networking*. DOI: 10.1109/CSCN53733.2021.9686156 (cit. on pp. 86, 121).

- Ashraf, Imran, Soojung Hur, and Yongwan Park (2017). “An Investigation of Interpolation Techniques to Generate 2D Intensity Image From LIDAR Data”. In: *IEEE Access*. DOI: 10.1109/ACCESS.2017.2699686 (cit. on p. 66).
- Balakrishnan, Arjun (Dec. 2020). “Integrity Analysis of Data Sources in Multi-modal Localization System” (cit. on p. 15).
- Bar-Shalom, Y. (July 2002). “Update with Out-of-Sequence Measurements in Tracking: Exact Solution”. In: *IEEE Transactions on Aerospace and Electronic Systems*. DOI: 10.1109/TAES.2002.1039398 (cit. on p. 94).
- Bar-Shalom, Yaakov, Fred Daum, and Jim Huang (Dec. 2009). “The Probabilistic Data Association Filter”. In: *IEEE Control Systems Magazine*. DOI: 10.1109/MCS.2009.934469 (cit. on p. 62).
- Barrios, Pablo, Martin Adams, Keith Leung, Felipe Inostroza, Ghayur Naqvi, and Marcos E. Orchard (Feb. 2017). “Metrics for Evaluating Feature-Based Mapping Performance”. In: *IEEE Transactions on Robotics*. DOI: 10.1109/TR0.2016.2627027 (cit. on p. 68).
- Behley, Jens, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall (Oct. 2019). “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *IEEE/CVF International Conference on Computer Vision*. DOI: 10.1109/ICCV.2019.00939 (cit. on p. 58).
- Bernardin, Keni and Rainer Stiefelhagen (Dec. 2008). “Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics”. In: *Journal on Image and Video Processing*. DOI: 10.1155/2008/246309 (cit. on p. 67).
- Bißmeyer, Norbert, Klaus Henrik Schröder, Jonathan Petit, Sebastian Mauthofer, and Kpatcha M. Bayarou (Dec. 2013). “Experimental Analysis of Misbehavior Detection and Prevention in VANETs”. In: *2013 IEEE Vehicular Networking Conference*. DOI: 10.1109/VNC.2013.6737612 (cit. on p. 122).
- Boritz, J. Efrim (Dec. 1, 2005). “IS Practitioners’ Views on Core Concepts of Information Integrity”. In: *International Journal of Accounting Information Systems*. DOI: 10.1016/j.accinf.2005.07.001 (cit. on p. 15).
- Brebion, Vincent, Julien Moreau, and Franck Davoine (Sept. 2022). “Real-Time Optical Flow for Vehicular Perception With Low- and High-Resolution Event Cameras”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2021.3136358 (cit. on p. 73).
- Busch, Steffen, Christian Koetsier, Jeldrik Axmann, and Claus Brenner (June 2022). “LUMPI: The Leibniz University Multi-Perspective Intersection Dataset”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IV51971.2022.9827157 (cit. on p. 95).
- Caesar, Holger, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom (June 2020). “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR42600.2020.01164 (cit. on p. 65).
- Caillot, Antoine, Safa Ouerghi, Pascal Vasseur, Rémi Boutteau, and Yohan Dupuis (2022). “Survey on Cooperative Perception in an Automotive Context”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2022.3153815 (cit. on pp. 85, 86).

- Capellier, Edouard, Franck Davoine, Vincent Fremont, Javier Ibanez-Guzman, and You Li (Nov. 2018). “Evidential Grid Mapping, from Asynchronous LIDAR Scans and RGB Images, for Autonomous Driving”. In: *IEEE International Conference on Intelligent Transportation Systems*. DOI: 10.1109/ITSC.2018.8569710 (cit. on pp. 58, 60).
- Chen, Qi, Sihai Tang, Qing Yang, and Song Fu (July 2019). “Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds”. In: *IEEE International Conference on Distributed Computing Systems*. DOI: 10.1109/ICDCS.2019.00058 (cit. on pp. 88, 95).
- Chen, Zhe, Jing Zhang, and Dacheng Tao (May 2019). “Progressive LiDAR Adaptation for Road Detection”. In: *IEEE/CAA Journal of Automatica Sinica*. DOI: 10.1109/JAS.2019.1911459 (cit. on p. 58).
- Chowdhury, Muktadir, Ashlesh Gawande, and Lan Wang (Apr. 18, 2017). “Secure Information Sharing among Autonomous Vehicles in NDN”. In: *International Conference on Internet-of-Things Design and Implementation*. DOI: 10.1145/3054977.3054994 (cit. on p. 87).
- Danquah, Paul and Henoch Kwabena-Adade (Aug. 21, 2020). “Public Key Infrastructure: An Enhanced Validation Framework”. In: *Journal of Information Security*. DOI: 10.4236/jis.2020.114016 (cit. on p. 87).
- Delooz, Quentin, Alexander Willecke, Keno Garlich, Andreas-Christian Hagau, Lars Wolf, Alexey Vinel, and Andreas Festag (2022). “Analysis and Evaluation of Information Redundancy Mitigation for V2X Collective Perception”. In: *IEEE Access*. DOI: 10.1109/ACCESS.2022.3170029 (cit. on p. 86).
- Dempster, A. P. (Apr. 1967). “Upper and Lower Probabilities Induced by a Multivalued Mapping”. In: *The Annals of Mathematical Statistics*. DOI: 10.1214/aoms/1177698950 (cit. on pp. 23, 26).
- Deng, Jiajun, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li (May 18, 2021). “Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. DOI: 10.1609/aaai.v35i2.16207 (cit. on p. 58).
- Denceux, Thierry (Feb. 1, 2008). “Conjunctive and Disjunctive Combination of Belief Functions Induced by Nondistinct Bodies of Evidence”. In: *Artificial Intelligence*. DOI: 10.1016/j.artint.2007.05.008 (cit. on p. 28).
- Du, Yujun, Jinling Wang, Chris Rizos, and Ahmed El-Mowafy (Dec. 1, 2021). “Vulnerabilities and Integrity of Precise Point Positioning for Intelligent Transport Systems: Overview and Analysis”. In: *Satellite Navigation*. DOI: 10.1186/s43020-020-00034-8 (cit. on p. 70).
- Dubois, Didier and Henri Prade (2008). “A Set-Theoretic View of Belief Functions”. In: *Classic Works of the Dempster-Shafer Theory of Belief Functions*. DOI: 10.1007/978-3-540-44792-4_14 (cit. on p. 27).
- Duraisamy, Bharanidhar and Tilo Schwarz (Sept. 2015). “Combi-Tor: Track-to-Track Association Framework for Automotive Sensor Fusion”. In: *IEEE International Conference on Intelligent Transportation Systems*. DOI: 10.1109/ITSC.2015.263 (cit. on p. 61).
- Durrant-Whyte, H. and M. Stevens (2001). “Data Fusion in Decentralised Sensing Networks”. In: *International Conference on Information Fusion* (cit. on p. 36).

- Elfes, A. (June 1989). “Using Occupancy Grids for Mobile Robot Perception and Navigation”. In: *Computer*. DOI: 10.1109/2.30720 (cit. on p. 60).
- Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *International Conference on Knowledge Discovery and Data Mining* (cit. on p. 55).
- ETSI (2019). *TR 103 562: Analysis of the Collective Perception Service* (cit. on p. 85).
- Ettinger, Scott, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurélien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov (2021). “Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset”. In: (cit. on pp. 65, 108).
- Franken, D. and A. Hupper (July 2005). “Improved Fast Covariance Intersection for Distributed Data Fusion”. In: *International Conference on Information Fusion*. DOI: 10.1109/ICIF.2005.1591849 (cit. on p. 38).
- Gabb, Michael, Holger Digel, Tobias Müller, and Rüdiger-Walter Henn (June 2019). “Infrastructure-Supported Perception and Track-level Fusion Using Edge Computing”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2019.8813886 (cit. on p. 88).
- Geiger, Andreas, Philip Lenz, and Raquel Urtasun (June 2012). “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2012.6248074 (cit. on pp. 58, 65).
- Gottschalg, Grischa, Matthias Becker, and Stefan Leinen (Nov. 2020). “Integrity Concept for Sensor Fusion Algorithms Used in a Prototype Vehicle for Automated Driving”. In: *European Navigation Conference*. DOI: 10.23919/ENC48637.2020.9317323 (cit. on p. 17).
- Günther, Hendrik-Jörn, Björn Mennenga, Oliver Trauer, Raphael Riebl, and Lars Wolf (Dec. 2016). “Realizing Collective Perception in a Vehicle”. In: *IEEE Vehicular Networking Conference*. DOI: 10.1109/VNC.2016.7835930 (cit. on pp. 86, 88, 89).
- Herpel, Thomas, Christoph Lauer, Reinhard German, and Johannes Salzberger (Nov. 2008). “Multi-Sensor Data Fusion in Automotive Applications”. In: *International Conference on Sensing Technology*. DOI: 10.1109/ICSENST.2008.4757100 (cit. on p. 87).
- Héry, Elwan, Philippe Xu, and Philippe Bonnifait (May 9, 2017). “One-Dimensional Cooperative Localization for Vehicles Equipped with Mono-Frequency GNSS Receivers”. In: (cit. on p. 43).
- Héry, Elwan, Philippe Xu, and Philippe Bonnifait (2021). “Consistent Decentralized Cooperative Localization for Autonomous Vehicles Using LiDAR, GNSS and HD Maps”. In: *Journal of Field Robotics* (cit. on p. 38).
- Higham, Nicholas J. (Jan. 2002). *Accuracy and Stability of Numerical Algorithms*. Second. DOI: 10.1137/1.9780898718027 (cit. on p. 36).

- Hoss, Michael, Maike Scholtes, and Lutz Eckstein (Aug. 1, 2022). “A Review of Testing Object-Based Environment Perception for Safe Automated Driving”. In: *Automotive Innovation*. DOI: 10.1007/s42154-021-00172-y (cit. on p. 68).
- Houenou, Adam, Philippe Bonnifait, Veronique Cherfaoui, and Jean-François Boissou (June 2012). “A Track-to-Track Association Method for Automotive Perception Systems”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2012.6232261 (cit. on p. 62).
- Huet, Rémy (2022). *Perception Coopérative Pour Les Véhicules Autonomes : Méthodes de Traitement Pour Jeux de Données*. Master Internship Report (cit. on p. 169).
- Huet, Rémy, Antoine Lima, Philippe Xu, Philippe Bonnifait, and Veronique Cherfaoui (Sept. 2023). “Collaborative Grid Mapping for Moving Object Tracking Evaluation”. In: *IEEE Intelligent Transportation Systems Conference* (cit. on pp. 167, 169).
- Hurl, Braden, Robin Cohen, Krzysztof Czarnecki, and Steven Waslander (Oct. 2020). “TruPercept: Trust Modelling for Autonomous Vehicle Cooperative Perception from Synthetic Data”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IV47402.2020.9304695 (cit. on pp. 94, 122, 123).
- Jiménez, Víctor, Jorge Godoy, Antonio Artuñedo, and Jorge Villagra (2021). “Ground Segmentation Algorithm for Sloped Terrain and Sparse LiDAR Point Cloud”. In: *IEEE Access*. DOI: 10.1109/ACCESS.2021.3115664 (cit. on pp. 53, 55, 75).
- Jousselme, Anne-Laure, Dominic Grenier, and Éloi Bossé (June 1, 2001). “A New Distance between Two Bodies of Evidence”. In: *Information Fusion*. DOI: 10.1016/S1566-2535(01)00026-4 (cit. on p. 131).
- Julier, S.J. and J.K. Uhlmann (June 1997a). “A Non-Divergent Estimation Algorithm in the Presence of Unknown Correlations”. In: *American Control Conference*. DOI: 10.1109/ACC.1997.609105 (cit. on p. 36).
- Julier, Simon and Jeffrey Uhlmann (June 20, 2001). “General Decentralized Data Fusion with Covariance Intersection (CI)”. In: *Handbook of Multisensor Data Fusion, Theory and Practice*. DOI: 10.1201/9781420038545.ch12 (cit. on p. 39).
- Julier, Simon J. and Jeffrey K. Uhlmann (July 28, 1997b). “New Extension of the Kalman Filter to Nonlinear Systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. DOI: 10.1117/12.280797 (cit. on p. 35).
- Kalman, Rudolph Emil (1960). “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME—Journal of Basic Engineering* (cit. on pp. 32, 35).
- Keysight (Aug. 31, 2018). *Whitepaper: DSRC 802.11p Is Ready for Advanced Driver Assistance Systems (ADAS)* (cit. on p. 84).
- Khan, Muhammad Altamash (Oct. 2019). “Comparison of Track to Track Fusion Methods for Nonlinear Process and Measurement Models”. In: *Sensor Data Fusion: Trends, Solutions, Applications*. DOI: 10.1109/SDF.2019.8916652 (cit. on p. 76).
- Kim, Seong-Woo, Wei Liu, Marcelo H. Ang, Emilio Frazzoli, and Daniela Rus (Aut. 2015). “The Impact of Cooperative Perception on Decision Making and

- Planning of Autonomous Vehicles”. In: *IEEE Intelligent Transportation Systems Magazine*. DOI: 10.1109/MITS.2015.2409883 (cit. on pp. 68, 87).
- Ku, Jason, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander (Oct. 2018). “Joint 3D Proposal Generation and Object Detection from View Aggregation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IRROS.2018.8594049 (cit. on p. 57).
- Kurdej, Marek and Véronique Cherfaoui (Dec. 19, 2013). “Conservative, Proportional and Optimistic Contextual Discounting in the Belief Functions Theory”. In: *International Conference on Information Fusion* (cit. on p. 29).
- Lacoste, Johann, Elie Randriamiarintsoa, Abderrahim Kasmi, François Pomerleau, Roland Chapuis, Christophe Debain, and Romuald Aufrère (Sept. 2021). “Dynamic Lambda-Field: A Counterpart of the Bayesian Occupancy Grid for Risk Assessment in Dynamic Environments”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IRROS51168.2021.9636804 (cit. on p. 60).
- Lang, Alex H., Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom (June 2019). “PointPillars: Fast Encoders for Object Detection From Point Clouds”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2019.01298 (cit. on p. 58).
- Li, Chuyi, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei (Sept. 7, 2022). *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications* (cit. on p. 57).
- Li, H. and F. Nashashibi (Oct. 2011). “Multi-Vehicle Cooperative Perception and Augmented Reality for Driver Assistance: A Possibility to ‘See’ through Front Vehicle”. In: *IEEE Conference on Intelligent Transportation Systems*. DOI: 10.1109/ITSC.2011.6083061 (cit. on p. 87).
- Li, Hao, Fawzi Nashashibi, and Ming Yang (Dec. 2013). “Split Covariance Intersection Filter: Theory and Its Application to Vehicle Localization”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2013.2267800 (cit. on pp. 39, 43, 47).
- Li, You, Julien Moreau, and Javier Ibanez-Guzman (May 19, 2022). *Unconventional Visual Sensors for Autonomous Vehicles*. DOI: 10.48550/arXiv.2205.09383 (cit. on pp. 72, 73).
- Lima, Antoine, Philippe Bonnifait, Véronique Cherfaoui, and Joelle Al Hage (Sept. 2021). “Data Fusion with Split Covariance Intersection for Cooperative Perception”. In: *IEEE International Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC48978.2021.9564963 (cit. on pp. 40, 41).
- Lima, Antoine, Véronique Cherfaoui, and Philippe Bonnifait (Oct. 26, 2022). “Evidential Trustworthiness Estimation for Cooperative Perception”. In: *International Conference on Belief Functions* (cit. on p. 124).
- Lima, Antoine, Anthony Welte, Philippe Bonnifait, and Philippe Xu (Dec. 2020). “LiDAR Observations by Motion Compensation and Scan Accumulation”. In: *International Conference on Control, Automation, Robotics and Vision*. DOI: 10.1109/ICARCV50220.2020.9305365 (cit. on pp. 16, 48).

- Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick (2014). “Microsoft COCO: Common Objects in Context”. In: *European Conference on Computer Vision*. DOI: 10.1007/978-3-319-10602-1_48 (cit. on pp. 64, 65).
- Liu, Xiruo, Lily Yang, Ignacio Alvarez, Kathiravetpillai Sivanesan, Arvind Merwaday, Fabian Oboril, Cornelius Buerkle, Manoj Sastry, and Leonardo Gomes Baltar (July 2021). “MISO- V: Misbehavior Detection for Collective Perception Services in Vehicular Communications”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IV48863.2021.9575970 (cit. on p. 123).
- Luiten, Jonathon, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe (Feb. 1, 2021). “HOTA: A Higher Order Metric for Evaluating Multi-object Tracking”. In: *International Journal of Computer Vision*. DOI: 10.1007/s11263-020-01375-2 (cit. on pp. 68, 69).
- Luthardt, Stefan, Chao Han, Volker Willert, and Matthias Schreier (June 2017). “Efficient Graph-Based V2V Free Space Fusion”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2017.7995843 (cit. on p. 59).
- Ma, Jianbing, Weiru Liu, Didier Dubois, and Henri Prade (Aug. 2011). “Bridging Jeffrey’s Rule, Agm Revision and Dempster Conditioning in the Theory of Evidence”. In: *International Journal on Artificial Intelligence Tools*. DOI: 10.1142/S0218213011000401 (cit. on p. 125).
- Madl, Tobias (July 2021). “Security Concept with Distributed Trust-Levels for Autonomous Cooperative Vehicle Networks”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IV48863.2021.9576024 (cit. on p. 87).
- Mao, Jiageng, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li (June 19, 2022). *3D Object Detection for Autonomous Driving: A Review and New Outlooks*. DOI: 10.48550/arXiv.2206.09474 (cit. on p. 52).
- Meagher, Donald (June 1, 1982). “Geometric Modeling Using Octree Encoding”. In: *Computer Graphics and Image Processing*. DOI: 10.1016/0146-664X(82)90104-6 (cit. on p. 55).
- Mercier, David, Benjamin Quost, and Thierry Dencœux (2005). “Contextual Discounting of Belief Functions”. In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. DOI: 10.1007/11518655_47 (cit. on p. 29).
- Milan, Anton, Konrad Schindler, and Stefan Roth (June 2013). “Challenges of Ground Truth Evaluation of Multi-target Tracking”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. DOI: 10.1109/CVPRW.2013.111 (cit. on p. 67).
- Milanés, Vicente, David González, Francisco Navas, Imane Mahtout, Alexandre Armand, Clément Zinoune, Arunkumar Ramaswamy, Farid Bekka, Nievsabel Molina, Emmanuel Battesti, Yvon Kerdoncuff, Carlos Guindel, Jorge Beltrán, Irene Cortés, Alejandro Barrera, and Fernando Garcia (July 2022). “The Tornado Project: An Automated Driving Demonstration in Peri-Urban and Rural Areas”. In: *IEEE Intelligent Transportation Systems Magazine*. DOI: 10.1109/MITS.2021.3068067 (cit. on p. 13).
- Milioto, Andres, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss (Nov. 2019). “RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IRoS40897.2019.8967762 (cit. on p. 57).

- Miucic, R., A. Sheikh, Z. Medenica, and R. Kunde (Aug. 2018). “V2X Applications Using Collaborative Perception”. In: *IEEE Vehicular Technology Conference*. DOI: 10.1109/VTCFa11.2018.8690818 (cit. on p. 69).
- Moras, Julien, Véronique Cherfaoui, and Philippe Bonnifait (May 2011). “Credibilist Occupancy Grids for Vehicle Perception in Dynamic Environments”. In: *IEEE International Conference on Robotics and Automation*. DOI: 10.1109/ICRA.2011.5980298 (cit. on p. 60).
- Muntzinger, Marc., Michael Aeberhard, Sebastian Zuther, Mirko Mählich, Matthias Schmid, Jürgen Dickmann, and Klaus Dietmayer (2010). “Reliable Automotive Pre-Crash System with out-of-Sequence Measurement Processing”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2010.5548149 (cit. on p. 93).
- Nettleton, Eric W. and Hugh F. Durrant-Whyte (Oct. 4, 2001). “Delayed and Asequent Data in Decentralized Sensing Networks”. In: *Sensor Fusion and Decentralized Control in Robotic Systems IV*. DOI: 10.1117/12.444148 (cit. on p. 94).
- Nguyen, Tien Viet, Patil Shailesh, Baghel Sudhir, Gulati Kapil, Libin Jiang, Zhibin Wu, Durga Malladi, and Junyi Li (Nov. 2017). “A Comparison of Cellular Vehicle-to-Everything and Dedicated Short Range Communication”. In: *IEEE Vehicular Networking Conference*. DOI: 10.1109/VNC.2017.8275618 (cit. on p. 84).
- Nielsen, W. (July 2002). “Information Fusion Based on Fast Covariance Intersection Filtering”. In: *International Conference on Information Fusion*. DOI: 10.1109/ICIF.2002.1020907 (cit. on p. 38).
- Noack, Benjamin, Joris Sijs, Marc Reinhardt, and Uwe D. Hanebeck (May 1, 2017). “Decentralized Data Fusion with Inverse Covariance Intersection”. In: *Automatica*. DOI: 10.1016/j.automatica.2017.01.019 (cit. on p. 38).
- Nuss, D., S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer (2018). “A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-Time Application”. In: *International Journal of Robotics Research*. DOI: 10.1177/0278364918775523 (cit. on pp. 60, 89).
- Obst, Marcus, Laurens Hobert, and Pierre Reisdorf (Dec. 2014). “Multi-Sensor Data Fusion for Checking Plausibility of V2V Communications by Vision-Based Multiple-Object Tracking”. In: *IEEE Vehicular Networking Conference*. DOI: 10.1109/VNC.2014.7013333 (cit. on p. 122).
- Ko-PER Project (2014). “Communication for Cooperative Perception” (cit. on p. 84).
- Phillion, Jonah, Amlan Kar, and Sanja Fidler (June 2020). “Learning to Evaluate Perception Models Using Planner-Centric Metrics”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR42600.2020.01407 (cit. on p. 69).
- Pierre, Cyrille, Roland Chapuis, Romuald Aufrère, Jean Laneurit, and Christopher Debain (July 2018). “Range-Only Based Cooperative Localization for Mobile Robots”. In: *International Conference on Information Fusion*. DOI: 10.23919/ICIF.2018.8455692 (cit. on pp. 39, 43, 47).

- Qian, Rui, Xin Lai, and Xirong Li (Oct. 1, 2022). “3D Object Detection for Autonomous Driving: A Survey”. In: *Pattern Recognition*. DOI: 10.1016/j.patcog.2022.108796 (cit. on p. 52).
- Rauch, Andreas, Felix Klanner, Ralph Rasshofer, and Klaus Dietmayer (June 2012). “Car2X-based Perception in a High-Level Fusion Architecture for Co-operative Perception Systems”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2012.6232130 (cit. on pp. 88, 89).
- Reid, Tyler G.R., Sarah E. Houts, Robert Cammarata, Graham Mills, Siddharth Agarwal, Ankit Vora, and Gaurav Pandey (Sept. 24, 2019). “Localization Requirements for Autonomous Vehicles”. In: *SAE International Journal of Connected and Automated Vehicles*. DOI: 10.4271/12-02-03-0012 (cit. on p. 16).
- Reineking, Thomas (Mar. 24, 2014). “Belief Functions: Theory and Algorithms”. Universität Bremen (cit. on p. 26).
- Rheume, Francois and Abder Rezak Benaskeur (Dec. 2008). “Forward Prediction-Based Approach to Target-Tracking with Out-of-Sequence Measurements”. In: *IEEE Conference on Decision and Control*. DOI: 10.1109/CDC.2008.4738848 (cit. on p. 94).
- SAE-J2735 (2022). *J2735: V2X Communications Message Set Dictionary*. Society of Automotive Engineers (cit. on p. 84).
- SAE-J3016 (2021). *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Society of Automotive Engineers (cit. on p. 11).
- Sandblom, Fredrik and Joakim Sörstedt (June 2014). “Sensor Data Fusion for Multiple Configurations”. In: *IEEE Intelligent Vehicles Symposium Proceedings*. DOI: 10.1109/IVS.2014.6856557 (cit. on p. 76).
- Schiegg, Florian A., Ignacio Llatser, Daniel Bischoff, and Georg Volk (Jan. 2021). “Collective Perception: A Safety Perspective”. In: *Sensors*. DOI: 10.3390/s21010159 (cit. on p. 69).
- Schmidt, Robert K, Tim Leinmueller, Elmar Schoch, Albert Held, and Guenter Schaefer (2008). “Vehicle Behavior Analysis to Enhance Security in VANETs”. In: *IEEE Vehicle-to-Vehicle Communications Workshop* (cit. on pp. 122, 123).
- Schuhmacher, Dominic, Ba-Tuong Vo, and Ba-Ngu Vo (Aug. 2008). “A Consistent Metric for Performance Evaluation of Multi-Object Filters”. In: *IEEE Transactions on Signal Processing*. DOI: 10.1109/TSP.2008.920469 (cit. on p. 68).
- Seeliger, F. and K. Dietmayer (Oct. 2014). “Inter-Vehicle Information-Fusion with Shared Perception Information”. In: *IEEE Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC.2014.6958011 (cit. on pp. 43, 88, 95).
- Shafer, Glenn (1976). *A Mathematical Theory of Evidence*. DOI: 10.2307/j.ctv10vm1qb (cit. on p. 23).
- Smets, P. (2000). “Data Fusion in the Transferable Belief Model”. In: *Proceedings of the Third International Conference on Information Fusion*. DOI: 10.1109/IFIC.2000.862713 (cit. on p. 28).
- Smets, Philippe (Aug. 1, 1993). “Belief Functions: The Disjunctive Rule of Combination and the Generalized Bayesian Theorem”. In: *International Journal of Approximate Reasoning*. DOI: 10.1016/0888-613X(93)90005-X (cit. on p. 27).

- Smets, Philippe and Robert Kennes (Apr. 1, 1994). “The Transferable Belief Model”. In: *Artificial Intelligence*. DOI: 10.1016/0004-3702(94)90026-4 (cit. on p. 23).
- Smith, Randall C. and Peter Cheeseman (Dec. 1, 1986). “On the Representation and Estimation of Spatial Uncertainty”. In: *The International Journal of Robotics Research*. DOI: 10.1177/027836498600500404 (cit. on p. 91).
- Song, Yan, Zheng Hu, Tiancheng Li, and Hongqi Fan (Jan. 2022). “Performance Evaluation Metrics and Approaches for Target Tracking: A Survey”. In: *Sensors*. DOI: 10.3390/s22030793 (cit. on p. 68).
- Song, Yutong and Yong Deng (2019). “Divergence Measure of Belief Function and Its Application in Data Fusion”. In: *IEEE Access*. DOI: 10.1109/ACCESS.2019.2932390 (cit. on p. 131).
- Song, Zhiying, Fuxi Wen, Hailiang Zhang, and Jun Li (2022). *An Efficient and Robust Object-Level Cooperative Perception Framework for Connected and Automated Driving* (cit. on p. 88).
- Sridhar, Srivatsan and Azim Eskandarian (2019). “Cooperative Perception in Autonomous Ground Vehicles Using a Mobile-Robot Testbed”. In: *IET Intelligent Transport Systems*. DOI: 10.1049/iet-its.2018.5607 (cit. on p. 88).
- Steyer, Sascha, Christian Lenk, Dominik Kellner, Georg Tanzmeister, and Dirk Wollherr (July 2020). “Grid-Based Object Tracking With Nonlinear Dynamic State and Shape Estimation”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2019.2921248 (cit. on p. 169).
- Tao, Z. and P. Bonnifait (Sept. 2016). “Sequential Data Fusion of GNSS Pseudoranges and Dopplers With Map-Based Vision Systems”. In: *IEEE Transactions on Intelligent Vehicles*. DOI: 10.1109/TIV.2017.2658185 (cit. on p. 112).
- Taş, Ömer Şahin, Florian Kuhnt, J. Marius Zöllner, and Christoph Stiller (June 2016). “Functional System Architectures towards Fully Automated Driving”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2016.7535402 (cit. on p. 12).
- Thandavarayan, Gokulnath, Miguel Sepulcre, and Javier Gozalvez (June 2019). “Analysis of Message Generation Rules for Collective Perception in Connected and Automated Driving”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2019.8813806 (cit. on p. 86).
- Van der Heijden, Rens Wouter, Stefan Dietzel, Tim Leinmüller, and Frank Kargl (2019). “Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems”. In: *IEEE Communications Surveys Tutorials*. DOI: 10.1109/COMST.2018.2873088 (cit. on pp. 121, 122).
- Vasic, Milos, David Mansolino, and Alcherio Martinoli (Oct. 2016). “A System Implementation and Evaluation of a Cooperative Fusion and Tracking Algorithm Based on a Gaussian Mixture PHD Filter”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IRoS.2016.7759614 (cit. on pp. 68, 89).
- Vo, Ba-Ngu, Mahendra Mallick, and Yaakov Bar-Shalom (2015). “Multitarget Tracking”. In: *Wiley Encyclopedia of Electrical and Electronics Engineering* (cit. on p. 60).
- Wang, Tsun-Hsuan, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyan Zeng, and Raquel Urtasun (2020). “V2VNet: Vehicle-to-Vehicle Communica-

- tion for Joint Perception and Prediction”. In: *European Conference on Computer Vision*. DOI: 10.1007/978-3-030-58536-5_36 (cit. on p. 88).
- Wang, Yimin and X. Rong Li (Jan. 2012). “Distributed Estimation Fusion with Unavailable Cross-Correlation”. In: *IEEE Transactions on Aerospace and Electronic Systems*. DOI: 10.1109/TAES.2012.6129634 (cit. on p. 38).
- Wang, Yuan, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu (Sept. 25, 2018). *PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud*. DOI: 10.48550/arXiv.1807.06288 (cit. on p. 57).
- Wu, Bo and R. Nevatia (June 2006). “Tracking of Multiple, Partially Occluded Humans Based on Static Body Part Detection”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2006.312 (cit. on p. 67).
- Wu, Xiaopei, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu, and Deng Cai (July 4, 2022). “Sparse Fuse Dense: Towards High Quality 3D Detection with Depth Completion”. In: *Computer Vision and Pattern Recognition Conference*. DOI: 10.48550/arXiv.2203.09780 (cit. on p. 58).
- Xu, Runsheng, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma (May 2022). “OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication”. In: *International Conference on Robotics and Automation*. DOI: 10.1109/ICRA46639.2022.9812038 (cit. on p. 95).
- Yager, Ronald R. (Mar. 1, 1987). “On the Dempster-Shafer Framework and New Combination Rules”. In: *Information Sciences*. DOI: 10.1016/0020-0255(87)90007-7 (cit. on p. 27).
- Yavvari, Chaitanya, Zoran Duric, and Duminda Wijesekera (Oct. 2017). “Vehicular Dynamics Based Plausibility Checking”. In: *IEEE International Conference on Intelligent Transportation Systems*. DOI: 10.1109/ITSC.2017.8317883 (cit. on p. 122).
- Yoon, DoHyun Daniel, Beshah Ayalew, and G. G. Md. Nawaz Ali (July 2022). “Performance of Decentralized Cooperative Perception in V2V Connected Traffic”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2021.3063107 (cit. on p. 68).
- Yu, Chunlei, Véronique Cherfaoui, and Philippe Bonnifait (Dec. 2014). “An Evidential Sensor Model for Velodyne Scan Grids”. In: *International Conference on Control Automation Robotics Vision*. DOI: 10.1109/ICARCV.2014.7064369 (cit. on pp. 60, 69).
- Zabalegui, Paul, Gorka De Miguel, Alejandro Pérez, Jaizki Mendizabal, Jon Goya, and Iñigo Adin (2020). “A Review of the Evolution of the Integrity Methods Applied in GNSS”. In: *IEEE Access*. DOI: 10.1109/ACCESS.2020.2977455 (cit. on p. 70).
- Zacharia, Giorgos and Pattie Maes (Oct. 2000). “Trust Management through Reputation Mechanisms”. In: *Applied Artificial Intelligence*. DOI: 10.1080/08839510050144868 (cit. on p. 123).
- Zadeh, Lofti (1979). *On the Validity of Dempster’s Rule of Combination of Evidence* (cit. on p. 171).

- Zermas, D., I. Izzat, and N. Papanikolopoulos (May 2017). “Fast Segmentation of 3D Point Clouds: A Paradigm on LiDAR Data for Autonomous Vehicle Applications”. In: *IEEE International Conference on Robotics and Automation*. DOI: 10.1109/ICRA.2017.7989591 (cit. on pp. 53, 55).
- Zhang, Haolin, Dongfang Yang, Ekim Yurtsever, Keith A. Redmill, and Ümit Özgüner (Sept. 2021). “Faraway-Frustum: Dealing with Lidar Sparsity for 3D Object Detection Using Fusion”. In: *IEEE International Intelligent Transportation Systems Conference*. DOI: 10.1109/ITSC48978.2021.9564990 (cit. on p. 58).
- Zhou, Pengyuan, Pranvera Kortoçi, Yui-Pan Yau, Benjamin Finley, Xiujun Wang, Tristan Braud, Lik-Hang Lee, Sasu Tarkoma, Jussi Kangasharju, and Pan Hui (2022). “AICP: Augmented Informative Cooperative Perception”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2022.3155175 (cit. on p. 86).
- Zhu, Ni, Juliette Marais, David Bétaille, and Marion Berbineau (Sept. 2018). “GNSS Position Integrity in Urban Environments: A Review of Literature”. In: *IEEE Transactions on Intelligent Transportation Systems*. DOI: 10.1109/TITS.2017.2766768 (cit. on pp. 15, 16, 70).
- Zhu, Xinge, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin (June 2021). “Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR46437.2021.00981 (cit. on p. 57).
- Zhuo, Xuejun, Jianguo Hao, Duo Liu, and Yiqi Dai (Oct. 26, 2009). “Removal of Misbehaving Insiders in Anonymous VANETs”. In: *Proceedings of the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. DOI: 10.1145/1641804.1641824 (cit. on p. 123).
- Zoghby, Nicole, Véronique Cherfaoui, and Thierry Dencœux (July 9, 2013). “Optimal Object Association from Pairwise Evidential Mass Functions”. In: *International Conference on Information Fusion* (cit. on pp. 61, 103).
- Zoghby, Nicole El, Véronique Cherfaoui, and Thierry Denœux (June 2014). “Evidential Distributed Dynamic Map for Cooperative Perception in VANets”. In: *IEEE Intelligent Vehicles Symposium*. DOI: 10.1109/IVS.2014.6856550 (cit. on p. 64).