Cooperative Perception Algorithms for Networked Intelligent Vehicles

THÈSE N^{O} 7856 (2017)

PRÉSENTÉE LE 18 AOÛT 2017

À LA FACULTÉ DE L'ENVIRONNEMENT NATUREL, ARCHITECTURAL ET CONSTRUIT
LABORATOIRE DE SYSTÈMES ET ALGORITHMES INTELLIGENTS DISTRIBUÉS
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Milos VASIC

acceptée sur proposition du jury:

Dr D. Gillet, président du jury
Prof. A. Martinoli, directeur de thèse
Prof. P. Lima, rapporteur
Dr S. Geronimi, rapporteur
Prof. J.-Ph. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

Acknowledgments

THE accomplishment of this work would not have been possible without the help and support of many people. First and foremost, I would like to express my gratitude to my advisor, Alcherio Martinoli. His vision, guidance, and constant encouragement have pushed me forward and without him, the work in this form would not have been possible. It has been a great journey.

I would also like to thank my thesis committee — composed of Dr. Denis Gillet, Prof. Pedro Lima, Dr. Stéphane Geronimi, and Prof. Jean-Philippe Thiran — for taking the time to review this manuscript and for providing valuable feedback.

It was a true privilege to work with people from PSA Groupe — the financial and technical sponsor of my thesis. I am grateful to Olivier Pajot, Stéphane Geronimi, Jean-François Boissou, Franck Guillemard, and David Allard for their interactions and strategic decisions.

I have been fortunate to collaborate with a number of brilliant colleagues at the DISAL lab. The efforts of David Mansolino, Iñaki Navarro, and Guillaume Jornod have directly contributed to the content of this thesis. Thanks to Gael Lederrey, Jonathan Gan, and Johannes Løje, who joined us for semester or Master's projects, and whose work has made it into this thesis. A special thanks also goes to Ali Marjovi, whom I had the pleasure to write papers with, for all numerous discussions we have had over the past few years. This list would not be complete without thanking Steven Roelofsen for his willingness to brainstorm on many occasions, Sven Gowal, from whom I have learned the first things about intelligent vehicles, as well as Denis Rochat and Emmanuel Droz, who have supported me in my work the on the C-ZEROs. I am also thankful to Corinne Farquharson, who made sure the lab operations were always running smoothly, as well as the remaining DISAL members for all the moments, either fun or busy, that we shared together.

Operating a two-vehicle platform is not a one-man job. I would therefore like to thank Adrian Arfire for his help with the deployment, as well as Petar Pjanic, Ilija Bogunovic, Stanko Novakovic, Anwar Quraishi, and Roger Fong, who came and helped out during my experimental campaigns.

Acknowledgments

But there is more to life than work, and friends are the the important part of it. I am particularly grateful to my closest friends for being there for me ubiquitously; to the ones around EPFL for making this adventure one to remember, and to the distant ones for always being there for me. Thanks to my colleagues for making this journey more than just work, as we collaborated, shared our thoughts, and became friends.

Last but not least, I owe a big thank you to my family: my sister Ivana, my parents Branka and Dragomir, and my fiancée Irena, for their unconditional encouragement, support, and sincere love.

Lausanne, 1 June 2017 Milos Vasic

Abstract

THE degree of intelligence built-in in today's vehicles in constantly on the rise. The vehicles are being equipped with sensors, with the goal to estimate the state of the vehicle and the environment surrounding it. Intelligent algorithms that process the sensory data can give their output at different levels, ranging from simple warnings, to evasive maneuvers (such as emergency braking), to complete autonomy.

While it has been demonstrated that autonomous vehicles can rely solely on their on-board sensors, their performance can be optimized through cooperation with other road vehicles. Information coming from infrastructure can be fused in as well. This is where the communication between vehicles, as well as between vehicles and the infrastructure, comes into play. The main benefits of cooperation include larger coverage and extended situational awareness through sharing sensor data and vehicle intentions (trajectories).

In this thesis, we address the cooperative perception problem. To solve this problem efficiently, we construct an end-to-end framework in three steps.

First, we design an experimental platform that allows for testing our cooperative perception algorithms. In particular, we equip two fully electric Citroën C-ZERO cars with sensors, on-board computers and communication equipment. At the same time, we reproduce our platform in Webots, a high-fidelity simulation tool originally developed for mobile robots and recently upgraded for road vehicles. We develop and calibrate vehicle and sensor models with the goal to reproduce the real-world conditions as closely as possible, and in turn facilitate the deployment of algorithms developed in simulation on real cars.

Second, we design two cooperative algorithms for tracking multiple objects (cars and pedestrians) using laser and camera sensors. The key components of the algorithms are our cooperative fusion methods, which allow for fusion of data obtained from a cooperative vehicle with the data obtained from on-board sensors. The algorithms are first evaluated in simulation and tested in specific scenarios. For instance, to showcase the power of our approach in a potential application, we design an overtaking decision algorithm that uses our cooperative perception algorithm as a baseline. The overtaking application proves the added-value of cooperative

perception in situations with occluded or insufficient sensory field of view.

Third, we deploy a selected algorithm on real vehicles and validate it in real time. A distributed software framework is designed for this purpose, enabling a relatively smooth transition from simulated to real environments. Moreover, the cooperative perception algorithm is subsequently enhanced for operating in more complex scenarios. Furthermore, we develop a cooperative localization method to achieve increased accuracy in cooperative vehicles' relative localization, thus enabling our cooperative perception algorithm to work properly when deployed on moving vehicles.

Overall, we develop an end-to-end framework for cooperative perception, which unifies many different sensory technologies. Despite the end goal has always been that of deploying the framework on our test vehicles, we make substantial effort to keep it as general as possible. Our framework represents a stepping stone towards more complex, multi-vehicle automated systems.

Keywords: cooperative perception, sensor fusion, intelligent vehicles, vehicle-to-vehicle communication (V2V), vehicle-to-infrastructure communication (V2I), real vehicle deployment

Résumé

Le degré d'intelligence embarqué de nos jour dans les véhicules est en constante augmentation. Ils sont équipés de capteurs afin d'acquérir des informations sur leur état et sur leur environnement. Les algorithmes intelligents qui traitent ces informations ont un large domaine d'application, de la création de simples messages d'alerte à l'autonomie complète du véhicule, en passant par des manœuvres d'évitement tel que le freinage d'urgence.

Bien qu'il a été démontré que les véhicules autonomes peuvent se fier uniquement à leur capteurs embarqués, la performance de leur autonomie peut être optimisée par la coopération avec d'autres véhicules. De même, les informations provenant d'infrastructures peuvent aussi être fusionnées. C'est ainsi que la communication entre véhicules, et entre véhicules et infrastructures, vient jouer un rôle. Les intérêts principaux de cette coopération comprennent un champ de vision augmenté et une compréhension étendue de la situation par le partage des données des capteurs et des intentions des véhicules, par la communication des trajectoires prévues par exemple.

Au long de cette thèse, nous nous concentrons sur le problème de la perception collaborative. Pour résoudre ce problème de manière efficiente, nous construisons une architecture complète en trois étapes.

Premièrement, nous concevons une plateforme expérimentale qui nous permet de tester nos algorithmes de perception coopérative. Plus particulièrement, nous équipons deux Citroën C-ZERO entièrement électriques avec capteurs, ordinateurs de bord et équipements de communication. Simultanément, nous reproduisons ces plateformes dans Webots, un simulateur haute-fidélité originalement développé pour les robots mobiles, récemment adapté aux véhicules routiers. Nous développons et calibrons les modèles des véhicules et des capteurs afin de reproduire les conditions du monde réel aussi précisément que possible, et en conséquence faciliter le déploiement des algorithmes développés en simulation sur les voitures.

Deuxièmement, nous concevons deux algorithmes coopératifs pour le suivi de multiples objets (voitures et piétons) en utilisant lasers et caméras. Les composants clefs de ces algorithmes

sont nos méthodes coopératives permettant la fusion de données obtenues d'un véhicule coopératif avec les données obtenues des capteurs embarqués. Les algorithmes sont d'abord évalués et testés dans des scénarios spécifiques. Par exemple, afin de démontrer la portée de notre approche au sein d'une application potentielle, nous concevons un algorithme de décision de dépassement qui a pour fondation notre algorithme de perception coopérative. Cette application au scénario de dépassement prouve la valeur ajoutée de la perception coopérative dans une situation de champ de vision insuffisant ou occulté.

Troisièmement, nous déployons un algorithme choisi sur nos véhicules de test et le validons en temps réel. Une architecture logicielle distribuée est conçue à cette fin, permettant une transition facilitée de l'environnement simulé à l'environnement réel. Pour ce faire, l'algorithme de perception coopérative est étendu afin de pouvoir fonctionner dans des scénarios plus complexes. De plus, nous développons une méthode de localisation coopérative afin de parvenir à une grande précision dans la localisation relative des véhicules coopératifs, ainsi permettant notre algorithme de perception coopérative de fonctionner correctement une fois déployé sur des véhicules en mouvement.

Dans l'ensemble, nous développons donc une architecture complète pour la perception coopérative qui unifie différents types de capteurs. Bien que le but ultime ait toujours été le déploiement sur nos véhicules de tests, nous consacrons un effort important à conserver le caractère général de notre approche, autant que faire se peut. Notre architecture constitue un tremplin vers des systèmes plus complexes de plusieurs véhicules automatisés.

Mots clefs : perception coopérative, fusion de capteurs, véhicule intelligent, communication de véhicule à véhicule (V2V), communication de véhicule à infrastructure (V2I), déploiement sur véhicules réels

Zusammenfassung

DER Grad der Intelligenz in heutigen Fahrzeugen nimmt stetig zu. Die Fahrzeuge werden mit Sensoren ausgerüstet mit dem Ziel, den Zustand des Fahrzeugs sowie dessen Umgebung zu ermitteln. Intelligente Algorithmen zur Verarbeitung der Sensordaten können für Aktionen auf verschiedenen Ebenen eingesetzt werden, angefangen bei einfachen Warnungen, über Ausweichmanöver (wie z.B. Notbremsung), bis zu vollständiger Autonomie.

Obwohl gezeigt wurde, dass sich autonome Fahrzeuge lediglich auf ihre vorhandenen Sensoren verlassen können, kann ihre Leistungsfähigkeit durch Kooperation mit anderen Fahrzeugen optimiert werden. Ebenso kann Information von intelligenter Infrastruktur mit einbezogen werden. Hierzu spielt die Kommunikation zwischen den Fahrzeugen sowie zwischen Fahrzeugen und Infrastruktur eine wichtige Rolle. Die entscheidenden Vorteile einer solchen Kooperation sind grössere Abdeckung und erweiterte situative Aufmerksamkeit durch das Teilen der Sensordaten und Entscheidungen des Fahrzeugs (Trajektorien).

In dieser Arbeit wird das Problem der kooperativen Perzeption angegangen. Zur effizienten Lösung dieses Problems entwickeln wir ein End-to-End-Framework in drei Schritten.

Zuerst entwerfen wir eine experimentelle Plattform für das Testen unserer Algorithmen für kooperative Perzeption. Insbesondere statten wir zwei Elektrofahrzeuge Citroën C-ZERO mit Sensoren, Bordrechner und Kommunikationsausrüstung aus. Zugleich bilden wir unsere Plattform in Webots ab, einem für mobile Roboter entwickelten hochspezialisierten Simulationswerkzeug, welches neulich für Strassenfahrzeuge erweitert wurde. Wir entwickeln und kalibrieren Fahrzeug- und Sensormodelle mit dem Ziel, so gut wie möglich reale Bedingungen zu reproduzieren, um im Gegenzug die Anwendung der mithilfe der Simulation entwickelten Algorithmen bei realen Fahrzeugen zu erleichtern.

Im zweiten Schritt entwickeln wir zwei kooperative Algorithmen für laser- und kamerasensorgestütze Verfolgung multipler Objekte (Fahrzeuge und Fussgänger). Die Schlüsselkomponenten der Algorithmen sind unsere Methode der kooperativen Datenfusion, welche es erlaubt die Daten einnes im kooperativen Verbund befindlichen Fahrzeugs mit den Daten der Bordsensorik zu verknüpfen. Die Algorithmen werden zunächst in einer Simulation evaluiert und anschliessend in spezifischen Szenarien getestet. Um die Leistungsfähigkeit unseres

Ansatzes in einer möglichen Anwendung zu demonstrieren, entwerfen wir beispielsweise einen Entscheidungsalgorithmus für Überholungsmanöver, welcher auf unseren Algorithmus für kooperative Perzeption basiert. Diese Anwendung zeigt den Mehrwert der kooperativen Perzeption in Situationen mit einem verdeckten oder unzureichenden Sensorsichtfeld.

Schliesslich wenden wir den ausgewählten Algorithmus bei realen Fahrzeugen an und validieren ihn in Echtzeit. Für diesen Zweck wird ein verteiltes Software-Framework entworfen, welches einen relativ reibungslosen Übergang von simulierten zu realen Umgebungen ermöglicht. Zudem wird der Algorithmus für kooperative Perzeption ständig erweitert, um in komplexeren Szenarien angewendet werden zu können. Des Weiteren entwickeln wir eine Methode für kooperative Lokalisierung, um erhöhte Genauigkeit bei relativer Position der kooperativen Fahrzeuge zu erreichen, was unserem Algorithmus für kooperative Perzeption eine Anwendung auch bei bewegten Fahrzeugen ermöglicht.

Zusammengefasst entwickeln wir ein End-to-End-Framework für kooperative Perzeption, welches mehrere verschiedene Sensortechnologien vereinigt. Ungeachtet dessen, dass das Endziel immer die Anwendung des Frameworks bei unseren Testfahrzeugen war, machen wir erhebliche Bestrebungen, um es so allgemein wie möglich zu halten. Unser Framework stellt ein Sprungbrett in Richtung automatisierter Systeme hoher Komplexität mit multiplen Fahrzeugen dar.

Schlüsselwörter: kooperative Perzeption, Sensor-Fusion, intelligente Fahrzeuge, Fahrzeug-Fahrzeug-Kommunikation (V2V), Fahrzeug-Infrastruktur-Kommunikation (V2I), Anwendung bei realen Fahrzeugen

Contents

Part I Introduction

1 Intelligent Vehicles

We have been witnesses of many car accidents and fatalities arising from them. In 2015, 35,092 people died in motor vehicle traffic crashes in the United States, according to the U.S. Department of Transportation [1]. In the European Union, according to [2], the number of fatalities has a descending trend, with 26,100 fatalities in 2015, representing a 17% decrease since 2010.

Robotics and intelligent systems are getting more and more involved in the automotive industry. Vehicles are being equipped with sensors that serve to monitor the state of the vehicle and the environment surrounding it, as well as with intelligent algorithms that process the sensor data. The main goals are improving the safety of passengers and other people, as well as increasing the performance of the vehicles and traffic in general. However, we need to keep in mind that intelligent systems also introduce new types of hazards, and sometimes present a hazard themselves. A hazard can be, for example, caused by a bug in a system or a malfunction of a robotic system.

The development of intelligent vehicles spans over different levels of automation (see Figure 1.1 for an overview of automation levels). Much of the development is carried out in the domain of advanced driver assistance systems, in terms of autonomy classified as Level 1 (also known as *hands on*, the system controls either steering or acceleration/deceleration) and Level 2 (*hands off*, the system controls both steering and acceleration/deceleration) according to the SAE standard [3]. At the same time, some approaches target full autonomy, where the goal is to achieve Level 5 of the SAE standard.

1.1 Driver Assistance Systems

In order to reduce the number of accidents, there has been a lot of research conducted in the field of driver assistance. Among the most investigated fields are the braking assistance and collision avoidance. This is easily explicable based upon human behavior. More specifically, humans easily underestimate the speed of their vehicle and the distance to the vehicle in

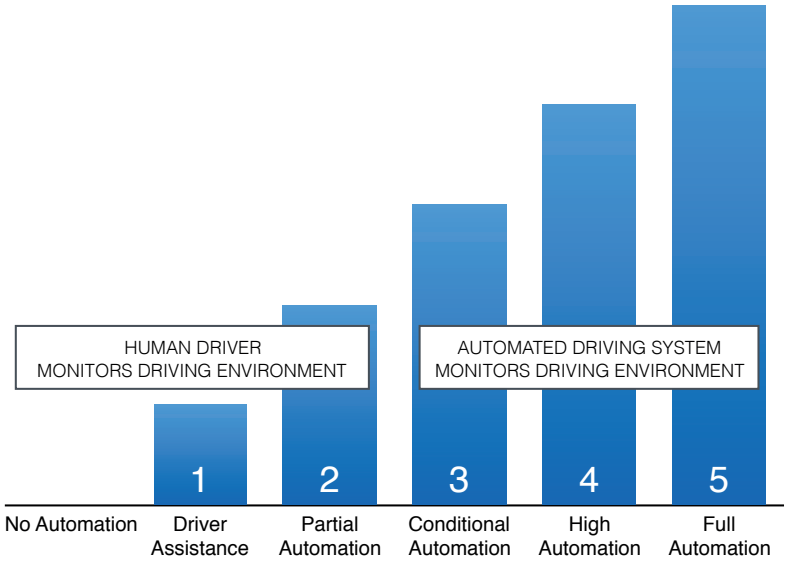


Figure 1.1 – SAE automation levels (as defined in [3]).

front of them. Since the vehicles operate at high speeds, and have enormous kinetic energy, injuries arising from car accidents could be serious and even fatal. For instance, an inattentive driver might overlook a stopped or a slowly moving vehicle ahead, or underestimate its threat level until it is too late. Another likely scenario is one of the major causes of accidents on the highways. In such scenario, the distance to the vehicle ahead is not very large (e.g., around 50 m when driving at 100 km/h) at the moment when the front vehicle suddenly starts to brake. The requirement for the driver of the rear vehicle to detect an abrupt change in relative distance and acceleration is high and this often leads to accidents due to lack of attention, tiredness, fatigue, etc. These are just some of the situations where the driver assistance systems are of crucial importance.

Assistance systems at SAE Level 0 provide warnings to the driver but do not take any control over the vehicle. One example of such systems is the AFIL technology [4] by Citroën (alerte de franchissement involontaire de ligne / lane departure warning system), which uses infrared sensors to detect a lane marker, and warns the driver when it is crossed without previously activating the turn indicator. Another good example is the Citroën blind spot monitoring system, which uses ultrasound sensors to monitor the blind spots and issues warnings to the driver [5].

Modern vehicles are often equipped with forward-looking sensors such as radars, cameras and ultrasound sensors, and implement some kind of braking assistance or collision avoidance systems. These systems are typically based on (i) time metrics, such as *time-to-collision* [6], [7], *time headway* [8], or *time-to-last-second-braking* [9], or (ii) distance metrics, such as *projected minimum distance* during a collision avoidance process [10]. They all have a common goal, which is to quantify the danger or threat level caused by the proximity of the leading vehicle

objectively, and assess the urgency level for the required braking action by the rear vehicle. Recent research takes a different approach to the same problem: Cabrera et al. proposed in [11] a *time-to-last-second-acceleration* algorithm that represents a warning algorithm for the driver of the front vehicle in a rear-end collision situation. It computes the time left before the extreme evasive action (i.e., full throttle acceleration) needs to be performed by the driver of the front vehicle, in order to avoid the rear-end collision. Adaptive Cruise Control (ACC) systems go one step further and implement a controller that follows a leading vehicle at the safe distance and brake as the leading vehicle brakes (see for example the ACC systems of Citroën [12], Mercedes-Benz [13], or BMW [14]). For vehicles that have one of the above-mentioned systems enabled, we may say that they belong to the SAE Level 1 of automation, since they have an automatically controlled braking and acceleration system.

Despite the shorthand *hands off*, in a system belonging to the SAE automation Level 2, the contact between a driver's hand and the steering wheel is often mandatory. Parking assistance systems that control both steering and speed of a car belong to this category (e.g., see remote parking aid systems by Valeo [15]). Other examples include lane change assistance systems [16] and intersection assistance [17], when complemented with a controller that performs steering and acceleration/deceleration actions instead of an Human Machine Interface (HMI). Finally, traffic jam assistance and automation systems also belong to the SAE Level 2. A good example is a DISTRONIC PLUS Stop&Go with steering assist by Daimler for the Mercedes E- and S-Class [18]. It basically merges the functionalities of an ACC system with a lateral lane guidance system.

1.2 Autonomous Vehicles

Many car manufacturers, academic laboratories that are in the field of intelligent vehicles, as well as technology companies such as Google, Uber, or Faraday Future, seek to produce totally autonomous systems. Claims that autonomous vehicles are going to decrease the number of accidents and eventually perform better than human drivers are common [19], [20]. For instance, whereas humans have the ability to track only a few objects in the environment at once, robots can do much more. In fact, the autonomous car and the software on the car are, in the long run, potentially better than people in keeping track of a lot of elements populating the surrounding environment of a vehicle [19].

State-of-the-art autonomous vehicles are equipped with a number of sensors, including, among others, radars, Light Detection and Ranging (LIDAR) sensors, cameras, proximity sensors, Global Navigation Satellite Systems (GNSSs), and inertial navigation systems (see for example [21]–[23]). They use sensory information to localize themselves in the environment and make distinctions between drivable surfaces and obstacles. Sensors need to enable vehicles to see far ahead (depending on the goal driving speed, usually at least $150-200$ m) and adjust their speed in accordance with the situation on the road in a timely fashion. Environments in which autonomous vehicles operate are highly dynamic. Many objects are not found

in maps and/or their mobility pattern cannot be accurately predicted from the current sensory information. Those objects include, but are not limited to, other vehicles on the road and pedestrians. Therefore, autonomous vehicles must have the ability to detect and determine the location of each street sign, lamp post, curb, tree, bush, and so on, as well as every moving object in the vicinity, including people, animals, cyclists, other vehicles, and stray soccer balls.

There are situations when not even humans can avoid accidents (cf. [1] for statistics). Sometimes there is just not enough time to brake. Take as an example a child chasing a ball into the street. Even when assuming the robotic car was obeying all traffic rules and was actively aware of the surroundings, upon the kid or the ball appearing from behind a fence, bush, car or other obstruction, an accident could simply be unavoidable. However, since the human reaction time is quite long, under identical circumstances, the child might have better chances to avoid the collision if behind the wheel was not a human driver, but a robot. In fact, we can expect that future robotic cars with more sensors and faster reaction times could potentially significantly outperform a human driver.

Bicycles also present a safety issue. In the worst case situation, a cyclist can fall off his or her bicycle and stop immediately, lying on the road. A vehicle following a bicycle has therefore to leave enough space to assure it can stop before such an occurrence, including reaction time. As already mentioned, reaction time should be better for robotic cars than for humans. Moreover, a robotic car would be pro-actively more conservative than a human driver who typically tends to leave less space behind any vehicle, a habit probably promoted by usual tailgating behavior behind other cars. If a car hits another car at slow speeds, it is tolerable — nobody will be seriously hurt due to well-engineered cars' passive safety systems. However, hitting a cyclist (or a pedestrian) at slow speeds can lead to serious injuries or even death. It is a real challenge to find a good balance between colliding into someone and abrupt braking because of an obstacle as small as a butterfly.

No matter how low the rate of accidents robotic cars will manage to achieve, there will always be their opponents. Questions of liability will arise — is it the fault of the car owner, vehicle manufacturer, or even of the researchers and programmers who worked on the software? As a matter of fact, the cause of the accident might be easier to prove due to all the perception and measurement equipment, which are able to record what is happening around the accident. Future regulations might ask to have a black box installed in each autonomous vehicle, similar to airplanes. The effectiveness and liability of safety measures for autonomous vehicles will play a key role in their societal acceptance in the years to come. However, this and other ethical issues are beyond the scope of this thesis.

Summary

Nowadays, more and more vehicles are being equipped with sensors and intelligent algorithms processing their data. Work is being carried out across different levels of automation, spanning from warning, controlling acceleration/deceleration and/or steering, to fully autonomous vehicles. While the main goals of this initiative are to improve the safety and performance of vehicles and traffic in general, intelligent systems also introduce new types of hazards. Improving robustness and fault-tolerance of these systems is still in the hands of researchers.

2 Networked Vehicles

THE concept of connected vehicle enables automated links from the vehicle to all other connected objects, including smartphones, tracking devices, traffic lights and other motor vehicles. Developments related to connected vehicles are centered around seven functional areas [24]:

- **Autonomous driving** — Operation of a vehicle without a human driver at the controls.
- **Safety** — Providing warnings to the driver with respect to road problems and automatically sense and prevent potential collisions.
- **Entertainment** — Functions that provide multimedia content to the passengers and the driver.
- **Well-being** — Assistance related to driver's health and competence, such as fatigue monitoring.
- **Vehicle management** — Remote control of car features, as well as display of service and vehicle status.
- **Mobility management** — Real-time traffic information displays and guidance on faster, safer and more economical driving.
- **Home integration** — Links between a vehicle and buildings such as homes and offices.

Some of these functional areas rely on the car having Internet connection. In this thesis, we will focus on the areas which can operate when only Vehicular Ad-Hoc Networks (VANETs) are available. Local communication encompasses direct communication between two vehicles — known as Vehicle-to-Vehicle (V2V) communication, and Vehicle-to-Infrastructure (V2I) communication, which assumes communication between an intelligent infrastructure node (e.g., a traffic light) and a vehicle, but also communication from the vehicle to the infrastructure. A common term Vehicle-to-Everything (V2X) communication comprises communication

between vehicles, infrastructure and service providers. We refer to vehicles satisfying the conditions above as **networked vehicles**, representing a sub-category of connected vehicles.

The number of vehicles having some sort of communication device is on the rise. Already, most new cars are equipped with sensors and connected to high-speed wireless networks. One key catalyst for this is the European Union's mandate that all auto makers implement emergency calling technology (eCall) in new cars by 2018 [25]. When a collision or other incident happens, an eCall device in each car will automatically alert authorities and send data about the impact.

Although it is demonstrated that autonomous road vehicles can rely solely on their on-board perception sensors, it is foreseen that they will greatly profit from the introduction of a V2X communication. V2X communication can help expand situational awareness for both human drivers and automated driving systems. Some benefits of cooperation include increased availability and reliability in cooperative relative positioning, as well as cooperative perception by sharing sensor information and the execution of collaborative maneuvers between automated road vehicles. In this way, a higher degree of safety is achievable without sacrificing efficiency by driving with large safety distances and increased caution.

On the global level, V2X communication can significantly enhance safety and traffic fluidity. Information about traffic can be collected, aggregated, and used for planning or control of traffic. Until common communication protocols are established and the sufficient penetration of communication-enabled vehicles to the market is achieved, benefits will mostly be reflected in an increase of individual vehicle's safety and performance. For example, a vehicle may be able to optimize its trajectory and thus save time and/or energy resources. There exist different studies and proposals from the European Commission and organizations (CAR 2 CAR Communication Consortium for instance) targeting legislation and standardization of V2X communication. The United States Department of Transportation has recently proposed a rule to require V2V communication technology in new cars [26].

Vehicle cooperation techniques can be split into two groups, as shown in Figure 2.1: (i) cooperation on perception and (ii) cooperation on control. This thesis focuses on cooperative perception.

2.1 Cooperative Perception

V2X communication shows potential for functions such as intersection assist, left-turn assist, do-not-pass warning, advanced warning of a vehicle braking ahead, forward-collision warning, and blind-spot/lane-change warning [28]. To implement these functions, a vehicle can exchange locally available information with other vehicles in its proximity, which might help it to make a more intelligent and safer decision about its action. An example of a car following system based on inter-vehicular communication is given in [29]. A recent survey of cooperative localization techniques can be found in [30]. In his PhD thesis, Li covered topics such as cooperative localization, cooperative detection of moving objects and cooperative

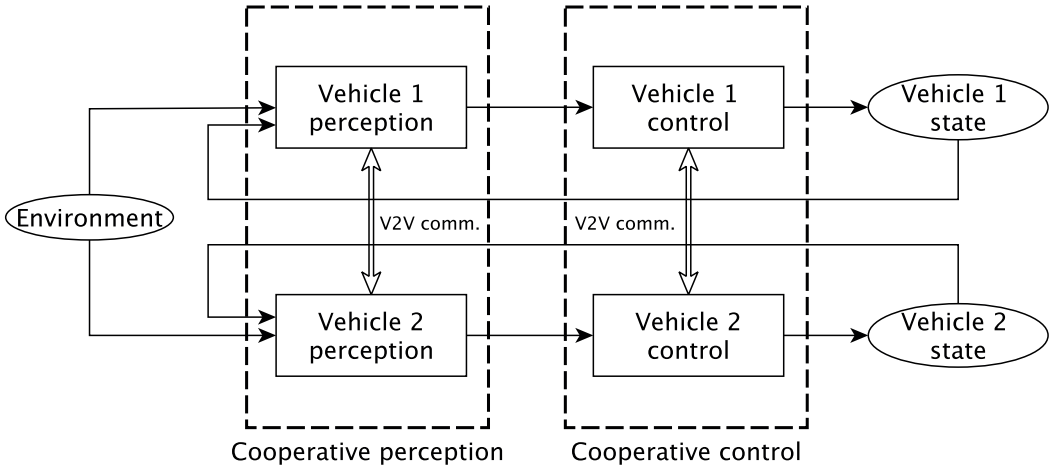


Figure 2.1 – Vehicle cooperation paradigm (adapted from [27]).

augmented reality [27]. As perception sensors often aid the localization process and increase its accuracy, localization can be put in the perception group.

The focus of this thesis is on cooperative perception techniques for intelligent vehicles. Let us consider a situation in which, due to obstructed view, the vehicle might overlook an approaching dynamic obstacle (e.g., another vehicle or a pedestrian) and in turn make an unsafe decision. This thesis aims to show that, if the two vehicles exchange their sensor data among themselves (raw or some processed representation of it), it is possible to gain a more complete and accurate situational awareness through the results of a sensor data fusion process.

Works with similar ideas can be found in the literature. In [31], an infrastructure sensor network is deployed at an intersection, and data from multiple LIDARs and cameras are gathered and processed on a central computer. An approach based on distributed dynamic maps for cooperative perception was presented in [32]. However, the performance evaluation was carried out in simulation only, the level of realism of which is unclear (e.g., how sensors are simulated). In [33], a cooperative perception algorithm for autonomous driving using V2V communication is presented. It is based on map merging, and raw LIDAR data are shared along with the camera image (it was experimented with raw, compressed, or processed image). The positive effects of cooperative perception on autonomous driving in terms of decision making and planning were demonstrated in [34]. Rauch et al. presented in [35], [36] a high-level architecture for cooperative perception systems. In their work, they used the concept of V2X communication being a remote sensor. They address the temporal and spatial alignment problems, which are important subproblems of cooperative perception; however, they do not propose any algorithm for track-to-track fusion and do not deal with the track-to-track association problem.

Our goal is to develop a framework for cooperative perception, which integrates multiple

sensor modalities but only requires communicating the processed data (thus reducing the requirements in terms of the communication bandwidth and making the approach scale better). We aim to operate the framework on moving vehicles in real-world scenarios.

2.2 Cooperative Control

Developing cooperative strategies for groups of future intelligent vehicles is one of the main objectives in the automotive industry aiming at improving road traffic flow and safety, reducing CO_2 emissions and enhancing driving comfort. Grouping neighboring vehicles into formations is one possible way to achieve these objectives. The formation is called platoon if spanning over a single lane, or convoy if spanning over multiple lanes.

Many previous research projects in this field (e.g., PATH [37], CHAUFFEUR I and II [38], and SARTRE [39], [40]) have proven that fuel consumption could be decreased by 15% to 30% while the road throughput on motorways can potentially be multiplied by a factor of 3 to 5 through group control of vehicles.

In the scope of the European project AutoNet2030 to which Distributed Intelligent Systems and Algorithms Laboratory (DISAL) participated, we developed an algorithm for distributed control of dynamic vehicle convoys, which spread over multiple lanes [41], [42]. In this work, local V2V communication was used, and any given vehicle needed to be aware only of other vehicles in its vicinity. Vehicles were able to join or leave the formation, which dynamically reconfigured. We extended this approach to allow for convoys of heterogeneous vehicles [43]. A heterogeneous convoy accepts vehicles of different type and length (e.g., cars and trucks). This approach used curvilinear coordinates in order to reduce longitudinal formation errors in curves.

Makarem and Gillet presented a decentralized algorithm for coordination of autonomous vehicles at intersections [44]. This coordination prevents simultaneous deceleration of two vehicles approaching an intersection, thus reducing energy consumption. In the paper of Debada et al. was shown that the throughput at roundabouts can be increased by applying distributed algorithms to coordinate autonomous vehicles [45]. A virtual vehicle approach was used for coordination of intelligent vehicles in roundabouts in the presence of non-cooperative vehicles [46].

Summary

Although it is demonstrated that autonomous road vehicles can rely solely on their on-board perception sensors, it is foreseen that they will greatly profit from the introduction of V2X communication. Cooperation between vehicles can be established at the level of perception and/or control. Cooperative perception enables participating vehicles to extend their situational awareness, by fusing information obtained from their own sensors with the information communicated by cooperative vehicles in their vicinity. Cooperation in terms of control means that vehicles possess more information about states and intentions of the vehicles surrounding them. This information can be used for optimization of trajectories and results in energy savings and traffic flow enhancement.

3 Scope of the Thesis

THE purpose of this chapter is to define the thesis statement, detail our main objectives and list important challenges. Additionally, it provides the outline of the thesis to give each part its purpose. Finally, it points out our research contributions and related publications.

In the previous two chapters we provided an insight into developments in the areas of intelligent and cooperative vehicles. We mentioned driver warning systems, collision avoidance systems, cruise control systems, as well as distributed multi-vehicle systems at intersection and on highways (platoons and convoys). Figure 3.1 lays down on a 2D chart different intelligent vehicle applications as a function of their autonomy and cooperation levels. It also indicates the position of this thesis on the chart. The approach we take in this thesis is to exploit cooperation among intelligent vehicles for general-purpose sensing tasks. The outcomes of our algorithms can be useful in many traffic situations, and various autonomous systems can be built on its foundations. We demonstrate one such application. The main claim of this thesis can be summarized as follows:

> Cooperation among vehicles equipped with noisy sensors increases their coverage area and enhances their situational awareness, thus providing ground for improving their decision-making performance and ultimately safety.

3.1 Objectives and Outline

The main objective of this thesis is to develop the end-to-end framework for cooperative perception. The specific objectives are defined as follows:

1. identify components relevant for an experimental platform;
2. build the experimental platform and reproduce it in simulation;

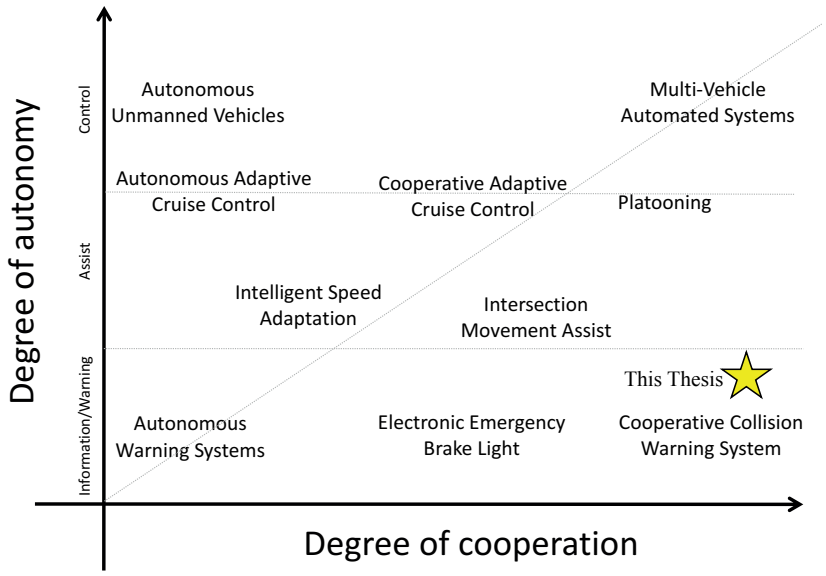


Figure 3.1 – Intelligent vehicle applications as a function of vehicle intelligence and inter-vehicle cooperation (adapted from [47]).

3. develop the core algorithms for cooperative perception and measure their performance;
4. deploy and operate the algorithms on real vehicles and demonstrate their effectiveness in real-world scenarios.

To achieve the aforementioned objectives, a number of challenges needs to be addressed. They relate to noisy sensing, occlusion and clutter, multi-measurement to multi-object and multi-object to multi-object association, inaccurate relative localization, and integration of work into an end-to-end framework.

This thesis is laid down in five parts:

Part I - Introduction In this part, we have briefly introduced the problems encountered in the fields of intelligent and networked vehicles. We have shown and explained why these fields are important for future traffic systems. Additionally, we laid out the problematics linked to this thesis and clearly exposed our main objectives.

Part II - Tools and Platforms In the second part, we introduce our platform consisting of the two Citroën C-ZERO vehicles equipped with automotive sensors, computation and communication equipment. Furthermore, we introduce the simulation tool that we rely on in this thesis: Webots, a high-fidelity robot and intelligent vehicle simulator. We calibrate the simulator using real cars and real sensors in an attempt to provide a smooth transition of the developed algorithms to real vehicles.

Part III - Cooperative Perception This part lays out our main contributions. We build a framework for tracking of multiple objects from a moving vehicle and fusion of multi-object posteriors in a cooperative fashion. We evaluate the algorithms in Webots. Moreover, we develop an overtaking assistance system based on our cooperative perception algorithm, demonstrating one of its possible applications.

Part IV - Algorithmic Enhancements and Validation using Real Vehicles This part illustrates what needs to be done to deploy our framework on real vehicles and how it behaves in the real world. This part also presents the localization approach adopted and the corresponding developed algorithms. In particular, it explains the steps we took to improve the accuracy of the relative localization of our heterogeneous platforms.

Part V - Conclusion Finally, we summarize this thesis and restate the benefits and limitations of our framework. We also describe possible promising directions for pursuing further the research presented in the thesis.

3.2 Research Contributions

The work done in the scope of this thesis has led to several contributions and related publications.

Part I - Introduction The first contribution concerns a literature survey on safety issues in human-robot interactions. The survey, among other topics, covers safety related to intelligent and robotic vehicles. A part of the survey is used in this part.

- **M. Vasic** and A. Billard, 'Safety issues in human-robot interactions', in *IEEE International Conference on Robotics and Automation*, 2013, pp. 197–204.

Part II - Platforms and Tools Our next contribution concerns a design of an experimental platform that can support the evaluation of cooperative perception algorithms. We also develop simulated models of vehicles and sensors, that we calibrate using real hardware, as to minimize the simulation-to-reality gap. This framework enables us to develop algorithms in controlled, simulated environments, and smoothly transfer them to real vehicles.

- **M. Vasic**, G. Jornod and A. Martinoli, 'From high fidelity simulation to real vehicles: a flexible software framework for networked intelligent vehicles', *in preparation*, 2017.

Part III - Cooperative Perception We develop two algorithms for cooperative tracking of multiple objects for intelligent vehicles. Both algorithms use a Probability Hypothesis Density (PHD) filter as their backbone for multi-object tracking. The first algorithm exploits a Gaussian Mixture approximation of the PHD function, while the second one uses a Sequential Monte Carlo approximation. The novelty in our algorithms is their ability

to fuse estimates originating from two intelligent vehicles whose Fields Of View (FOVs) overlap only partially (the sensors of each vehicle cover a different region). PHD intensities are communicated between cooperative vehicles and fused in a distributed fashion using the approaches we developed. Finally, we develop an overtaking recommendation algorithm that uses the cooperative perception algorithm. It yields safer vehicle operation and outperforms the algorithm version in which the overtaking vehicle relies on its own sensors only.

- **M. Vasic** and A. Martinoli, 'A collaborative sensor fusion algorithm for multi-object tracking using a Gaussian mixture probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2015, pp. 491–498.
- J. Gan, **M. Vasic** and A. Martinoli, 'Cooperative multiple dynamic object tracking on moving vehicles based on sequential Monte Carlo probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2016, pp. 2163–2170.
- **M. Vasic**, G. Lederrey, I. Navarro and A. Martinoli, 'An overtaking decision algorithm for networked intelligent vehicles based on cooperative perception', in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 1054–1059.

Part IV - Algorithmic Enhancements and Validation using Real Vehicles With a desire to deploy and validate our algorithms on real vehicles, we carry out additional algorithmic enhancements. More specifically, we perform an optimization of the fusion weight with the goal to improve the algorithm performance. We develop a flexible software architecture which enables us to deploy our algorithms on real vehicles in two phases: (i) we first implement and test the necessary modifications in simulation, and (ii) we replace simulated vehicles and sensors with the real ones. During the validation of our approach, we propose two algorithms for cooperative localization and a novel method of fusion of LIDAR and camera sensors.

- **M. Vasic**, D. Mansolino and A. Martinoli, 'A system implementation and evaluation of a cooperative fusion and tracking algorithm based on a Gaussian mixture PHD filter', in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 4172–4179.
- **M. Vasic**, G. Jornod, J. Løje and A. Martinoli, 'An end-to-end framework for cooperative perception based on a GM-PHD filter', *in preparation*, 2017.

Addressing this diverse list of topics implied a collaborative endeavor, and although I was directly involved in all of the work presented in this thesis, I would like to give credit to my close collaborators from the DISAL laboratory. In particular, R&D Engineer David Mansolino led the modeling and calibration aspect of the work presented in Chapter 5. David also assisted with experiments and data processing presented in Chapter 14. Master student Jonathan Gan worked on the fusion algorithm presented in Chapter 9 under my supervision. A first version of the overtaking algorithm presented in Chapter 10 was developed in the course of the semester

project of Master student Gael Lederrey, co-supervised by Post-Doctoral fellow Iñaki Navarro and myself, and was subsequently improved and put in the framework of the cooperative perception algorithm. R&D Engineer Guillaume Jornod and Master student Johannes Løje assisted with the deployment and experimental evaluation presented in Chapter 13 and Chapter 16.

Finally, the credit goes to our industrial partner PSA Groupe for their financial and technical support, and in particular for their interactions leading to strategic choices that were essential for this thesis, both in terms of system design and research topics.

Summary

This thesis focuses on cooperative perception algorithms for networked intelligent vehicles. It aims to provide a flexible framework for distributed fusion of object posteriors obtained on individual vehicles using different automotive-grade sensors. A very important ingredient for cooperative fusion is the accurate relative localization of the participating vehicles. This thesis thus contributes to increasing the localization accuracy as well. We develop our algorithms in a high-fidelity simulator, and only deploy them on real vehicles in a second step. This thesis thus contributes to development of simulation models and a flexible software framework, aiming at reducing the simulation-to-reality gap, and therefore achieving a smoother transition from simulated to real environments.

Part II Platforms and Tools

4 Experimental Platform

OUR experimental platform is composed of two Citroën C-ZERO vehicles. They are zero-emission electrical vehicles which makes them very suitable for work both outdoors and inside the workshop. Their range is around 90 km on one charge which is sufficient for the whole-day operation in our experimental settings. In the remainder of this section, we first present different sensor technologies considered for achieving our research objectives and finish by showing their actual deployment on our vehicles.

4.1 Perception Sensors

Perception sensors are exteroceptive and are used for the detection of objects in the vicinity of the vehicle. The most commonly used sensors are LIDARs cameras, radars, and ultrasound sensors. In our research, we use the former two.

4.1.1 LIDARs

A Light Detection and Ranging (LIDAR) sensor is a scanning measurement device which uses a laser to transmit a coherent light pulse and a receiver to measure the reflected light. The distance to the object is determined based on the time-of-flight principle, by recording the time between transmitted and received pulses and multiplying it by the speed of light to calculate the distance traveled. In this manuscript we will often use the word LIDAR for referring to the actual device.

We own four Ibeo LUX LIDARs[1]. The LUX LIDAR is characterized by the maximum range of 200 m, and the maximum horizontal FOV of 110°. The actual range depends on the reflectivity and the size of an object, and the horizontal FOV on the selected scanning frequency. It contains four vertical layers with a total FOV of 3.2°; two layers are tilted towards the ground and they cover the area below the horizon, whereas the other two layers are tilted upwards and

[1] https://www.ibeo-as.com

cover the area above the horizon. For this reason, Ibeo LUX can be categorized as a 2.5D sensor. Ibeo LUX operates at scanning frequencies of up to 50 Hz; we have configured it to work at 12.5 Hz in order to obtain the maximal horizontal FOV. The scanning schedule of all Ibeo LUX sensors deployed on the same vehicle is synchronized with the help of a synchronization unit. A central computer merges individual point clouds into a joint point cloud. It also flags points reflected from the ground or from *fake* objects such as dirt particles or rain drops as clutter. The merged point cloud is a set

$$Z = \bigcup_i \mathbf{z}_i, \tag{4.1}$$

where an individual point z is given by its 3D Cartesian coordinates and the clutter flag f:

$$z = \left[\begin{array}{ccc|c} x & y & z & f \end{array} \right]^\top. \tag{4.2}$$

According to the Ibeo specification sheet, the accuracy of a LUX sensor ray is 10 cm (one standard deviation) in range, and is independent of the actual range. The angular resolution of the sensor is 0.25°, which makes the point cloud less dense as the distance of the object from the sensor is increased. It is a very accurate sensor with relatively low noise and a few clutter points per scan. It can be used for detecting and tracking different objects such as vehicles or pedestrians. The classification of objects based on the LIDAR point cloud only is difficult, as the size and shape of detected objects vary with the perspective and are subject to occlusions.

4.1.2 Automotive Cameras

There exist many cameras on the market that can be used in automotive applications. However, processing images from cameras is a tedious job that requires a lot of time and computational resources. A good alternative to these systems is a **Mobileye** camera — a monocular camera that comes with a specialized system on chip on which the image processing is performed in real time.

Our version of the Mobileye camera is connected to a computer over a Control Area Network (CAN bus), the throughput of which is 500 kbps. A Mobileye camera does not provide an image. Instead, it works like a black box which sends a list of detected objects of certain supported classes and their poses. Supported object classes are cars, trucks, pedestrians, bicycles and motorcycles.

Limited details about the camera system can be found in [55]. After processing each frame using the software embedded in its chip, the camera sends the data describing objects over the CAN bus in multiple consecutive binary-encoded messages at approximately 10 Hz. We synchronize all messages related to a single image frame and decode them to create a set Z of

measurements **z** that relate to separate objects present in that frame:

$$Z = \bigcup_i \mathbf{z_i}, \tag{4.3}$$

where

$$\mathbf{z} = \left[\begin{array}{ccc|c} x & y & \theta & c \end{array} \right]^\top \tag{4.4}$$

is a measurement of a single object, and describes its center point (x, y), orientation θ and class c (e.g., car or pedestrian). The Mobileye camera can only detect objects for which it was trained, as mentioned above — other classes of objects or the raw camera image are not available. It is important to mention that our version of Mobileye camera only detects cars from the front or from the back (cars facing sideways or under a large angle are not detected). This greatly limits the choice of a scenario when this sensor is involved.

In the absence of detailed camera specifications, the important tracking-related camera parameters are determined experimentally. Based on our case study, involving two cars with Mobileye cameras and two legacy cars which drive in the opposite direction, we characterize the camera (see Chapter 14 for the description of our case study). We compare the output of the Mobileye camera to a point cloud obtained by an Ibeo LIDAR. We consider the point cloud to be the ground truth, given its significantly higher accuracy as compared to the Mobileye camera. We determine that the camera's FOV is 45° (from −22.5° to 22.5°). Its maximum range is between 50 and 70 m, and depends on the size of the detected object. While the lateral object position accuracy is relatively good, the longitudinal distance suffers from larger errors. This is most probably due to the fact that there is only one imaging device, so the distance has to be inferred from the object size in the picture. Since cars exist in different sizes, and pedestrians are of different height, the accuracy in the distance measurement is limited.

Camera Calibration

During our experiments we noticed a significant bias between the camera measurements and the ground truth. We have therefore re-calibrated the output of the two cameras for our scenario with respect to the LIDAR ground truth data. We have determined that a second-order polynomial function provides sufficiently good fit for the longitudinal distance, while for the lateral distance and the orientation only an offset is applied. Although the two cameras we use are identical, they were initially calibrated directly on vehicles using the automatic calibration procedures, thus their calibration parameters are slightly different.

4.2 Localization Sensors

Localization of moving vehicles is very important for a number of reasons. In tracking applications, it is typical to assume some kind of motion for tracked objects (e.g., an assumption can

be that a tracked car will continue to move straight with constant velocity). Perception sensors are mounted on the body of the tracking (ego) vehicle and the tracking is naturally performed in the ego vehicle's coordinate frame. However, the motion of the ego vehicle easily invalidates the assumed track motion model, if it is not compensated for. It is therefore very important to be able to estimate the motion of the ego vehicle, which can naturally be obtained directly from the vehicle's localization solution.

In outdoor applications, global localization is usually achieved using a system of satellites, which transmits time signals by radio to electronic receivers, allowing them to determine their location. GNSS is a generic term for satellite navigation systems. This term includes satellite systems managed by different regions or countries, such as for example the Global Positioning System (GPS) (American), Galileo (European), GLONASS (Russian), and BeiDou (Chinese). We will use the generic term GNSS whenever possible, except for when we speak about one particular satellite receiver that can only receive GPS signals.

In cooperative applications, localization is even more important, as it provides a global reference when the data is converted from the local coordinate frame of one car to the local coordinate frame of another car. A logical choice for the global frame in outdoor applications is the the Universal Transverse Mercator (UTM), which is a set of projections of GNSS positions (given in latitude and longitude) clustered to 60 zones to a set of 2D Cartesian coordinate systems. A given zone corresponds to an area covering 6° of longitude in width (corresponding to roughly 800 km). For example, Switzerland almost entirely lies in UTM zone 32, with the exception of the western part of the Canton of Geneva which is in UTM zone 31.

4.2.1 Low-End Localization System

Our low-end localization solution consists of four different low-cost sensors, that we fuse together as explained in Chapter 15. Below we provide details about each sensor.

U-blox 5 GPS receiver is a low-cost global localization solution. It was readily built-in in our on-board computer and was thus a logical and cheap option. It provides global localization accurate to about 5 m. Accuracy additionally depends on the position of the GPS satellites at the time the data was recorded and the characteristics of the surroundings (buildings, tree cover, valleys, etc.) as multi-path signals arriving with a small time delay at the receiver result in inaccurately calculated position. It is accessed through a serial-to-USB protocol and it provides measurements with a frequency of 1 Hz.

Yocto-3D is a low-cost Inertial Measurement Unit (IMU) that integrates an accelerometer, a gyroscope and a compass in a single package. It is characterized by a substantial amount of noise and unreliable behavior, especially when the compass is in the proximity of metal objects. It connects to the on-board computer using the Universal Serial Bus (USB) and provides measurements with an average frequency of 17.5 Hz.

Bosch Yaw Rate Sensor YRS 3 is a factory-installed sensor in the Citroën C-ZERO. It provides

its output at a frequency of 15 Hz, which is directly available on the vehicle's CAN bus, which can be accessed through a service On-board Diagnostics (OBD-II) port.

The vehicle's longitudinal (forward) speed generated by a factory installed **speedometer** is also available on the vehicle CAN bus. We access it in the same way as the YRS 3 sensor, albeit at the rate of 12.5 times per second.

4.2.2 High-End Localization System

We own a high-end localization system **Applanix POS LV 120**[2]. It consists of two GNSS antennas, a high-accuracy external optical wheel encoder (odometer), and a MEMS-based IMU. All sensors are fused together in a black-box fashion, and an accurate localization solution is available from the Applanix filter at 100 Hz. We decided to use the Applanix output at 20 Hz, considering that the output of our fastest system does not exceed 18 Hz. Additionally, the system supports Real Time Kinematic (RTK) corrections from a system of land-based stations. Using RTK, the accuracy of the system when stationary reaches the centimeter level. However, when moving, the error grows to around 0.3 m.

4.3 Computational Equipment

In the luggage compartment of our vehicles, we installed the on-board computers.

The first computer model is a **FleetPC-7**, a fanless PC that can be connected directly to a 12 V battery. It is powered by an Intel Core i7 2710QE dual-core CPU running at 2.1 GHz, has 8 GB of RAM memory and a 512 GB Solid-state Drive (SSD).

The second computer model is a **FleetPC-8**. Similar to the first model, it connects directly to a 12 V battery. It is powered by an Intel Core i7-3517UE dual-core CPU running at 1.7 GHz, has 8 GB of RAM and a 512 GB SSD drive.

Both PC models are rich in input-output ports, which was an important design choice for our system given the variety of sensors running different interfaces. SSD drives and fanless cooling technology represent a good vibration-resistant option for automotive applications.

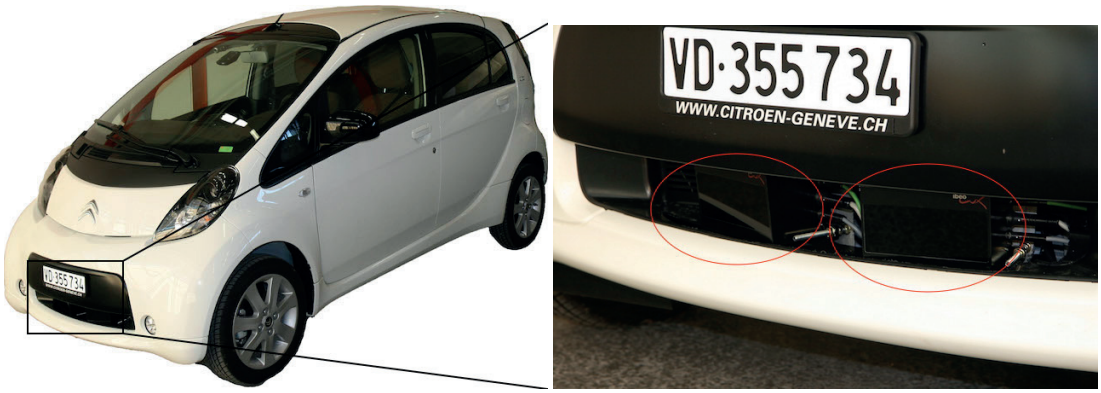
4.4 Deployment

In order to perform cooperative tasks, at least two vehicles are needed. We have equipped two electric Citroën C-ZERO vehicles with the previously mentioned sensors, computers and communication equipment. Our deployment has evolved over two different phases, which we describe subsequently.

[2] https://www.applanix.com/

(a) `car0` **(b)** `car1`

Figure 4.1 – The two Citroën C-ZERO vehicles used in the first deployment phase.



(a) `car2` **(b)** LIDARs

Figure 4.2 – (a) The new Citroën C-ZERO used in the second deployment phase. (b) Two LIDARs integrated in the front grille of the C-ZERO.

4.4.1 First Phase

The initial phase of our deployment consisted of two Citroën C-ZERO cars shown in Figure 4.1, as described below. They were used in experiments described in Chapter 14.

We had one prototype C-ZERO car which was not allowed to have license plates. We will often refer to this car as the *yellow car* or `car0`.

On this car we deployed four Ibeo LUX LIDARs, two of which were covering the front, and the other two the rear of the vehicle. We additionally installed a Mobileye camera in the top-middle region of the windshield, behind the rear-view mirror.

In terms of the localization sensors, we deployed the low-cost solution mentioned above. However, it was never used in the first phase, so we will describe it below in the framework of the Phase 2.

Table 4.1 – Deployment of equipment on test vehicles.

	Phase 1		Phase 2	
Equipment	car0	car1	car1	car2
FleetPC-7	✓	✗	✗	✓
FleetPC-8	✓	✓	✓	✓
Wi-Fi communication	✓	✓	✓	✓
Ibeo LUX LIDAR system	✓	✗	✓	✓
Mobileye camera	✓	✓	✓	✓
Applanix localization system	✗	✓	✓	✗
U-blox GPS receiver	✓	✗	✗	✓
Yocto-3d IMU	✓	✗	✗	✓
Vehicle CAN bus sensors	✓	✓	✓	✓

The other C-ZERO has a regular license for circulating on public roads; it will often be referred to as the *green car* or `car1`. In terms of perception sensors, this car only hosted the Mobileye camera. In terms of localization sensors, we have installed on this second car the Applanix localization system. The two GNSS antennas have been placed on the roof at a distance of around 2 m from each other. This separation is needed to meet the requirement for extracting the heading from the dual GNSS setup. The IMU is mounted in the center of the vehicle's rear axle, facing the forward driving direction. The odometer is mounted on the rear left wheel. The distance vectors between the IMU, odometer and the GNSS receivers had to be calculated very accurately. For this reason, we placed reflective markers on our equipment and used a motion capture system available at our laboratory for achieving a millimeter accuracy in the placement of these components.

Finally, the `car0` hosts one FleetPC-7 and one FleetPC-8 computer, whereas the `car1` hosts a FleetPC-8 computer. A high-speed wireless router in the luggage compartment of the `car0` represents a backbone of the cars' network, to which both computers are connected through an Ethernet cable. The computer in `car1` connects to the router over WiFi, and so does any laptop used for accessing the car network. All the equipment is powered by two large capacity deep-cycle external batteries, one per vehicle.

An overview of the complete deployment in Phase 1 is given in Table 4.1.

4.4.2 Second Phase

In our second (and current) phase of deployment, we replaced the `car0` by a new C-ZERO car, which has license plates and can thus circulate on public roads (see Figure 4.2a). We will refer to this car as the *white car* or `car2`. Besides the `car2`, in this phase `car1` is used as well. This physical setup is used in the experiments explained in Chapter 15 and Chapter 16.

Having in total four Ibeo LIDARs and two Mobileye cameras, we made the perception system

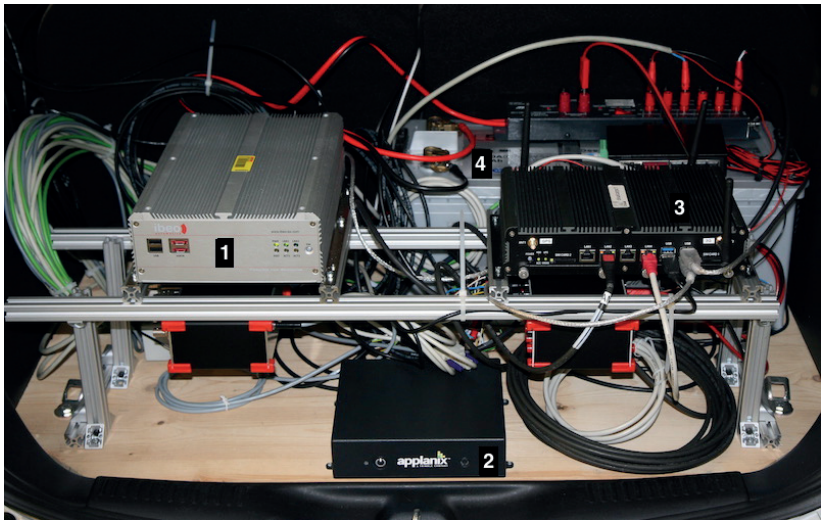


Figure 4.3 – The trunk of `car1` in the Phase 2 of the deployment. 1) Ibeo LUX control unit. 2) Applanix control unit and IMU. 3) FleetPC-8 computer. 4) 12V DC deep cycle battery.

homogeneous across the two cars: we have installed two Ibeo LIDARs and a Mobileye camera on each of the vehicles. The two LIDARs were integrated in the front grille of the car, below the bumper (see Figure 4.2b). Their joint FOV is approximately 160°, measured from the front center point of the vehicle. The Mobileye camera was installed in the same way as in Phase 1.

The localization sensors on `car1` remained unchanged since the first development phase. `car2` got endowed with all localization sensors that were initially on `car0`. In particular, the u-blox GPS receiver was installed on the roof of the car. As the accuracy of this receiver is not very high, its exact position on the roof was not crucial. Additionally, we installed the Yocto-3d IMU on the roof, above the rear axle of the car. This decision was made based on the desire of limiting the influence of any electro-magnetic radiation sources, such as the vehicle battery positioned under the trunk. The factory built-in yaw rate sensor and the odometry sensor are accessed through a vehicle's CAN bus, which is connected to the OBD-II port located below the steering wheel.

The computer setup on the `car1` remained unchanged, while the computers from `car0` got moved to the `car2`. Figure 4.3 shows the equipment installed in trunk of the `car1`.

An overview of the complete deployment of Phase 2 is equally given in Table 4.1.

Summary

In this chapter we provided details about the perception and localization sensors we have used in this thesis. We presented our experimental platform, which has undergone two deployment phases, both of them leveraging two C-ZERO electric cars. In the first phase, one of them was a factory prototype car, which we endowed with four LIDARs and a Mobileye camera and had no license plates. The second car was a regular market car, on which we deployed a Mobileye camera and an accurate localization system. In the second phase, we replaced the prototype car by a second regular car. The sensor deployment was made as uniform as possible across the two cars, which allowed for a richer set of experiments.

5 A High-Fidelity Simulation Tool

THERE exist multiple reasons why researchers often resort to simulation tools. Firstly, real platforms can be very expensive or unavailable in a sufficient quantity. Secondly, real sensors provide limited accuracy, and the real world environment is ofter very complex and noisy. It is therefore simpler to work in simulation, where the accuracy of different sub-systems can be set independently of the rest of the system. In this way, algorithms can be developed starting from ideal conditions, which can subsequently be gradually made more realistic. Due to their simplicity, repeatability and reliability, simulation tools allow for faster prototyping than when working with real hardware directly. However, it is often important that in the simulation toolbox there is at least a tool that allows for reducing as much as possible the simulation-to-reality gap, in order to facilitate the transition of the developed techniques and algorithms from simulation onto real hardware.

In this work, we rely on Webots, a single but flexible high-fidelity simulation tool for mobile robots and intelligent vehicles [56], [57]. Together with Cyberbotics Ltd. we have worked on additionally bridging the gap between simulation of mobile robots and intelligent vehicles [58] by developing calibrated vehicle and sensor models. The focus of our efforts is heavily influenced by the physical hardware we have at our disposal (cf. Chapter 4).

5.1 Simulation of Vehicles

Prior to our work, building a complete vehicle model in Webots was cumbersome. A vehicle was composed of basic components, such as a vehicle body, wheels, and joints connecting them to the body, as well as rotational motors and sensors. Each of the motors had to be controlled separately. For example, turning required controlling both left and right steering motors at the same time; acceleration was achieved by applying positive torque onto drive wheels, while for braking negative force had to be applied.

We developed a generic vehicle model based on the Ackerman steering model [59]. The model is parametrized by vehicle's track and wheelbase, and it accepts an arbitrary shape for the

vehicle body, which can be composed of simple polygons or imported from a Computer-Aided Design (CAD) file. The control of vehicles is simplified through the usage of two libraries we developed for this purpose. Additionally, we made available in Webots the complete model of a Citroën C-ZERO car, starting from original CAD files provided by PSA Gruope, appropriately reduced to allow for a reasonable computational cost of the simulation.

5.1.1 Electric Motor Model

A Citroën C-ZERO has an entirely electric traction motor with 49 kW power, which can develop the maximum torque of 180 Nm, while the maximum rotational speed of the motor is 8000 r/min [60]. It only has one gear which is able to drive the motor over its entire speed range. A characteristic of electric motors is the regenerative braking or a negative torque applied by the motor when a throttle pedal is released in order to return the excess of the energy back to the battery.

Electric motors are characterized by a maximum torque at low motor speeds. For generating our simulation model, we have identified three different operational modes on the performance curve. At low motor speeds, the motor torque output is limited by the total available torque ($\tau_{\max} = 180$ Nm). Between the rotational motor speed of $v_{\text{rpm}} = 2600$ r/min and $v_{\text{rpm}} = 8000$ r/min, the motor torque is limited by total available power $P_{\max}$. Finally, the curve is limited by the motor speed (8000 r/min). Thus, for v_{rpm} in range from 0 to 8000 r/min, the motor output torque τ is given by

$$\tau = \min\left(\tau_{\max}, \frac{P_{\max}}{2\pi \cdot v_{\text{rpm}}/60}\right). \tag{5.1}$$

The model is shown in Figure 5.1.

5.1.2 Libraries

Two libraries have been developed in order to facilitate the control of vehicles.

The `driver` library provides the usual functionalities that are available to a human driver. The most important ones are directly related to the vehicle control, such as setting the steering angle, pressing the throttle or brake, setting the cruising speed or the gear. These functions come with their counterparts that return the actual steering angle, speed, gear, and state of the throttle and brake pedals. Other functions of the library include turning on/off the indicators and lights.

The `car` library provides abstraction of advanced functions that are typically not available to a human driver. Examples of these functions include setting the indicator blinking frequency or getting information such as wheels' encoder counts, angles of the steering wheels or the vehicle track (the distance between the centers of the front and rear wheels) and wheelbase

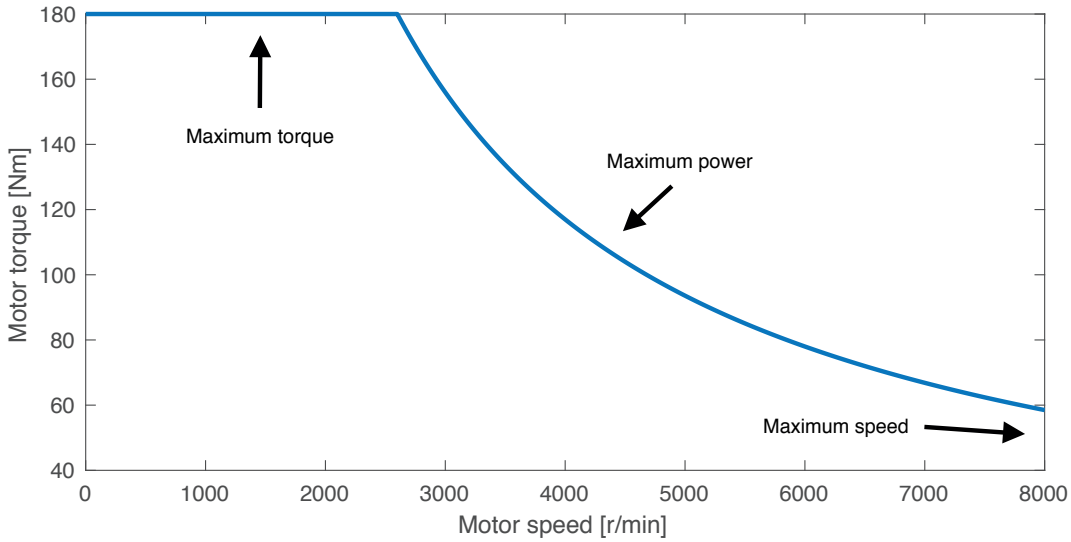


Figure 5.1 – The Citroën C-ZERO motor model, according to the owner's manual.

(the distance between the front and the rear axle).

5.1.3 Calibration

We calibrate our Webots model by running experiments on `car1`. We use the output from the Applanix localization system as the ground truth data. Other variables are directly obtained from the CAN bus of the C-ZERO.

The car control based on torque is implemented in the `driver` library as follows. First, the rotational speed of the motor is estimated from the rotational speed of the wheels, which can be measured by sensors available in Webots. A ratio between the motor speed and the car speed is obtained from Figure 5.2. Second, the output torque of the motor is computed using the rotational speed of the motor, based on Equation (5.1). The output torque is multiplied by the state of the throttle and the gearbox coefficient. We use a linear mapping of the throttle in the range $0-1$, but using a more complex non-linear model is possible. Finally, the torque is distributed (respecting the differential constraint) to the driven wheels (C-ZERO has rear-wheel drive), and the Webots physics simulation takes care of the rest.

In the first experiment, we accelerate from 0 to 100 $^{km}/_{h}$ at full throttle. We measure the motor torque as a function of motor speed directly from the car's CAN bus. We realize that we can adjust the maximum motor power in order for the model to better reflect the measured values. Both the data sheet and the adjusted model are shown in Figure 5.3, alongside the measured motor torque.

We put our adjusted model to test by comparing the behaviors of the real and simulated C-ZEROs. First, we compare the speed of the car while accelerating from 0 to 100 $^{km}/_{h}$ at full

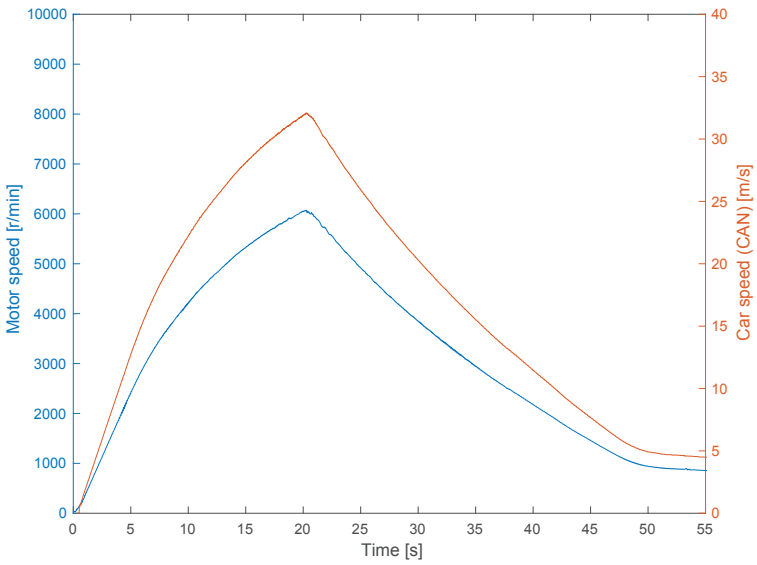


Figure 5.2 – Car speed (obtained from the car speedometer available on the CAN bus) vs. motor rotation speed.

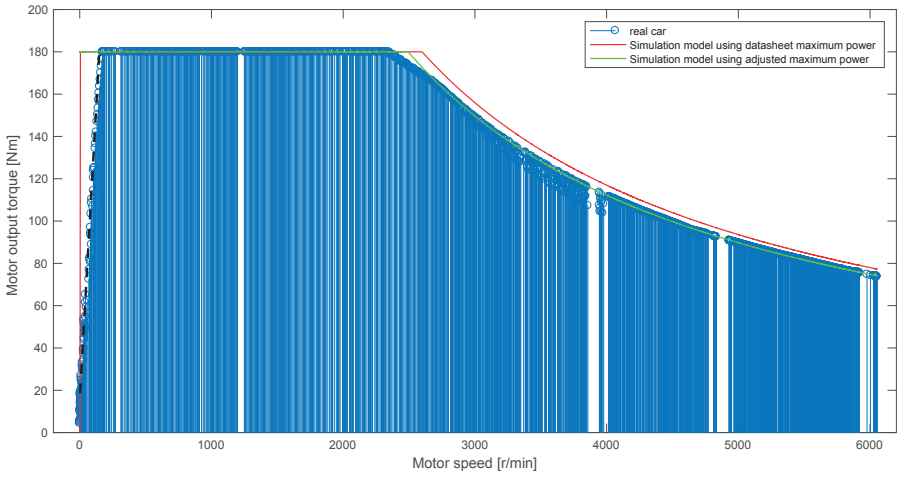
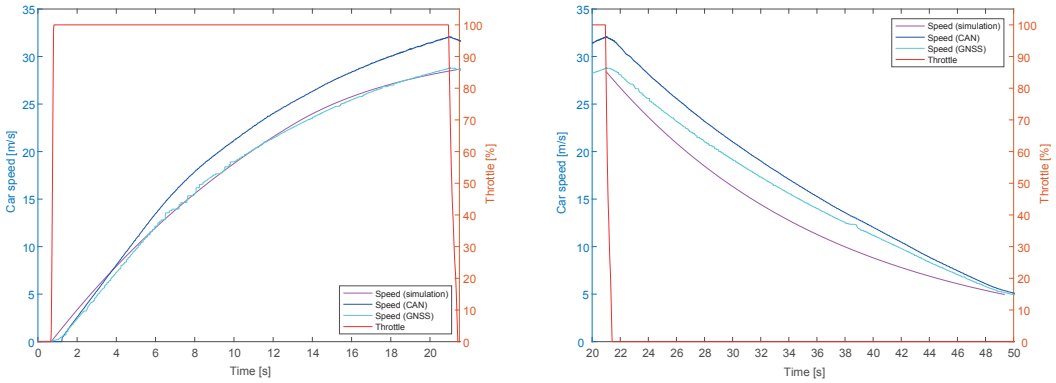


Figure 5.3 – The Citroën C-ZERO motor model (in green) calibrated using the motor speed and torque measurements (in blue) obtained from the C-ZERO while accelerating from 0 to 100 km/h with full throttle. The data sheet model from Figure 5.1 is shown in red.

acceleration obtained in simulation and from the real car. The result is shown in Figure 5.4a.

When the throttle is released, the car decelerates naturally due to different factors, such as friction, air drag and regenerative braking. These effects are jointly simulated by adding a damping factor to each of the wheel joints. A damping force F is proportional to the effective



(a) Acceleration **(b)** Deceleration

Figure 5.4 – Car speed obtained in Webots using our motor model compared with the data collected with the real C-ZERO. (a) The car accelerates from 0 to 100 km/h at full throttle. (b) The car decelerates without braking and with the throttle pedal released.

joint velocity:

$$F = -B\,v, \tag{5.2}$$

where B is the damping coefficient and amounts to 1.55 Nms for the C-ZERO. B is obtained by calibration from the real car. The results of this experiment are shown in Figure 5.4b, where one can observe that the simulated car's speed closely follows the measured speed of the real car when the throttle is released.

Brakes are implemented by applying damping on the rotational joints of each of the four wheels. Here the damping coefficient is normalized by the car speed, in order to achieve a constant friction independent from the speed, which is a concept similar to the friction applied by the brake pads on the brake disks on real vehicles. The position of the brake pedal is linearly mapped to the joint damping coefficient. The maximum braking torque amounts to 580 Nm and is obtained from the two real-car experiments, in which we apply maximum braking while driving at the speeds of 40 and 70 km/h, respectively. The resulting behavior is visualized in Figure 5.5.

5.2 Simulation of Automotive Sensors

Many sensors of interest for intelligent vehicles are common in mobile robotics and were therefore readily available in Webots. The list of these sensors includes the GNSS, inertial sensors (accelerometers and gyroscopes), compass and odometry sensors. These sensors were normally implemented by accessing the Webots physics engine and retrieving relevant properties of the body on which they are deployed (cars). They can therefore return perfect measurements. Alternatively, an arbitrary amount of uncorrelated Gaussian noise can be

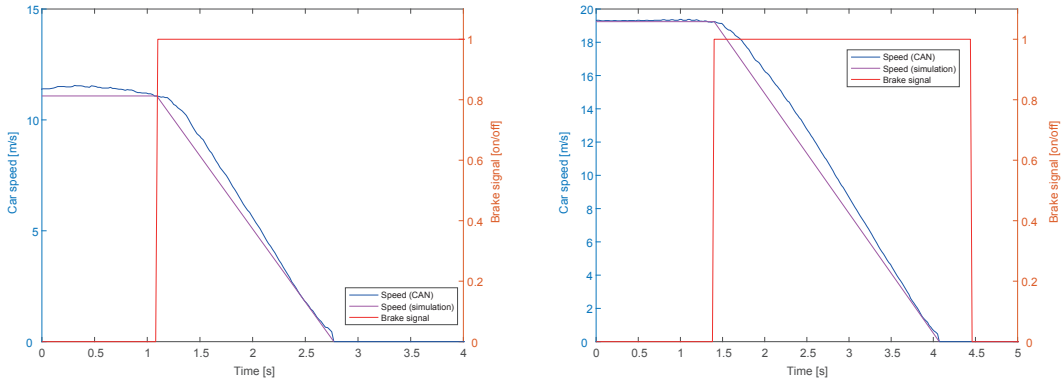


Figure 5.5 – Real car data and Webots model braking model: maximum braking from (a) 40 km/h and (b) 70 km/h.

added to the sensor outputs. An exception is the GNSS sensor, for which a correlated noise option also exists.

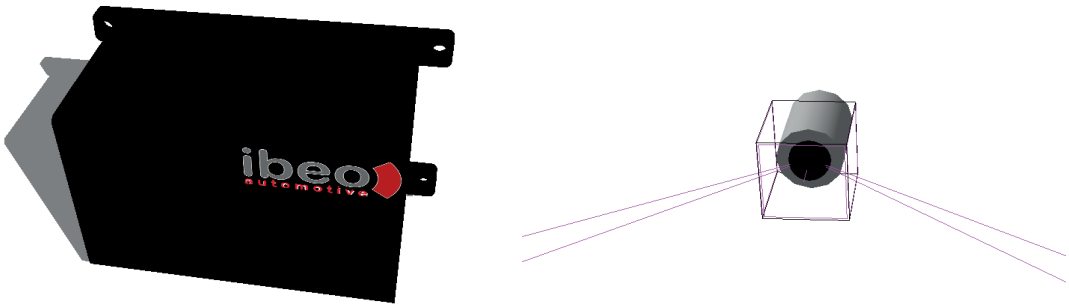
On the other hand, the implementation of LIDARs required significant improvements to meet the standards used in the automotive domain. Moreover, while an implementation of a general camera existed in Webots, we needed a specific one the Mobileye camera.

5.2.1 LIDARs

A general implementation of a single layer LIDAR sensor existed in Webots prior to our work, as LIDARs are commonly used in robotics. It was implemented using a 1-D depth camera. We implemented the Ibeo LUX model (see Figure 5.6a), which is characterized by four vertical layers forming the vertical FOV of 3.2°. Specifications from the sensor data sheet (FOV, resolution, noise) were carefully transfered to simulation.

The simulated Ibeo LIDAR is implemented using `OpenGL` 2D depth cameras and a spherical projection which is natively supported in Webots. The general principle is to set the FOV of a depth camera to match the FOV of the LIDAR, and then choose a row from the depth image that corresponds to a given LIDAR layer. While an `OpenGL` camera supports FOVs of up to 180°, using a wide FOV results in distortions at the boundaries of the FOV due to the planar projection. For that reason, for FOVs larger than 90° we use a spherical projection, which yields better depth accuracy. In total, three `OpenGL` cameras are used for simulation of an Ibeo LIDAR in Webots, one for the front face and one at each side, as illustrated in Figure 5.6b.

Finally, Gaussian noise is added to each LIDAR return. A small amount of random clutter is added to each scan as well.



(a) (b)

Figure 5.6 – (a) A simulated Ibeo LUX LIDAR in Webots. (b) A spherical camera with 110° horizontal FOV simulated using three cameras, one using the front, and the other two using the two side faces. A purple face of the cube represents an active camera (in total three cameras are active).

5.2.2 Mobileye Cameras

As mentioned in the previous chapter, the Mobileye camera system does not provide users with an image in real-time. Instead, image processing and object recognition is done on chip in a black-box fashion. Finally, the user only gets a list of detected objects such as cars, trucks, pedestrians, or cyclists.

Using the camera available in Webots and processing the image is a tedious job. One can either resort to libraries such as `OpenCV`, or can train a machine learning algorithm for object recognition (e.g., deep learning using a convolutional neural network). However, the model created in such way would have different characteristics than the real Mobileye camera.

Therefore, we have decided instead to emulate the Mobileye camera in Webots at the level of the supervisor layer. The supervisor layer has access to extra functions that are typically not available to a regular robot (such as the true state of any robot's component or any other object in the world). In our specific case, the supervisor can retrieve a list of objects of a certain category (e.g., cars or pedestrians), along with their real pose. We add noise to measurements generated in this way in order to make them follow the same patterns we could see by collecting data with the real camera.

Observations collected during the camera calibration (reported in Section 4.1.2) are used for emulating the Mobileye sensor in Webots. We noticed that the noise on the longitudinal axis is not normally distributed. Instead, for a stationary object, the camera returns a constant distance affected by a bias. The bias is nevertheless different for different runs, which makes its complete removal through calibration impossible. For our emulated camera we use a linear noise model dependent on the longitudinal distance $e = f(x_{\text{gt}})$, the slope of which changes for every simulated camera in every run: the maximal bias e_{max} is sampled at random from a truncated normal distribution with $\mu = 0$ and $\sigma = 1$, and the added noise is in the range $\pm e_{\text{max}}$

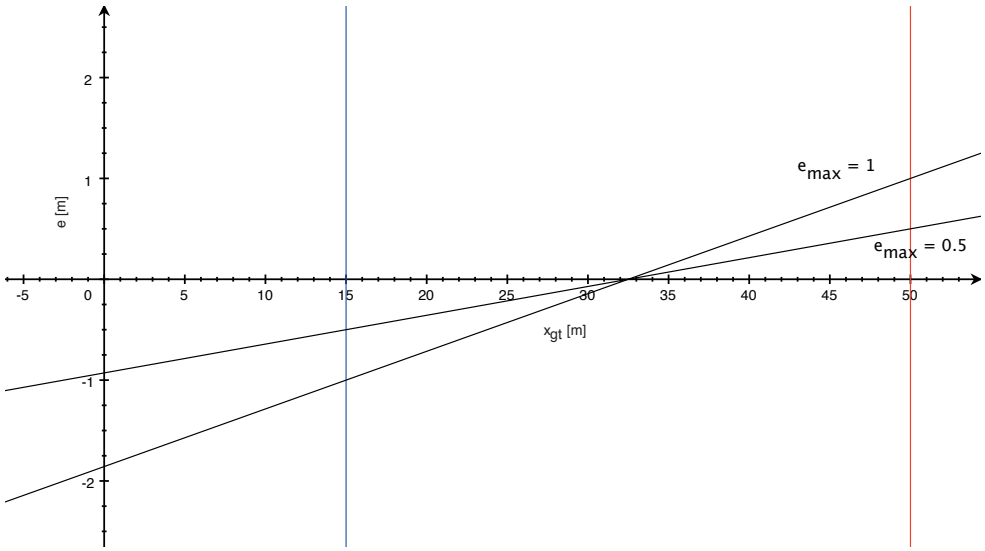


Figure 5.7 – Linear noise e added to the ground truth measurement x_{gt} when simulating the Mobileye camera. The maximal bias e_{max} at the min/max range of the camera (the vertical lines at $x_{\text{gt}} = 15$ and $x_{\text{gt}} = 50$ m, respectively) is sampled at random from a truncated normal distribution with $\mu = 0$ and $\sigma = 1$, and the slope of the linear model is computed accordingly. Two examples are drawn, for $e_{\text{max}} = 0.5$ and $e_{\text{max}} = 1$ m.

from the ground truth value. Thus, the simulated range of the Mobileye camera is obtained as $x = x_{\text{gt}} + e(x_{\text{gt}})$. Two examples are illustrated in Figure 5.7. A normally distributed noise is added to the y and θ camera returns, with $\sigma_y = 0.1$ m and $\sigma_\theta = 0.01$ rad.

Summary

We presented our efforts in developing a high-fidelity simulation tool for intelligent vehicles. We created a model of a Citroën C-ZERO vehicle in simulation and we calibrated its parameters by performing experiments using the real car. While many sensors of interest for automotive research were readily available in simulation, we had to implement or upgrade some of them. We implemented an Ibeo LUX LIDAR model and emulated a Mobileye camera, and we performed calibration of both new sensory devices.

6 Conclusion

REAL platforms are among the most exciting aspects of the work related to intelligent vehicles. There exist a multitude of sensors for perception and localization purposes. We presented our design choices and deployment of sensing, computation and communication equipment that can support our experimental campaign targeting cooperative perception. Our deployment spanned over two phases. Two Citroën C-ZERO electric cars represented the foundation of each of the phases. They were equipped by perception sensors such as Ibeo LUX LIDARs and Mobileye cameras, and localization sensors including an accurate Applanix localization system and a less-accurate but more affordable alternative based on a GPS, a speedometer, a compass and inertial sensors.

Realistic simulators are essential tools for prototyping and developing algorithms for robotics and intelligent vehicles. We presented our efforts towards developing a calibrated, high-fidelity simulator for intelligent vehicles. We developed a complete Citroën C-ZERO model, starting from the vehicle base and Ackerman steering, via the CAD model of the vehicle body and the electric motor model, to the libraries that facilitate interfacing the models. The model was calibrated using the data gathered with a real C-ZERO car. We further developed and calibrated models of Ibeo LUX LIDARs and Mobileye cameras implemented in the Webots simulator, which were necessary in order to have the same hardware in simulation and on the real cars.

6.1 Discussion

We developed our experimental platforms for the purpose of deploying and testing cooperative perception algorithms. From the definition of cooperative perception, it is clear that we need at least two vehicles equipped with some perception sensors and communication equipment. Furthermore, it is obvious that cooperation cannot occur unless the platforms can leverage an accurate solution for relative localization. While homogeneous platforms are not strictly required by our algorithms, from the practical perspective they are welcome. Having multiple exteroceptive modalities on a single car (i.e., a minimally rich set of sensors) was also a goal of our work, as this setup enables performing both intra-vehicle and inter-vehicle fusion. Our

experimental platforms were designed by keeping our research objectives in mind, taking into account the interests of the sponsor (PSA Groupe), and considering budget constraints at the same time.

The developments of appropriate simulation tools was aimed to facilitate our research on intelligent vehicles, otherwise only achievable by using complex experimental platforms. The flexible simulator allowed us to experiment with different settings while working on our algorithms. We intentionally invested effort in more sophisticated, high-fidelity simulation tools able to reduce the simulation-to-reality gap, as we knew, from our experience with mobile robots, that such a strategic choice would have paid off when we had to port our algorithms developed in simulation to real vehicles. Despite the fact that our goal was to faithfully reproduce our experimental platforms in simulation, the learned good practice principles are generalizable to other cars or sensory sets, also thanks to the flexibility of the chosen simulator. For instance, using our baseline Ackerman model, different car models can be easily reproduced. The many aspects of our work can be reused to implement other LIDAR or radar sensors.

From our experience, selecting the appropriate components for the real cars is not straightforward. It is difficult to know upfront the requirements for a given application in terms of accuracy, resolution, bandwidth, etc. However, designing and prototyping algorithms in a flexible simulation tool first represents a good way to determine the hardware requirements.

Cooperative Perception Part III

7 Introduction

One of the key components of an intelligent vehicle is its perception system. By having a precise representation of the environment, the intelligent vehicle can identify hazards, assess the level of danger, give warnings and recommendations to the driver or perform evasive actions if the driver is unable to do so in a timely fashion. Moreover, complete and accurate information about the environment is a prerequisite for the vehicle to drive by itself.

There often exist situations in which a vehicle does not have a complete view of the environment surrounding it. For instance, it can be due to occlusions, limited FOV, or sensor range limitations. In these situations, information coming from other vehicles or the road infrastructure can substantially improve the decision-making process of the intelligent vehicle.

Information sharing is usually achieved using wireless communication links. Communication can be considered as a type of virtual sensor, such as in [61], [62]. For example, it can enable vehicles to see behind corners, as the wireless communication does not have the same Line-Of-Sight (LOS) constraints as most conventional sensors have. Cooperative perception considers sharing of perception data (raw or processed) among two or more vehicles. If a cooperative vehicle uses simpler sensors (e.g., a speed sensor whose output is available on the vehicle's CAN bus), it can transmit raw sensory data. However, in case of richer sensors (e.g., cameras or LIDARs), due to data volume and communication channel constraints, sharing only processed data (e.g., objects) and filtered data (i.e., tracks) may be the only feasible option.

In this part of the thesis we address the problem of cooperative object tracking. Specifically, we focus on tracking of cars (and not vehicles in general, such as trucks or bicycles). Multiple cars are equipped with sensors and track other cars that are located in their respective FOVs. Cars have the ability to cooperate, i.e., share and fuse object estimates among each other.

The contributions of this part are as follows. In Chapter 8, we propose a complete method for vehicle tracking based on a Gaussian Mixture PHD (GM-PHD) filter and a car detection model that takes the car shape into account. Moreover, we present a general method for cooperative fusion of GM-PHD intensities, in which vehicles' FOVs do not need to overlap.

Our approach enables extending the FOV of a single vehicle beyond the limits of its sensors by using communication, and decreasing uncertainty of estimation where an object is observed by multiple vehicles. We refer to our approach as the Cooperative GM-PHD (C-GM-PHD) filter.

In Chapter 9, we move on to a Sequential Monte Carlo (SMC) implementation of the PHD filter. We propose an integrated approach of localization with multiple object tracking, factoring uncertainties from the localization itself into tracking. Furthermore, we also propose an improvement to the distributed fusion of SMC-PHD filters by generalizing it for non-overlapping FOVs. We refer to our generalized approach, from this point forth, as the Cooperative SMC-PHD (C-SMC-PHD) filter.

Finally, in Chapter 10 we show one possible application of cooperative perception. On top of the C-GM-PHD filter we build an overtaking recommendation system. A car attempting to overtake gets aided by a car being overtaken in assessing the maneuver risk, and subsequently makes a decision whether to overtake or not.

7.1 Background

7.1.1 Multi-Object Tracking

Multi-object tracking is the process of estimating the number of objects (also called targets) and their states from imperfect sensor measurements. The association of measurements with appropriate objects is not known. Measurements are subject to noise, missed detections and false alarms (clutter). The main challenge in multi-object tracking is to filter out these effects and solve the measurement-to-track association problem, in order to get accurate estimates of the true object states.

Multi-object tracking filters based on the Global Nearest Neighbor (GNN) algorithm [63] and Multiple Hypothesis Tracking (MHT) filters [64] involve explicit association between measurements and objects. Issues with such filters are related to having multiple observations of a single object, clutter and occluded objects. Joint Probabilistic Data Association (JPDA) filters [64] weight observations using their association probabilities, but require the number of objects to be known a priori. In our work the number of objects is unknown and varies with time.

Another solution for multi-object tracking is based on Random Finite Set (RFS) models, in which a set of objects of variable cardinality is modeled as an RFS. Filters based on this theory, such as PHD filters and Cardinalized Probability Hypothesis Density (CPHD) filters, deal with the measurement-to-track association implicitly [65]. They can provide higher robustness and accuracy in scenarios where the number of objects is variable and/or not known in advance. Many different implementations of the PHD filter have been developed and used for multi-object tracking; the most common ones are the GM-PHD [66] and the SMC-PHD filter [67]. Typical implementations assume that, if a target is detected, it generates only one

measurement. Moreover, in the previously mentioned literature sources [66] and [67], the shape of the target is not taken into account.

A Bayesian approach to the multi-object tracking problem requires a measurement model to describe how the measurements are related to underlying object states. A *standard* measurement model, commonly called a point target model, assumes that each object produces at most one measurement at a given time. This model holds well for some low-resolution sensors. On the other hand, higher resolution sensors (including our LIDAR sensor) may produce multiple measurements per object in a single scan. There exist two possibilities for dealing with such situations:

- the first option is to preprocess the sensor data with the goal of detecting individual objects and producing only one measurement per object;
- the second option is to use *extended* object measurement models, which typically model the number of measurements generated by each object along with their spatial distribution.

Granstrom et al. have extended the GM-PHD framework to handle extended targets [68], [69]. Estimation of the target shape was also included in the GM-PHD framework. For instance, a thin stick model was used for bicycle tracking in [70] and tracking of rectangular and elliptical extended targets was addressed in [71]. The drawbacks of these two approaches are the unconventional framework where the measurement model depends on the current set of measurements, as well as a high degree of non-linearity which makes the state update using an Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) difficult. In [72] Swain and Clark tracked the kinematic state of elliptical targets along with their shape parameters by considering hierarchical point process representations of multiple extended targets. The particle representation was used for the shape, while the Gaussian mixture formulation was used as a representation of a kinematic state per particle. In the DISAL laboratory, Gaussian processes in connection with the GM-PHD filter have been used to model and estimate an arbitrary object shape [73].

SMC methods represent an alternative to Gaussian Mixture (GM) methods in approximating the PHD filter as shown in [74], and are able to handle nonlinear models with relatively low complexity. They typically require a large number of particles for sampling the state space, and rely on clustering techniques to provide state estimates. Ristic et al. proposed an improved SMC approach in [67] that is able to reliably extract state estimates without the use of an explicit clustering algorithm. Clustering algorithms such as K-means or the ones based on the Expectation-Maximization (EM) principle require a Gaussian assumption or an a priori knowledge of the number of targets, and hence are counterproductive to the advantages of the nonlinear SMC-PHD filter.

In this thesis, we have chosen the first option mentioned above because of its simplicity. Our tracking is based on the PHD filter, and we use a detection algorithm prior to tracking to extract

features from the observed sensor data. These features are then used as measurements during the PHD update. Our detection algorithm, explained in Section 7.3, is similar to the approach presented in [63]. However, to the best of our knowledge, this approach has not been used in the RFS framework so far.

It is important to mention that the PHD filter does not provide labels for tracked objects. Both the GM-PHD filter and the SMC-PHD filter only provide identity-free estimates of object states and hence no spatial association of estimates over time. Conventionally, a track denotes a sequence of object states over time. However, it has been shown that labels for the Gaussian components can be included into the filter in order to obtain individual target tracks, see for example [75]. Moreover, data association and track management techniques for PHD filters are discussed in [76].

7.1.2 Cooperative Fusion

Two major problems related to cooperative fusion are temporal and spatial alignment [36]. Temporal alignment deals with variable delays of wireless communication links. Authors of [77] report that the typical delay of IEEE 802.11p wireless communication protocol is in the order of 100-150 ms. Consequently, variables received through communication may not represent faithfully the state of objects at the time they were received. Ideally, vehicle receiving observations should help to predict the evolution of the objects' state between the original measurement time and that of reception. This is obviously more difficult if raw measurements are communicated instead of tracks. Moreover, the spatial alignment considers differences in coordinate systems of the sending and receiving vehicles. The difficulty here lies in the uncertainty of sending and receiving vehicle locations.

Cooperative tracking is not a new problem. An approach for ball tracking by a team of robots (i.e., single object cooperative tracking), based on a modified particle filter, is presented by Ahmad and Lima [78]. In this approach, each robot weights its set of particles (representing the tracked object posterior) by its localization uncertainty, and transmits it to its teammates. In the fusion step, each particle is sampled at random from one of the cooperative robot sets, where the sampling probability corresponds to the particle weight. The generalization of this approach to multiple object tracking requires solving the association and clustering problems (as will be discussed in Chapter 9). Furthermore, Ahmad et al. presented an approach for simultaneous localization and multi-object tracking [79]. Here, poses of robots, moving objects, and static landmarks are represented as a graph, and the problem is phrased as a least-squares minimization problem. However, the experimental evaluation covered only one tracked object. A decentralized particle filter for multi-object tracking was presented by Ong et al. [80]. They used a Gaussian Mixture Model (GMM) to approximate a particle distribution with a continuous distribution. GMMs were communicated between the flying vehicles and fused using a Covariance Intersection (CI) algorithm.

When using a PHD filter for tracking, communicating tracks is reduced to communicating

PHD intensities. A method for fusing PHD intensity functions is given by Clark et al. in [81]. Battistelli et al. used Exponential Mixture Density (EMD) for distributed fusion of Gaussian Mixture CPHD (GM-CPHD) densities [82]. In both works, it has been assumed that all PHD filters work in the same domain, i.e., that all agents share a common FOV in which targets are sensed. This type of fusion is in literature also known as *consensus* fusion — because a consensus between all sensors about the states of given objects is sought. This is a very limiting assumption for the application of aforementioned methods in the field of (moving) vehicles, where cooperation between vehicles aims to increase the amount of information each vehicle has of its surroundings. If there is no unanimous agreement on some object between all sensors, a consensus-based fusion algorithm may fail to include all object state estimates in the fused posterior.

The distributed fusion of information from multiple instances of the Sequential Monte Carlo CPHD (SMC-CPHD) filter was explored in [83] by using EMDs. Similarly to the two approaches mentioned above, it assumes that all agents share a common FOV in which targets are detected. Furthermore, applications of the SMC-PHD filter and its distributed variation have been only experimentally verified with stationary sensors, mainly targeting defense applications.

In this thesis, our goal is to fuse PHD intensities originating from sensors whose FOVs only partially overlap. This type of fusion is also known as *complementary* fusion, as the algorithm should be able to opportunistically combine the complementary information provided by various sensors.

7.2 Preliminaries

In this section, we introduce precise mathematical formulations for the Bayes filter, the RFSs for multi-object tracking, and the PHD filter.

7.2.1 Bayes Filter

A Bayes filter is a general probabilistic approach for estimating an unknown Probability Density Function (PDF) recursively over time using incoming measurements and a mathematical process model.

The true state is assumed to be a Markov process on the state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$. The probability density of a transition to a state x_k at time k given state x_{k-1} at time $k-1$ is given by

$$f_{k|k-1}(\cdot|\cdot). \tag{7.1}$$

This Markov process is partially observed in the observation space $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$. Given a state x_k at time k, the probability density of receiving the observation $z_k \in \mathcal{Z}$ is modeled by the likelihood

function

$$g_k(z_k|x_k). \tag{7.2}$$

The probability density of the state x_k at time k given all observation $z_{1:k} = (z_1, \ldots, z_k)$ up to time k is called the posterior density at time k, and can be computed using the Bayes recursion consisting of the prediction and the update step, respectively:

$$p_{k|k-1}(x_k|z_{1:k-1}) = \int f_{k|k-1}(x_k|x)\, p_{k-1}(x|z_{1:k-1})\mathrm{d}x, \tag{7.3}$$

$$p_k(x_k|z_{1:k}) = \frac{g_k(z_k|x_k)\, p_{k|k-1}(x_k|z_{1:k-1})}{\int g_k(z_k|x)\, p_{k|k-1}(x|z_{1:k-1})\mathrm{d}x}. \tag{7.4}$$

Computing the integral in Equation (7.3) and the multiplication in Equation (7.4) in closed form is not feasible for most problems. There exist a number of techniques and algorithms that are derived from the Bayes filter, relying on different assumptions regarding the measurement and the state transition probabilities and the initial PDF. The most well-known implementations are a Kalman Filter (KF) and a SMC filter (commonly called also a Particle Filter (PF)) [84].

7.2.2 Random Finite Sets Formulation of Multi-Object Tracking

Now consider a multiple object scenario. Let $M(k-1)$ be the number of objects at time $k-1$, whose states are $x_{k-1,1}, \ldots, x_{k-1,M(k-1)} \in \mathcal{X}$. At time k, some of these objects may die; the surviving objects evolve to their new states; new objects may appear. This results in $M(k)$ new states $x_{k,1}, \ldots, x_{k,M(k)}$. At the same time, $N(k)$ measurements $z_{k,1}, \ldots, z_{k,N(k)}$ are received from the sensor. Only some of the $N(k)$ measurements originate from tracked objects, while the others represent clutter. There exists no information about which objects generated which measurements. Moreover, the order in which objects and measurements appear bears no significance. Thus, the collections of objects and measurements can be naturally represented as finite sets:

$$X_k = \{x_{k,1}, \ldots, x_{k,M(k)}\} \in \mathcal{F}(\mathcal{X}) \tag{7.5}$$

$$Z_k = \{z_{k,1}, \ldots, x_{k,N(k)}\} \in \mathcal{F}(\mathcal{Z}) \tag{7.6}$$

where $\mathcal{F}(\mathcal{X})$ and $\mathcal{F}(\mathcal{Z})$ are the respective collections of all finite subsets of $\mathcal{X}$ and $\mathcal{Z}$. The target set X_k and the measurement set Z_k represent the multi-object state and multi-object observation, respectively. The multi-object tracking problem can then be posed as a filtering problem with the (multi-object) state space $\mathcal{F}(\mathcal{X})$ and observation space $\mathcal{F}(\mathcal{Z})$.

In a single-object system, uncertainty is characterized by modeling the state x_k and the measurement z_k as random vectors. Analogously, uncertainty in a multi-object system is characterized by modeling the multi-object state X_k and the multi-object measurement Z_k as RFSs. An

RFS X is a finite-set-valued random variable, which can be described as a discrete probability distribution defining the cardinality of X, and a family of joint probability densities which characterizes the joint distribution of elements of X.

For a given state $x_{k-1} \in X_{k-1}$ at time $k-1$, its transition to the next time step is modeled as the RFS

$$S_{k|k-1}(x_{k-1}) \tag{7.7}$$

that can either take $\{x_k\}$ if the target survives with probability $p_{S,k}(x_{k-1})$, or $\emptyset$ when the object dies with probability $1 - p_{S,k}(x_{k-1})$. A new target at time k can arise independently from any existing object (by birth), or be spawned from an object at time $k-1$. So the multi-object state X_k at time k is given by the union of surviving objects, spawned objects and newly born objects:

$$X_k = \left[\bigcup_{\xi \in X_{k-1}} S_{k|k-1}(\xi) \right] \cup \left[\bigcup_{\xi \in X_{k-1}} B_{k|k-1}(\xi) \right] \cup \Gamma_k, \tag{7.8}$$

where Γ_k is a RFS of spontaneous birth at time k, and $B_{k|k-1}(\xi)$ is a RFS of targets spawned at time k from a target with previous state ξ.

The RFS measurement model accounts for detection uncertainty and clutter. A given target $x_k \in X_k$ is either detected with probability $p_{D,k}(x_k)$ or missed with probability $1 - p_{D,k}(x_k)$. Consequently, each state $x_k \in X_k$ generates an RFS at time k

$$\Theta_x(x_k), \tag{7.9}$$

which can take on either z_k when the target is detected, or $\emptyset$ when it is not. A multi-object measurement model Z_k is formed by the union of target generated measurements and clutter set K_k:

$$Z_k = K_k \cup \left[\bigcup_{x \in X_k} \Theta_k(x) \right]. \tag{7.10}$$

Let $p_k(\cdot|Z_{1:k})$ denote the multi-object posterior density. Then, the optimal multi-object Bayes filter propagates the multi-object posterior in time via recursion consisting of the predict step

$$p_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X) p_{k-1}(X|Z_{1:k-1}) \mu_s(\mathrm{d}X) \tag{7.11}$$

and the update step

$$p_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k) p_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X) p_{k|k-1}(X|Z_{1:k-1}) \mu_s(\mathrm{d}X)}. \tag{7.12}$$

Similarly to the single-object case, here $f_{k|k-1}(\cdot|\cdot)$ and $g_k(\cdot|\cdot)$ represent the multi-object

transition density and the multi-object likelihood, respectively. Explicit expressions for $f_{k|k-1}(X_k|X_{k-1})$ and $g_k(Z_k|X_k)$ can be derived using Finite Set Statistics (FISST) [65], but they are not essential for this thesis. μ_s is an appropriate reference measure on $\mathcal{F}(\mathcal{X})$.

The recursion in Equations (7.11)-(7.12) involve multiple integrals over multi-object space, which are computationally intractable.

7.2.3 Probability Hypothesis Density Filter

A PHD filter is proposed as an alleviation for the computational intractability problem of the multi-object Bayes filter. Similarly to the constant gain KF, which propagates the first moment (the mean) of the single-object state, the PHD filter propagates the first moment of the posterior multi-object state. This first order moment of a RFS X with probability distribution P is a non-negative function v on $\mathcal{X}$ called *intensity*, such that for each region $S \subseteq \mathcal{X}$ [66]

$$\int |X \cap S| P(\mathrm{d}X) = \int_S v(x)\mathrm{d}x. \tag{7.13}$$

The integral of v over any region S gives the expected number of elements of X that are in S. Hence, the total mass $\hat{N} = \int v(x)\mathrm{d}x$ gives the expected number of elements of X. The local maxima of the intensity v are points in $\mathcal{X}$ with the highest local concentration of expected number of elements, and hence can be used to generate estimates for the elements of X. The simplest approach is to round $\hat{N}$ and choose the resulting number of highest peaks from the intensity. The intensity is also known in the tracking literature as the PHD [66].

To present the PHD filter, we define the important variables:

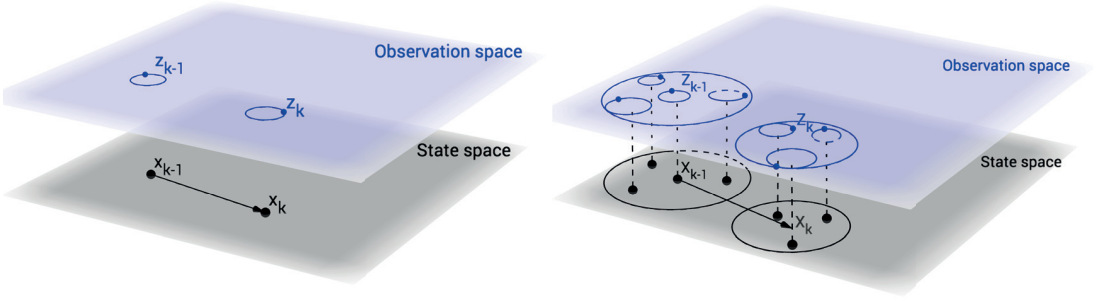
$\gamma_k(\cdot)$	=	intensity of the birth RFS Γ_k at time k,
$\beta_{k\|k-1}(\cdot\|\xi)$	=	intensity of the RFS $B_{k\|k-1}(\xi)$ spawned at time k by a target with previous state ξ,
$\kappa_k(\cdot)$	=	intensity of the clutter RFS K_k at time k.

Equations (7.11) and (7.12) can be approximated using the intensity measure, leading to the PHD recursion:

$$v_{k|k-1}(x) = \int p_{S,k}(\xi) f_{k|k-1}(x|\xi) v_{k-1}(\xi)\mathrm{d}\xi + \int \beta_{k|k-1}(x|\xi) v_{k-1}(\xi)\mathrm{d}\xi + \gamma_k(x), \tag{7.14}$$

$$v_k(k) = [1 - p_{D,k}(x)] v_{k|k-1}(x) + \sum_{z \in Z_k} \frac{p_{D,k}(x) g_k(z|x) v_{k|k-1}(x)}{\kappa_k(z) + \int p_{D,k}(\xi) g_k(z|\xi) v_{k|k-1}(\xi)\mathrm{d}\xi}. \tag{7.15}$$

Since the posterior intensity is a function on the single-object state space $\mathcal{X}$, the PHD recursion requires much less computational power than the multi-object recursion defined in Equations (7.11) and (7.12). However, the PHD recursion does not admit closed forms in general. Nevertheless, for a certain class of multi-object models — linear Gaussian multi-object models, the PHD recursion admits a closed-form solution, in a form similar to the KF (for the exact



(a) Single-object **(b)** Multi-object

Figure 7.1 – A simplistic comparison between a KF for single-object tracking and a GM-PHD filter for multi-object tracking. (a) The KF predicts the state of a single object and subsequently updates it using a single observation. (b) The GM-PHD filter predicts and updates a set of objects using a set of observations. It performs associations between measurements and objects, and to each reasonable association it assigns a KF to perform the prediction and the update steps.

form, see Equations (8.2)-(8.11)).

In a nutshell, we can think of a GM-PHD filter as a set of KFs, all communicating with a central authority deciding which KF to predict and update which target at every time step. The goal of individual KFs is to predict and update positions properly. The goal of the central authority is to maintain a persistent target-filter binding for each of the KFs, i.e., handle measurement-to-target association, give birth to new targets, and eliminate targets which are likely to no longer exist. A comparison between a single-object and a multi-object filter is shown in Figure 7.1.

7.3 LIDAR Point Cloud Preprocessing

Our LIDAR sensor typically produces multiple measurements corresponding to a single object, depending on the distance and angle from the sensor to that object. The goal of the preprocessing is to compress these point measurements to a single measurement per object, so the point target trackers could be used. Our preprocessing algorithm is inspired by the one presented by MacLachlan and Mertz in [63].

We start from a point cloud in which all points are given in the 3D Euclidean (x, y, z) coordinate system. The vertical (z) dimension is due to the fact that our LIDAR contains four scanning layers with a total vertical FOV of 3.2°. As our tracking algorithms operate in 2D, we project the point cloud to the ground plane. For each bearing angle of the LIDAR, we select the point (i.e., one of the four layers) that is closest to the sensor.

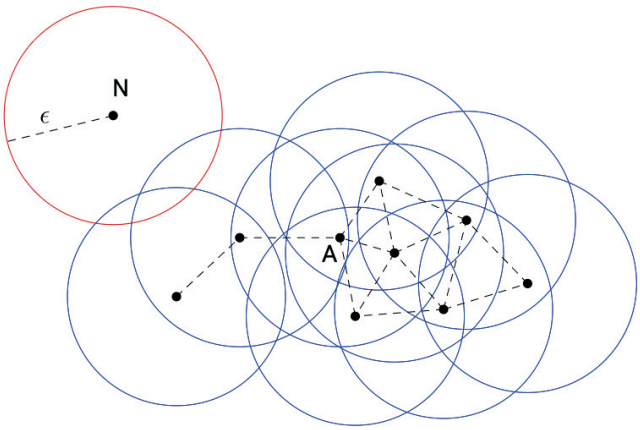


Figure 7.2 – An illustration of the DBSCAN algorithm for minPts = 4. The ϵ neighborhood of each point is drawn using a circle. All points with blue circle get added to the same cluster (e.g., A could be the point that is randomly selected in the beginning). Point N is noise, as it is neither within the ϵ neighborhood of any other point in the cluster, nor it is a core point (there are at least minPts in its ϵ neighborhood).

7.3.1 Clustering

Clustering divides the scanner data points into separate sets of points corresponding to distinct objects. We use a density-based spatial clustering algorithm called DBSCAN [85]. It has two parameters: a neighborhood size ϵ — defining how far a point has to be from any cluster to not be considered a part of that cluster — and minPts — the minimum number of points required to form a dense region (cluster).

The algorithm starts by selecting at random a point that has not yet been visited, and if at least minPts are contained in its ϵ-neighborhood, a new cluster is formed (see Figure 7.2). Otherwise this point is labeled as noise; this point might later be found in a sufficiently sized ϵ-neighborhood of another point and therefore be made part of a cluster. If a point is found to be part of a dense cluster, its ϵ-neighborhood is also made part of that cluster. Hence all the points found within the ϵ-neighborhood are added to the cluster, and so are the points belonging to their ϵ-neighborhood when they are also dense. The process continues until the complete density-connected cluster is found. Then, a new unvisited arbitrary point is selected and processed, leading to the discovery of a further cluster or noise.

Since our data points are located on a plane, the ϵ-neighborhood is simply defined using the Euclidean distance threshold. Due to the fixed horizontal angular resolution of our LIDAR and the nature of scanning, the distance between two adjacent points in the point cloud increases with their distance from the sensor. For this reason, our distance threshold d_{th} is dynamically computed by taking the fixed threshold e_{fix}, the distance between the point and the sensor

$r = \sqrt{x^2 + y^2}$ and the angular resolution of the sensor ϕ_{LiDAR} into account:

$$d_{th} = e_{fix} + 2r \tan \frac{\phi_{LiDAR}}{2}. \tag{7.16}$$

7.3.2 Feature Extraction

Once a cluster of points relating to one object has been created, it is necessary to analyze it and extract certain features related to that object. In this thesis, we are interested in detecting cars (Parts III and IV) and pedestrians (Part IV).

A **pedestrian** is characterized by a cluster whose size (length and width) is not larger than a parameter `minSizePed`. When a small cluster is detected, its center point is selected and returned as a measurement. The orientation of a pedestrian cannot be reliably detected using only a static LIDAR scan, and is therefore not extracted as a feature. For a given scan, a measurement set containing N_p detected pedestrians $Z^{(p)} = \{\mathbf{z}_i^{(p)}\}_{i=1}^{N_p}$, $\mathbf{z}_{(\cdot)}^{(p)} = [x_{(\cdot)}, y_{(\cdot)}]$ is returned by the algorithm. Uncertainty in the form of a covariance matrix is empirically evaluated. Note that many other small objects (such as a tree trunk) are detected as a pedestrian with this method.

For a **car** we assume a rectangular shape of a fixed size (L, W). Maximum two sides of a car can be detected at any time using a LIDAR. Therefore, when a cluster larger than `minSizePed` is detected, we fit one and two lines (i.e., a corner) to the cluster points and we compare the least squares error (LSE) of the two fits. If the LSE of the corner fit is N_{corner} times smaller than the LSE of the line fit, the corner is selected as a detected shape. Otherwise, a detected shape is a line. The corner point for the corner fitting is selected as a point with the largest Euclidean distance from the fitted line, as illustrated in Figure 7.3. Finally, the center point and the orientation of the car is computed from the best fit of the assumed rectangle to the detected line or corner. For a given scan, a measurement set containing the full 2D pose of N_c detected cars $Z^{(c)} = \{\mathbf{z}_i^{(c)}\}_{i=1}^{N_c}$, $\mathbf{z}_{(\cdot)}^{(c)} = [x_{(\cdot)}, y_{(\cdot)}, \theta_{(\cdot)}]$ is returned by the algorithm.

The measurement uncertainty of an individual LIDAR point can be expressed through the covariance matrix

$$R_{r,\alpha,\beta} = \begin{bmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\alpha & 0 \\ 0 & 0 & \sigma_\beta \end{bmatrix} \tag{7.17}$$

in the spherical coordinates, where r, α, and β represent the range, the azimuth and the polar angle, respectively. It is assumed that range, azimuth, and polar angle measurements are mutually independent. Alongside the processed measurement relating to a pedestrian or a car, the measurement noise (including the noise of the detection algorithm) in the observation space ($[x, y]^\top$ for a pedestrian, or $[x, y, \theta]^\top$ for a car) is also returned by the preprocessing algorithm. The noise is evaluated empirically in simulation using ground truth data for the targeted object. The process of car detection is summarized in Algorithm 7.1.

Algorithm 7.1: Car detection algorithm

Require: $\{[x^{(i)}, y^{(i)}, z^{(i)}]^\top, R^{(i)}_{r,\alpha,\beta}\}_{i=1}^{N_{pts}}$ {set of measurement points and their uncertainty}

clusters = DBSCAN($[x, y]^\top$)
for each cluster in clusters **do**
    Fit a line
    Fit a corner (corner point is point with largest distance to the line connecting first and last measurement point)
    Choose the best fit (line or corner) using the RMS error
    Fit a rectangle (with constant L, W) to a chosen line/corner
    Extract rectangle features x, y, θ
    Construct measurement covariance $R_{x,y,\theta}$
end for
return $\{[x^{(j)}, y^{(j)}, \theta^{(j)}]^\top, R^{(j)}_{x,y,\theta}\}_{j=1}^{N_{obj}}$ {set of object observations}

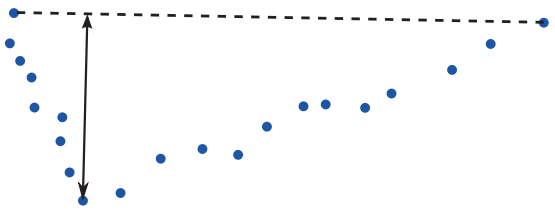


Figure 7.3 – The corner point is the point that is the furthest away from the dashed line connecting the first and last point LIDAR in this cluster (sorted by the bearing angle). LIDAR points are shown in blue.

Summary

This section provided an introduction to cooperative perception problem by braking it down into multi-object tracking and cooperative fusion problems. We gave an overview of the literature covering these two problems. Furthermore, we introduced fundamental concepts of multi-object tracking such as the Bayes filter, the Random Finite Set formulation, and the Probability Hypothesis Density filter. Finally, we described our LIDAR preprocessing algorithm for object detection prior to tracking.

8 Cooperative Multi-Object Tracking Using a Gaussian Mixture Approximation

To tackle the problem of cooperative multi-object tracking using the GM approximation, we propose a C-GM-PHD filter. It extends the GM-PHD filter from [66] with the fusion block, that has the ability to fuse PHD intensities coming from other GM-PHD or C-GM-PHD filters with its own PHD intensity, therefore increasing its estimation confidence and extending its FOV. The concept is illustrated in Figure 8.1.

8.1 Multi-Object Tracking Using a GM-PHD Filter

Assume that the posterior intensity at time $k-1$ is a Gaussian mixture of J_{k-1} components with weights w_{k-1}, means m_{k-1} and covariances P_{k-1} of the form

$$v_{k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k-1}^{(i)}, P_{k-1}^{(i)}) \tag{8.1}$$

In order to compute the posterior $v_k(\mathbf{x})$, the C-GM-PHD filter contains five fundamental steps, as described below. While the prediction, update, pruning and merging, and target extraction are taken from [66], the fusion step is our contribution towards the C-GM-PHD filter.

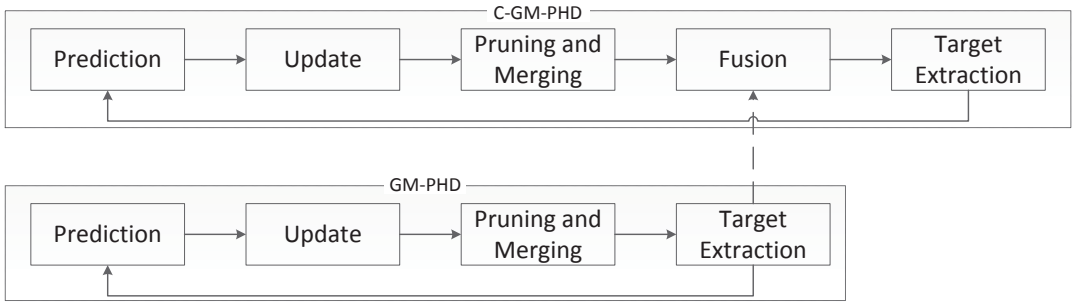


Figure 8.1 – Block diagram of C-GM-PHD steps. It fuses the intensity received from a GM-PHD (or C-GM-PHD) filter, if available.

8.1.1 Prediction

The predicted intensity for time k is also a Gaussian mixture

$$v_{k|k-1}(\mathbf{x}) = \sum_{j=1}^{J_{k-1}} p_{S,k}(m_{k|k-1}^{(j)}) w_{k-1}^{(j)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}) + \gamma_k(\mathbf{x}) \tag{8.2}$$

where $\gamma_k(\mathbf{x})$ is the birth intensity with $J_{\gamma,k}$ components given by

$$\gamma_k(\mathbf{x}) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(\mathbf{x}; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}) \tag{8.3}$$

and $p_{S,k}(m_{k|k-1}^{(j)})$ is the probability of survival of Gaussian component j at time k. Note that, for simplicity, we do not include spawning of targets in our filter implementation.

The predicted mean and covariance of estimated GM components can be computed using any Bayesian estimator. In our work we use an UKF:

$$m_{k|k-1} = \sum_{i=0}^{2\Lambda} \Upsilon^{(i)} \chi_{k|k-1}^{(i)} \tag{8.4}$$

$$P_{k|k-1} = \sum_{i=0}^{2\Lambda} \Upsilon^{(i)} [\chi_{k|k-1}^{(i)} - m_{k|k-1}][\chi_{k|k-1}^{(i)} - m_{k|k-1}]^\top, \tag{8.5}$$

where $\chi_{k|k-1}^{(i)}$ is a sigma point computed by the Unscented Transform and $\Upsilon^{(i)}$ its associated weight. The number of state dimensions is denoted by Λ, and the total number of sigma points used by the UKF is $2\Lambda + 1$.

8.1.2 Update

For a set of measurements Z_k, the update step yields a posterior intensity given by:

$$\begin{aligned} v_k(\mathbf{x}) = &\sum_{i=1}^{J_{k-1}} [1 - p_{D,k}(m_{k|k-1}^{(i)})] w_{k|k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) \\ &+ \sum_{\mathbf{z} \in Z_k} \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(\mathbf{z}) \mathcal{N}(\mathbf{x}; m_{k|k}^{(j)}(z), P_{k|k}^{(j)}) \end{aligned} \tag{8.6}$$

where

$$w_k^{(j)}(\mathbf{z}) = \frac{p_{D,k}(m_{k|k-1}^{(j)})\, w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z})}{\kappa_k(\mathbf{z}) + \sum_{l=1}^{J_{k|k-1}} p_{D,k}(m_{k|k-1}^{(l)})\, w_{k|k-1}^{(l)} q_k^{(l)}(\mathbf{z})}, \tag{8.7}$$

$$q_k^{(j)}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; H_k^{(j)} m_{k|k-1}^{(j)}, R_k + H_k^{(j)} P_{k|k-1}^{(j)} [H_k^{(j)}]^\top), \tag{8.8}$$

$$m_{k|k}^{(j)}(\mathbf{z}) = m_{k|k-1}^{(j)} + K_k^{(j)}(\mathbf{z} - H_k m_{k|k-1}^{(j)}), \tag{8.9}$$

$$P_{k|k}^{(j)} = [I - K_k^{(j)} H_k] P_{k|k-1}^{(j)}, \tag{8.10}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} H_k^\top (H_k P_{k|k-1}^{(j)} H_k^\top + R_k)^{-1}. \tag{8.11}$$

The parameters used in the update step are the clutter level $\kappa_k(\mathbf{z})$, the probability of detection $p_{D,k}(m_{k|k-1}^{(i)})$ dependent on the mean of the Gaussian component i, the observation model H_k and the observation noise covariance R_k.

8.1.3 Pruning and Merging

After the update step, the number of Gaussian components increases quadratically with the number of measurements. It is therefore necessary to limit the number of Gaussians, as to keep the problem tractable. A good approximation of the GM posterior intensity can be obtained by truncating components with weak weights. In other words, only a set containing Gaussian components $I = \{i = 1, \ldots, J_k | w_k^{(i)} > T_p\}$ with weights higher than a certain pruning threshold T_p are kept.

Furthermore, all Gaussian components that are close to each other can be approximated by a single Gaussian. This is done by selecting a Gaussian component with a highest weight, and finding all Gaussians whose mean is within the Mahalanobis distance U from the selected Gaussian. This yields a set $L = \{i \in I | (m_k^{(i)} - m_k^{(j)})^\top (P_k^{(i)})^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U\}$ of Gaussian components that are merged as follows:

$$\widetilde{w}_k^{(l)} = \sum_{i \in L} w_k^{(i)}, \tag{8.12}$$

$$\widetilde{m}_k^{(l)} = \frac{1}{\widetilde{w}_k^{(l)}} \sum_{i \in L} w_k^{(i)} x_k^{(i)}, \tag{8.13}$$

$$\widetilde{P}_k^{(l)} = \frac{1}{\widetilde{w}_k^{(l)}} \sum_{i \in L} w_k^{(i)} (P_k^{(i)} + (\widetilde{m}_k^{(l)} - m_k^{(i)})(\widetilde{m}_k^{(l)} - m_k^{(i)})^\top). \tag{8.14}$$

8.1.4 Extraction

The extraction of multiple-target state estimates is straightforward from the GM representation of the RFS intensity. According to [66], means of Gaussians that have weights greater than some threshold T_e (e.g., $T_e > 0.5$) are extracted and represent the most likely estimates.

8.2 Cooperative Fusion

After information (in the form of the GM-PHD intensity) has been received from the cooperative vehicle $\mathcal{C}$ over the communication link, it needs to be fused with the local intensity of the ego vehicle $\mathcal{E}$.

The first step is to transform the intensity from the cooperative vehicle's coordinate frame to the ego vehicle's coordinate frame. However, the transformation between the two frames is not exactly known, due to the pose uncertainty of the two vehicles. Hence, we will refer to it as an *approximate transformation*. Approximate transformations [86] allow for calculating the nominal location and associated error of any object relative to any other object. In Figure 8.2 we first describe the coordinates $\mathbf{x}^W_{\mathcal{NC}}$ of a non-cooperative vehicle $\mathcal{NC}$ (target) with respect to the world reference frame W, by carrying out transformations from W to the cooperative vehicle frame $\mathcal{C}$ ($\mathbf{x}^W_{\mathcal{C}}$) and from the frame of $\mathcal{C}$ to the target ($\mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}$). If the explicit transformation is given by a vector function

$$\mathbf{x}^W_{\mathcal{NC}} = f(\mathbf{x}^W_{\mathcal{C}}, \mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}) \tag{8.15}$$

and the vector function is approximated by a first-order Taylor series expansion about the means of the variables, the mean value of the approximate transformation is obtained by applying the function to variable means, i.e., $\hat{\mathbf{x}}^W_{\mathcal{NC}} = f(\hat{\mathbf{x}}^W_{\mathcal{C}}, \hat{\mathbf{x}}^{\mathcal{C}}_{\mathcal{NC}})$. The covariance matrix of this transformation is

$$P_3 = J \begin{pmatrix} P1 & 0 \\ 0 & P2 \end{pmatrix} J^\top = HP_1H^\top + KP_2K^\top, \tag{8.16}$$

where P_1 and P_2 are respectively covariances of transformations $\mathbf{x}^W_{\mathcal{C}}$ and $\mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}$, and J is the Jacobian of the transformation

$$J = \left(\frac{\partial f}{\partial \mathbf{x}^W_{\mathcal{C}}} \middle| \frac{\partial f}{\partial \mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}} \right) = [H|K]. \tag{8.17}$$

Transformation $\mathbf{x}^{\mathcal{E}}_W$ (see Figure 8.2) is obtained by reversing the transformation from W to $\mathcal{E}$ ($\mathbf{x}^W_{\mathcal{E}}$), and the covariance matrix for the reversed transformation (e.g., P') is estimated from the given covariance matrix of the transformation $\mathbf{x}^W_{\mathcal{E}}$ (e.g., matrix P) as

$$P' = R \cdot P \cdot R^\top, \tag{8.18}$$

where R is the Jacobian of the reversed transformation equations. Finally, the transformation from $\mathcal{E}$ to the target ($\mathbf{x}^{\mathcal{E}}_{\mathcal{NC}}$) and its covariance matrix are obtained using the transformations $\mathbf{x}^{\mathcal{E}}_W$ and $\mathbf{x}^W_{\mathcal{NC}}$, similarly as in the case of the transformation $\mathbf{x}^W_{\mathcal{NC}}$ being obtained from the transformations $\mathbf{x}^W_{\mathcal{C}}$ and $\mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}$.

We use the above method to perform coordinate transformations. Once the means and covari-

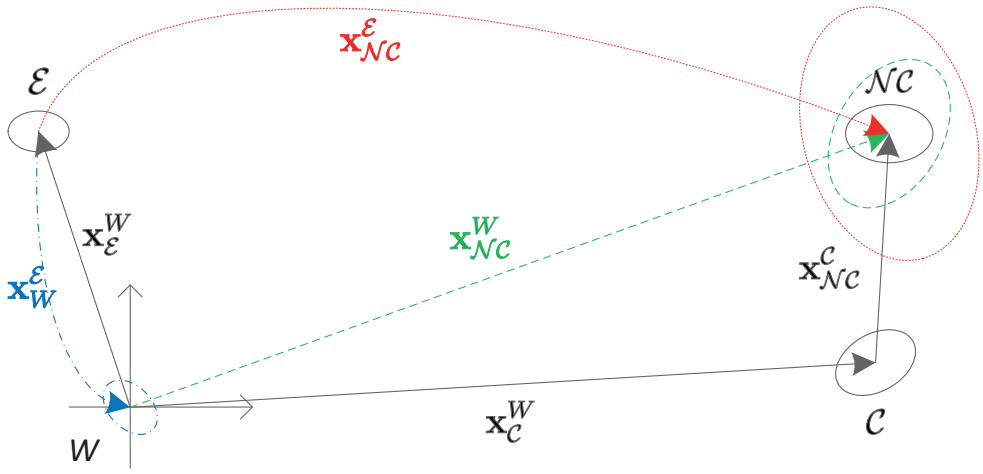


Figure 8.2 – A sequence of approximate transformations. The transformation from the world frame W to the cooperative vehicle frame $\mathcal{C}$ is denoted by $\mathbf{x}^{W}_{\mathcal{C}}$. Cooperative vehicle tracks a target $\mathcal{NC}$ in its own frame; the target's position is defined by the transformation $\mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}$. The location of the tracked target can be expressed in the world frame using the transformation $\mathbf{x}^{W}_{\mathcal{NC}} = \mathbf{x}^{W}_{\mathcal{C}} + \mathbf{x}^{\mathcal{C}}_{\mathcal{NC}}$. The transformation $\mathbf{x}^{\mathcal{E}}_{\mathcal{NC}}$ can be obtained as $\mathbf{x}^{\mathcal{E}}_{W} + \mathbf{x}^{W}_{\mathcal{NC}}$, where $\mathbf{x}^{\mathcal{E}}_{W} = -\mathbf{x}^{W}_{\mathcal{E}}$. Reference frames other than W are omitted for clarity.

ances of all Gaussians are expressed in the local reference frame of the ego vehicle, they can be fused with the local intensity produced by the C-GM-PHD filter.

Optimal (Bayesian) fusion techniques require statistically independent information, but this assumption does not hold if multiple sources are exchanging information between each other. If the common information across the sources is not known and cannot be canceled out, optimal fusion would lead to over-confidence [82]. CI [87] is a conservative method for fusing two Gaussian distributions, whose cross-correlation is unknown. CI is known to estimate the fused uncertainty that will neither be too small (over-confidence) nor too large (under-confidence). In [88], Mahler proposed a generalization of the CI algorithm for multi-object functions, called Generalized Covariance Intersection (GCI). It is shown in [81] that the location density of the two GM-PHD posteriors fused by the means of GCI becomes:

$$v_{\mathcal{W}}(\mathbf{x}) = \frac{v_1^{\mathcal{W}}(\mathbf{x}) v_2^{1-\mathcal{W}}(\mathbf{x})}{\int v_1^{\mathcal{W}}(\mathbf{y}) v_2^{1-\mathcal{W}}(\mathbf{y}) \mathrm{d}\mathbf{y}}, \tag{8.19}$$

where v_1 and v_2 are the two individual GM-PHD posterior intensities of the form

$$v(\mathbf{x}) = \sum_{j=1}^{J_k} w_j \mathcal{N}(\mathbf{x}; m_j, P_j), \tag{8.20}$$

and $0 \leq \mathcal{W} \leq 1$ is a fusion parameter.

However, the exponential of a GM does not have a GM form. Given that the posterior intensities

are GMMs, the following approximation of the Exponential Gaussian Mixture is proposed in [82] in order to preserve the GM form of the location PDF after fusion:

$$\left[\sum_{j=1}^{J_k} w_j \mathcal{N}(\mathbf{x}; m_j, P_j)\right]^{\mathcal{W}} \cong \sum_{j=1}^{J_k} [w_j \mathcal{N}(\mathbf{x}; m_j, P_j)]^{\mathcal{W}}. \tag{8.21}$$

To solve Equation (8.21), we list a few useful identities below. The power of the Gaussian component is a Gaussian component:

$$[w_j \mathcal{N}(\mathbf{x}; m_j, P_j)]^{\mathcal{W}} = w_j^{\mathcal{W}} \kappa(\mathcal{W}, P_j) \mathcal{N}(\mathbf{x}; m_j, \frac{P_j}{\mathcal{W}}), \tag{8.22}$$

where

$$\kappa(\mathcal{W}, P) = \frac{[\det(2\pi P \mathcal{W}^{-1})]^{\frac{1}{2}}}{[\det(2\pi P)]^{\frac{\mathcal{W}}{2}}}. \tag{8.23}$$

The product of two Gaussian components is a Gaussian component

$$w_1 \mathcal{N}(\mathbf{x}; m_1, P_1) \cdot w_2 \mathcal{N}(\mathbf{x}; m_2, P_2) = w_{12} \mathcal{N}(\mathbf{x}, m_{12}, P_{12}), \tag{8.24}$$

where

$$P_{12} = \left(P_1^{-1} + P_2^{-1}\right)^{-1}, \tag{8.25}$$
$$m_{12} = P_{12}(P_1^{-1} m_1 + P_2^{-1} m_2), \tag{8.26}$$
$$w_{12} = w_1 w_2 \cdot \mathcal{N}(m_1 - m_2; 0, P_1 + P_2). \tag{8.27}$$

Multiplying two exponents of GMs (as suggested in Equation (8.19)) yields

$$v_1^{\mathcal{W}}(\mathbf{x}) \cdot v_2^{1-\mathcal{W}}(\mathbf{x}) = \sum_{i=1}^{J_1} [w_i^{(1)} \mathcal{N}(\mathbf{x}; m_i^{(1)}, P_i^{(1)})]^{\mathcal{W}} \cdot \sum_{j=1}^{J_2} [w_j^{(2)} \mathcal{N}(\mathbf{x}; m_j^{(2)}, P_j^{(2)})]^{1-\mathcal{W}} \tag{8.28}$$
$$= \sum_{i=1}^{J_1} \sum_{j=1}^{J_2} w_{ij}^{(12)} \mathcal{N}(\mathbf{x}; m_{ij}^{(12)}, P_{ij}^{(12)}), \tag{8.29}$$

where

$$P_{ij}^{(12)} = \left[\mathcal{W}(P_i^{(1)})^{-1} + (1-\mathcal{W})(P_j^{(2)})^{-1}\right]^{-1}, \tag{8.30}$$
$$m_{ij}^{(12)} = P[\mathcal{W}(P_i^{(1)})^{-1} m_j^{(2)} + (1-\mathcal{W})(P_i^{(1)})^{-1} m_j^{(2)}], \tag{8.31}$$
$$w_{ij}^{(12)} = \left(w_i^{(1)}\right)^{\mathcal{W}} \left(w_j^{(2)}\right)^{1-\mathcal{W}} \kappa(\mathcal{W}, P_i^{(1)}) \kappa(1-\mathcal{W}, P_j^{(2)}) \mathcal{N}\left(m_i^{(1)} - m_j^{(2)}; 0, \frac{P_i^{(1)}}{\mathcal{W}} + \frac{P_j^{(2)}}{1-\mathcal{W}}\right). \tag{8.32}$$

As demonstrated above, by using the approximation from Equation (8.21), GCI fusion reduces to applying CI to each pair of Gaussian components from ν_1 and ν_2 in turn, resulting in the total number of $J_k(\nu_1) \times J_k(\nu_2)$ components in the fused intensity.

The aforementioned approach was developed for fusion of sensors whose FOVs overlap completely, and it does not work well in the general case, when the domains over which GMs are defined are not the same (i.e., when the FOVs of different sensors do not entirely overlap). It is evident that GCI assigns significant weight only to components located in the common FOV of multiple sensors.

Our fusion algorithm is given in Algorithm 8.1. We apply CI only to Gaussian components that are close to each other (lines 5-17), while keeping the rest of Gaussian components intact (lines 19-23). Closeness of Gaussian components is expressed through Mahalanobis distance with the threshold U_F (line 7). The cardinality of the fused set is computed as the weighted cardinality of the two initial sets (line 18). After fusing, we merge the components that are close to each other, similar to Section 8.1.3.

This approach enables us to have the same fusion experience in the common FOV of the two sensors as when GCI is used, while at the same time keeping track of all the targets that are in the FOV of only one sensor. We note that the accuracy of vehicles' pose plays an important role in the sensor fusion step.

8.3 Cooperative Car Tracking

Section 8.1 shows a general method that can be applied to tracking of any kind of object using any sensor modality. In this section, we provide additional details and models that we use to perform tracking of cars using LIDARs.

8.3.1 Kinematic State and Motion Model

Since in this chapter we focus on car tracking, we can afford to use a rectangular model for representing targets, whose state space is a combination of kinematic and shape parametric variables:

$$\mathbf{x} = \begin{bmatrix} x & y & v & \theta & \omega & L & W \end{bmatrix}^\top, \tag{8.33}$$

where x and y are the Euclidean coordinates of the center of the rectangle, v is the speed, θ denotes the heading angle and ω is the turn rate. Length and width of the rectangle are denoted by L and W, respectively. For state propagation we have selected a Constant Turn Rate and Velocity (CTRV) motion model, in literature also known as coordinated turn with polar velocity (see for example [89]–[91]), which we have augmented with object dimensions,

Algorithm 8.1: Fusion algorithm

Require: $\{m_1^{(i)}, P_1^{(i)}, w_1^{(i)}\}_{i=1}^{N_1}, \{m_2^{(j)}, P_2^{(j)}, w_2^{(j)}\}_{j=1}^{N_2}, U_F, \mathcal{W}$

$I_F = J_F = m = \emptyset$

$\alpha_1^{(i)} = w_1^{(i)} / \sum_k w_1^{(k)}, i = \{1, \ldots, N_1\}$

$\alpha_2^{(j)} = w_2^{(j)} / \sum_l w_2^{(l)}, j = \{1, \ldots, N_2\}$

$N_{12} = 0$

for $i = 1$ to N_1 **do**

  for $j = 1$ to N_2 **do**

    if $(m_1^{(i)} - m_2^{(j)})^\top (0.5(P_1^{(i)} + P_2^{(j)}))^{-1} (m_1^{(i)} - m_2^{(j)}) \leq U_F\}$ **then**

      $I_F = I_F \cup i$ {save indices of fused components}

      $J_F = J_F \cup j$

      $\alpha = (\alpha_1^{(i)})^{\mathcal{W}} (\alpha_2^{(j)})^{1-\mathcal{W}} \kappa(\mathcal{W}, P_1^{(i)}) \kappa(1-\mathcal{W}, P_2^{(j)})$
      $\cdot \mathcal{N}(m_1^{(i)} - m_2^{(j)}, 0, P_1^{(i)}/\mathcal{W} + P_2^{(j)}/(1-\mathcal{W}))$
      where $\kappa(\mathcal{W}, P) = [\det(2\pi P \mathcal{W}^{-1})]^{\frac{1}{2}} / [\det(2\pi P)]^{\frac{\mathcal{W}}{2}}$

      $P = [\mathcal{W}(P_1^{(i)})^{-1} + (1-\mathcal{W})(P_2^{(j)})^{-1}]^{-1}$

      $m = P[\mathcal{W}(P_1^{(i)})^{-1} m_1^{(i)} + (1-\mathcal{W})(P_2^{(j)})^{-1} m_2^{(j)}]$

      $\alpha_{12} = \alpha_{12} \cup \alpha, P_{12} = P_{12} \cup P, m_{12} = m_{12} \cup m$

      $N_{12} = N_{12} + 1$

    end if

  end for

end for

$w_{12}^{(n)} = \frac{\alpha_{12}^{(n)}}{\sum_k \alpha_{12}^{(k)}} (\sum_{i=1}^{N_1} w_1^{(i)})^{\mathcal{W}} (\sum_{j=1}^{N_2} w_2^{(j)})^{1-\mathcal{W}}$, for $n = 1, \ldots, N_{12}$ {Scale weights using weighted cardinalities of the two sets}

for $i \in I \setminus I_F \wedge j \in J \setminus J_F$ **do**

  $m_{12} = m_{12} \cup m_1^{(i)} \cup m_2^{(j)}$

  $P_{12} = P_{12} \cup P_1^{(i)} \cup P_2^{(j)}$

  $w_{12} = w_{12} \cup w_1^{(i)} \cup w_2^{(j)}$

end for

return $\{m_{12}^{(n)}, P_{12}^{(n)}, w_{12}^{(n)}\}_{n=1}^{N_{12}}$

assumed to be constant. The dynamic state equation is given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + G(\mathbf{x}_k)\xi_k, \tag{8.34}$$

where

$$f(\mathbf{x}) = \begin{bmatrix} x + \left(\frac{2v}{\omega}\right)\sin\left(\frac{\omega t}{2}\right)\cos\left(\theta + \frac{\omega t}{2}\right) \\ y + \left(\frac{2v}{\omega}\right)\sin\left(\frac{\omega t}{2}\right)\sin\left(\theta + \frac{\omega t}{2}\right) \\ v \\ \theta + \omega t \\ \omega \\ L \\ W \end{bmatrix} \tag{8.35}$$

$$G = \begin{bmatrix} \frac{t^2}{2}\cos(\theta) & \frac{t^2}{2}\sin(\theta) & t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{t^2}{2} & t & 0 & 0 \end{bmatrix}^\top \tag{8.36}$$

$$\xi = \begin{bmatrix} a & \alpha \end{bmatrix}^\top = \begin{bmatrix} \frac{dv}{dt} & \frac{d\omega}{dt} \end{bmatrix}^\top. \tag{8.37}$$

Input noise on linear and rotation acceleration a and α are threated as uncorrelated random variables with zero mean and standard deviations σ_a m/s^2 and σ_α rad/s^2, and t represents time.

8.3.2 Measurement Model

At each time step, a LIDAR returns a set of measurement points (a point cloud). Object detection is accomplished in several steps explained in Section 7.3. A processed measurement contains the center and orientation of one rectangle.

Measurements of rectangle center are directly used to update x and y variables of the state vector $\mathbf{x}$. The rectangle orientation has a π rad ambiguity (from a static LIDAR scan one cannot determine object heading but only its orientation). Therefore, before updating the heading $\theta_\mathbf{x}$ in the state vector, we adapt the orientation measurement $\theta_\mathbf{z}$ such that the difference between the orientation measurement and the state heading is wrapped to the interval $[-\frac{\pi}{2}, \frac{\pi}{2})$:

$$\begin{aligned} \delta &= \mod(\theta_\mathbf{z} - \theta_\mathbf{x} + \frac{\pi}{2}, \pi) - \frac{\pi}{2} \\ \theta_\mathbf{z} &= \theta_\mathbf{x} + \delta \end{aligned} \tag{8.38}$$

8.3.3 Occlusion and FOV Model

The aim of the occlusion and FOV model is to enable tracking of objects that are temporarily occluded or are located at the boundaries of the sensing FOV. This is achieved by setting the probability of detection $p_{D,k}(m^{(i)}_{k|k-1})$ to a low value for targets that are not likely to be detected by a sensor. If this is not performed, the C-GM-PHD filter would quickly decrease weights of

those targets and discard them.

We adapt the occlusion model of Granström et al. [92] by verifying occlusions for the two rectangle corners (with the smallest and the largest bearing angle), and augment it with FOV verification. Occlusions are verified against all other objects that are estimated to be in-between the sensor and the object of interest. The decrease in probability of detection is proportional to the weight of the estimate. One occluded corner decreases p_D by maximum 0.5. Gaussian kernels are used to smooth the transition from visible to occluded area. To keep track of targets that are outside of a sensor's FOV (through cooperation), we place the Gaussian kernels along the boundary of the sensor's FOV. They decrease p_D for targets that are at the boundary or outside of the FOV. The algorithm is described in Algorithm 8.2.

8.4 Experimental Evaluation

Experimental evaluation is performed in a submicroscopic high-fidelity simulator Webots (see Chapter 5 for description of the simulated models and their calibration). In the remainder of this section, we explain our experimental setup and the obtained results.

8.4.1 Experimental Setup

In our scenario, we use five simulated Citroën C-ZERO cars: two cars are used for sensing the other three represent legacy cars to be tracked. This setup allows us to reproduce a potential real-world scenario in a flexible way. Figure 8.3 shows a screenshot of our simulation environment.

Each of the two cars used for sensing is equipped with a front-facing Ibeo LUX LIDAR, a GNSS device, a compass and a wireless communication device. Resolution, accuracy, range, and other sensors' characteristics are calibrated using sensors' data sheets. The first sensing car is a stationary ego car ($\mathcal{E}$), while the second is a stationary cooperative car ($\mathcal{C}$). They are positioned in a way that their FOVs overlap partially. Two sensing cars are sufficient to showcase the proposed method, although the method supports multiple sensing cars. The remaining three non-cooperative cars ($\mathcal{NC}_i, i \in \{1,2,3\}$) move in an open space. Car $\mathcal{C}$ sends its PHD intensity augmented by its own localization state to $\mathcal{E}$. $\mathcal{NC}$ cars do not have any sensors and do not share any information. Poses and trajectories of cars are described in Figure 8.4.

Our dataset contains the data we collected during 20 distinctive simulation runs. In each run, $\mathcal{NC}$ vehicles ran an open-loop controller following pre-defined trajectories. Actuator and sensor (e.g., LIDAR) noise contribute to the randomness of the generated sensing data.

We then run the proposed C-GM-PHD filter in Matlab, feeding it with measurements from the dataset. We vary the localization accuracy by adding Gaussian noise with zero mean and standard deviation (SD) of $\sigma_l \in \{0, 0.5, 1\}$ to outputs of GNSS and compass devices (in m and °, respectively). We also vary the communication rate relatively to the sensing rate.

Algorithm 8.2: Occlusion and FOV Model

Require: Object of interest $\mathbf{x}$, set of all object estimates $\{\hat{\mathbf{x}}^{(i)}\}_{i=1}^{N}$ with weights $w^{(i)}$, sensor range r_s and FOV β_s

Gaussian kernel: $g(x;\mu,\sigma) = 0.5\exp(-((x-\mu)/\sigma)^2)$
$p_D = 0, \sigma_{r_s} = 1\text{m}, \sigma_{\beta_s} = 0.25°, \sigma_\beta = 1.5°$
Compute range and bearing to object corners: $r_j, \beta_j, j = 1,2,3,4$
Minimum/maximum bearings: $j_- = \arg\min_j \beta_j$; $j_+ = \arg\max_j \beta_j$
Mean range: $r_-^+ = 0.5(r_{j_+} + r_{j_-})$
Object at the boundary of the FOV:
if $-\beta_S/2 \le \beta_{j_-} \le \beta_S/2$ **then**
    $p_D = p_D + (0.5 - g(\beta_{j_-}; -\beta_s/2, \sigma_{\beta_s}) - g(\beta_{j_-}; \beta_s/2, \sigma_{\beta_s}))$
end if
if $-\beta_S/2 \le \beta_{j_+} \le \beta_S/2$ **then**
    $p_D = p_D + (0.5 - g(\beta_{j_+}; -\beta_s/2, \sigma_{\beta_s}) - g(\beta_{j_+}; \beta_s/2, \sigma_{\beta_s}))$
end if
Enforce maximum probability of detection: $p_D = \min\{p_D, p_D^{max}\}$
if $r_-^+ \ge r_S$ **then**
    $p_D = 0$
else
    $p_D = p_D - 2g(r_-^+; r_s, \sigma_{r_s})$
end if
Object occluded:
for $i = 1, \ldots, N$ **do**
    Range/bearing to corners of estimate: $\hat{r}_k^{(i)}, \hat{\beta}_k^{(i)}, k = 1,2,3,4$
    Minimum/maximum bearings: $k_- = \arg\min_k \hat{\beta}_k, k_+ = \arg\max_k \hat{\beta}_k$
    Mean range: $\hat{r}_-^+ = 0.5(\hat{r}_{k_+}^{(i)} + \hat{r}_{k_-}^{(i)})$
    if $\hat{r}_-^+ < r_-^+$ **and** $\hat{\beta}_{k_-}^{(i)} \le \beta_{j_-} \le \hat{\beta}_{k_+}^{(i)}$ **then**
        $p_D = p_D - (0.5 - g(\beta_{j_-}; \hat{\beta}_{k_-}^{(i)}, \sigma_\beta) - g(\beta_{j_-}; \hat{\beta}_{k_+}^{(i)}, \sigma_\beta))$
    end if
    if $\hat{r}_-^+ < r_-^+$ **and** $\hat{\beta}_{k_-}^{(i)} \le \beta_{j_+} \le \hat{\beta}_{k_+}^{(i)}$ **then**
        $p_D = p_D - (0.5 - g(\beta_{j_+}; \hat{\beta}_{k_-}^{(i)}, \sigma_\beta) - g(\beta_{j_+}; \hat{\beta}_{k_+}^{(i)}, \sigma_\beta))$
    end if
    Enforce minimum probability of detection: $p_D = \max\{p_D, p_D^{min}\}$
end for
return Probability of detection p_D for object of interest

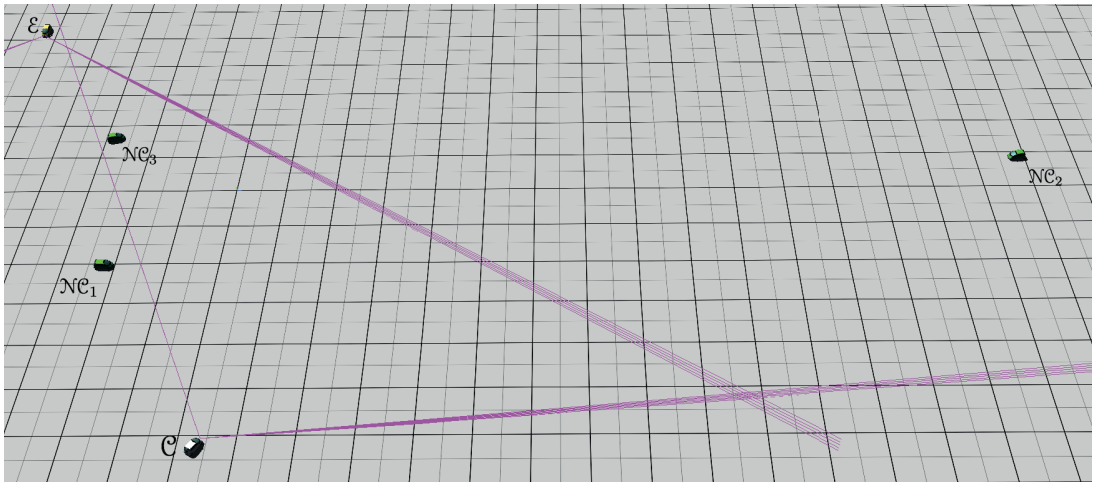


Figure 8.3 – A screenshot from the Webots simulator showing the experimental scenario.

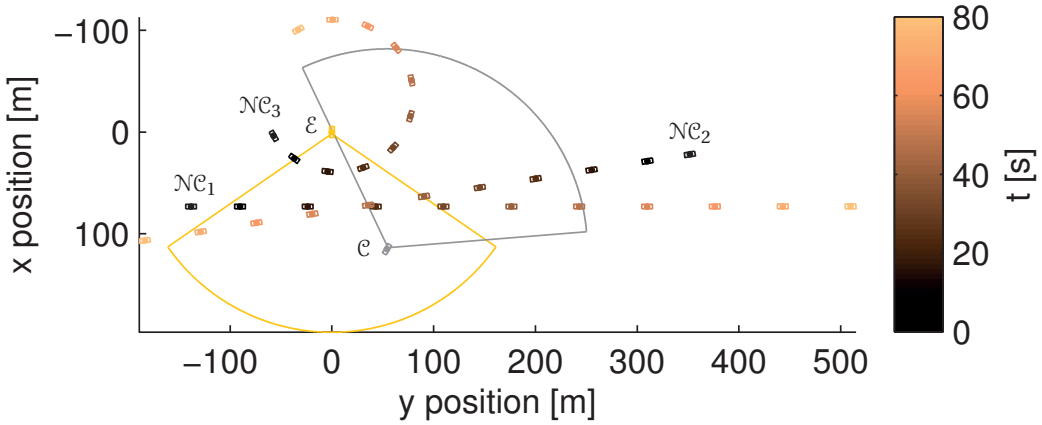


Figure 8.4 – Ground truth trajectories shown in the local frame of $\mathcal{E}$. Time is color coded, being black at 0 s and orange at 80 s. Cars $\mathcal{NC}_1$ and $\mathcal{NC}_2$ move on a straight line, while $\mathcal{NC}_3$ moves with rotational motion. Cars $\mathcal{E}$ and $\mathcal{C}$ and their fields of view are shown in yellow and gray, respectively. *Note: car rectangles are scaled up 3x for visibility reasons.*

In particular, we consider $\mathcal{T} \in \{1,2,5\}$, with a communication rate of $\mathcal{T} = n$ meaning that information is exchanged between cars each n samplings of the LIDAR. The LIDAR sampling rate is instead constant (80 ms) and corresponds to a simulation step in Webots and to the scanning frequency of the real Ibeo (12.5 Hz). For the sake of simplicity, there is no package loss and no communication delay in our current implementation (but dealing with communication delays is straightforward when tracks and motion models are shared).

Parameters

The tracking parameters were the same in all runs. The maximum number of Gaussian components per filter J_{max} is limited to 30 for the execution speed reasons. The birth model for the GM-PHD filter is a sum of eight Gaussian components with weight 0.001 located around the

sensory FOV boundary. The birth speed is 8 m/s, the heading is normal to the FOV boundary with direction towards inside of the FOV, and the turn rate is set to 0.025 rad/s. The SD for a birth component is 50 m for the lateral and 30 m for the longitudinal position, 6 m/s for the speed, π rad for the heading, and 0.2 rad/s for the turn rate. A new target entering the sensing FOV is very likely to be contained within one SD from one of the birth component means. The SD for the noise in the motion model is set to $\sigma_a = 1$ m/s^2 and $\sigma_\alpha = 0.1$ rad/s^2. The measurement noise is determined empirically and is modeled as a Gaussian with SD $\sigma_x = 2$ m, $\sigma_y = 2$ m, and $\sigma_\theta = 0.78$ rad. The clutter model uses a Poisson distribution with an expected cardinality of 10 measurements per sensor surveillance area. The merging parameter $U = 6$ and a Gaussian component is pruned if its weight is less than $T_p = 10^{-5}$. The extraction threshold T_e is set to 0.5. The fusion distance parameter U_F is empirically set to 50. The Ibeo LUX LIDAR has a FOV of 110° and a range of 200 m. The probability of survival in the joint FOV is set to 0.99. The probabilities of detection are set as follows: $p_D^{max} = 0.98$ and $p_D^{min} = 0.02$. The shape of the object is assumed to be rectangular, with length of 3.5 m and width of 1.5 m.

The literature suggests optimization methods for choosing a good value for the CI fusion weight $\mathcal{W}$ [83], as well as approximate solutions based on the fused covariance trace or determinant minimization [93]. However, Uney et al. [94] suggest that fixing $\mathcal{W}$ at 0.5 produces near optimal results on average. Therefore, the value for $\mathcal{W}$ is set to 0.5 in our experiments. This choice will be further refined in Part IV.

8.4.2 Results

We evaluate the performance of the C-GM-PHD filter by comparing its output to the ground truth data measured at the supervisor level in the high-fidelity simulator Webots. For multi-object performance evaluation we use the Optimal Sub-Pattern Assignment (OSPA) metric [95] defined for two arbitrary finite sets $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$, where $m, n \in \mathbb{N}_0$:

$$\bar{d}_p^{(c)}(X,Y) \triangleq \left(\frac{1}{n} \left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)})^p + c^p (n-m) \right) \right)^{1/p} \tag{8.39}$$

if $m \leq n$, and $\bar{d}_p^{(c)}(X,Y) \triangleq \bar{d}_p^{(c)}(Y,X)$ if $m > n$. The distance between x and y cut-off at $c > 0$ is denoted by $d^{(c)}(x,y) \triangleq \min(c, d(x,y))$, and Π_q is the set of permutations on $\{1,2,\ldots,q\}$ for any $q \in \mathbb{N}$. If $m = n = 0$, then the OSPA is set to 0 by definition.

As can be seen from Equation (8.39), the OSPA is comprised of two components, the first accounting for localization and the second for cardinality errors of the estimates. The OSPA is computed using the association between the sets X and Y that yield the smallest error. We choose the order (i.e., the sensitivity of the metric in penalizing estimated position) $p = 2$, and the cut-off parameter for cardinality errors penalties $c = 60$. The distance metric in $d(x,y)$ is the Euclidean distance between estimated and ground truth position. In general, the better performance, the lower OSPA error.

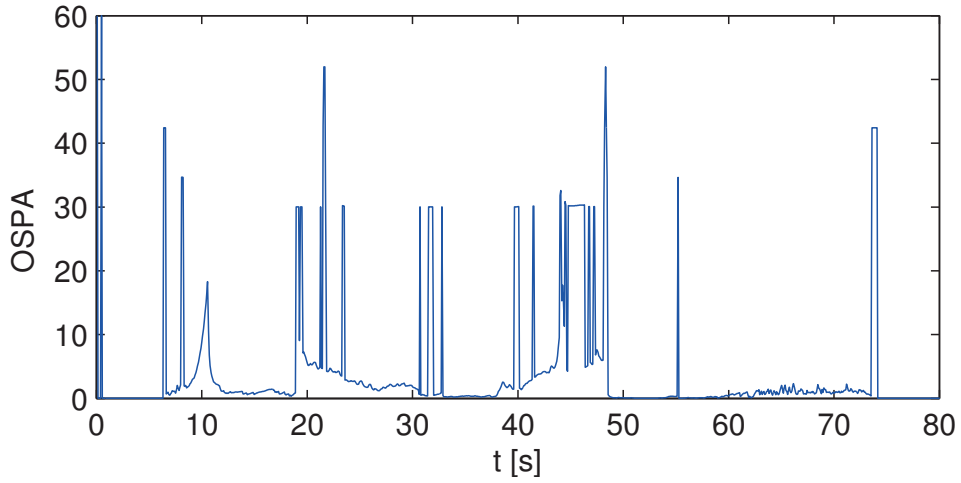


Figure 8.5 – OSPA distance versus time for a single run.

Figure 8.5 displays the OSPA error for a single run of C-GM-PHD filter, for $\sigma_l = 0$ and $\mathcal{T} = 1$. Figure 8.6 shows the localization and cardinality OSPA components. The target set cardinality was correctly estimated 91% of time. Short periods of wrong cardinality estimation, as shown in Figure 8.6, are the result of the filter delay (multiple consecutive observations of a new object are needed for the filter to give birth to a new target), transitions between the FOV of $\mathcal{E}$ and $\mathcal{C}$ vehicles, and occlusions. When a target goes from the FOV of only one filter to the common FOV, the C-GM-PHD filter might lose track of it for a short moment. This is due to fusing two Gaussian components, one with $w \approx 1$ and the second one with a low weight (e.g., target in the birth phase). For a chosen value of $\mathcal{W} = 0.5$, CI gives equal weights to the two Gaussian components, hence the weight of the fused component may get lower than the extraction threshold $T_e = 0.5$. While the choice of $\mathcal{W} = 0.5$ may be justified for the case when sensors' FOVs overlap fully and sensors are characterized by same noise, in our case an optimization of $\mathcal{W}$ is needed to correctly tackle this issue. We thus address the weight optimization problem in Part IV).

The occlusion and FOV model presented in Section 8.3.3 is dependent on the targets' position estimate error, and it happens that a detectable target is assigned low p_D or vice-versa. This can as well happen when the target is near to the boundaries of the FOV. Nevertheless, it is important to note that all wrong cardinality estimates represent false negatives, which is desirable when driver assistance systems are considered (this is however not the case for autonomous vehicles, where eliminating false negatives represents a crucial step towards achieving a save operation). The DBSCAN clustering algorithm discards all clusters with less than two measurements. In cases when two or more clutter measurements fall next to each other, the PHD filter successfully filters them out. Statistics over 20 runs for all combinations of σ_l and $\mathcal{T}$ is given in Figure 8.7. As expected, OSPA error increases with localization error and decreases with higher communication rate.

The estimated position of car $\mathcal{NC}_1$ is shown in Figure 8.8, together with its associated un-

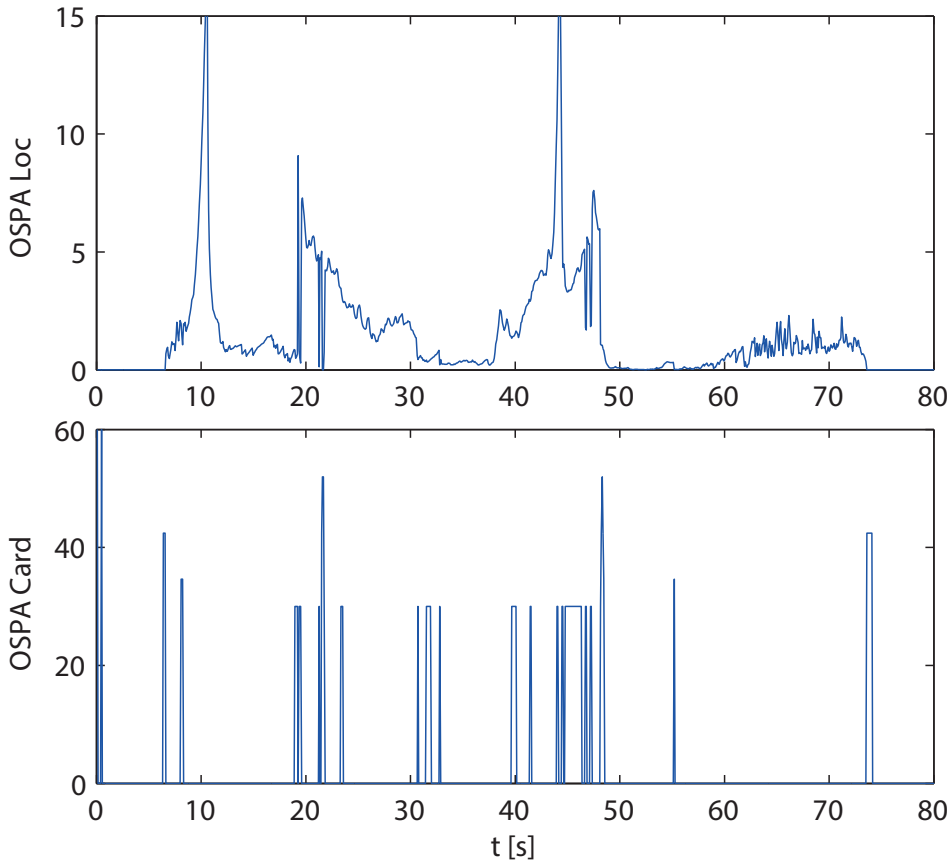


Figure 8.6 – Separate OSPA components for a single run. (Top) localization component versus time; (Bottom) cardinality component versus time.

certainty, for $\sigma_l = 1$ and $\mathcal{T} = 5$. Uncertainty is notably higher at around $t = 10$ s, when $\mathcal{NC}_1$ was occluded and therefore no measurement-based corrections were applied. After $t = 30$ s, $\mathcal{NC}_1$ leaves the FOV of car $\mathcal{E}$, and is thus only observable by car $\mathcal{C}$. The estimated position uncertainty is significantly higher in this region due to localization uncertainties of cars $\mathcal{C}$ and $\mathcal{E}$, as expected.

The performance of the C-GM-PHD filter is further compared to the performance of the non-cooperative GM-PHD filter, and the result is shown in Figure 8.9. A fair comparison can only be done in the FOV of $\mathcal{E}$ (because a non-cooperative filter does not track objects outside of this area). Thus, the ground truth targets are considered only in this region, and the C-GM-PHD filter is modified to perform the cooperative fusion only if targets are found in the FOV of $\mathcal{E}$. We observe that, thanks to cooperation, the C-GM-PHD filter needs less time to start tracking the first target, $\mathcal{C}$. With the exception of two peaks (moments when the C-GM-PHD briefly loses targets around $t = 22$ and 25 s, respectively), the C-GM-PHD filter performs consistently better or equal than its non-cooperative version.

The total tracking time of all objects is listed in Table 8.1. We can observe the added value of

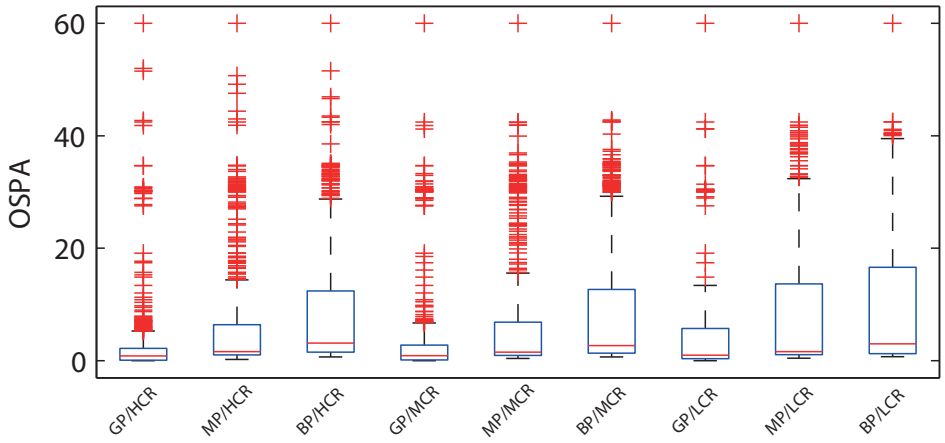


Figure 8.7 – OSPA statistics for 20 runs, for various values of localization (GNSS and compass) white noise standard deviation σ_l, and communication rate $\mathcal{T}$ parameters. GP, MP, BP stand for good, medium and bad positioning, i.e., $\sigma_l = 0, 0.5$ and 1, respectively. HCR, MCR and LCR stand for high, medium and low communication rate, i.e., $\mathcal{T} = 1, 2$ and 5, respectively.

Table 8.1 – Total tracking time per target in the union of the FOVs.

	$\mathcal{C}$	$\mathcal{NC}_1$	$\mathcal{NC}_2$	$\mathcal{NC}_3$
GM-PHD	78.7 s	24.1 s	18.6 s	33.5 s
C-GM-PHD	79.9 s	37.9 s	47.3 s	53.5 s

the extended FOV and filter reactiveness that is offered by the C-GM-PHD filter.

Summary

We presented an algorithm for cooperative car tracking that we named C-GM-PHD filter. It is based on the GM-PHD filter, which is run independently on each of the cooperative vehicles. Additionally, the ego vehicle fuses the PHD intensity that it receives from the cooperative vehicle. The proposed fusion algorithm is our main contribution in this chapter. We performed evaluation in the high-fidelity simulator Webots in a scenario that hosted a total of 5 vehicles: two sensing cars and three legacy cars.

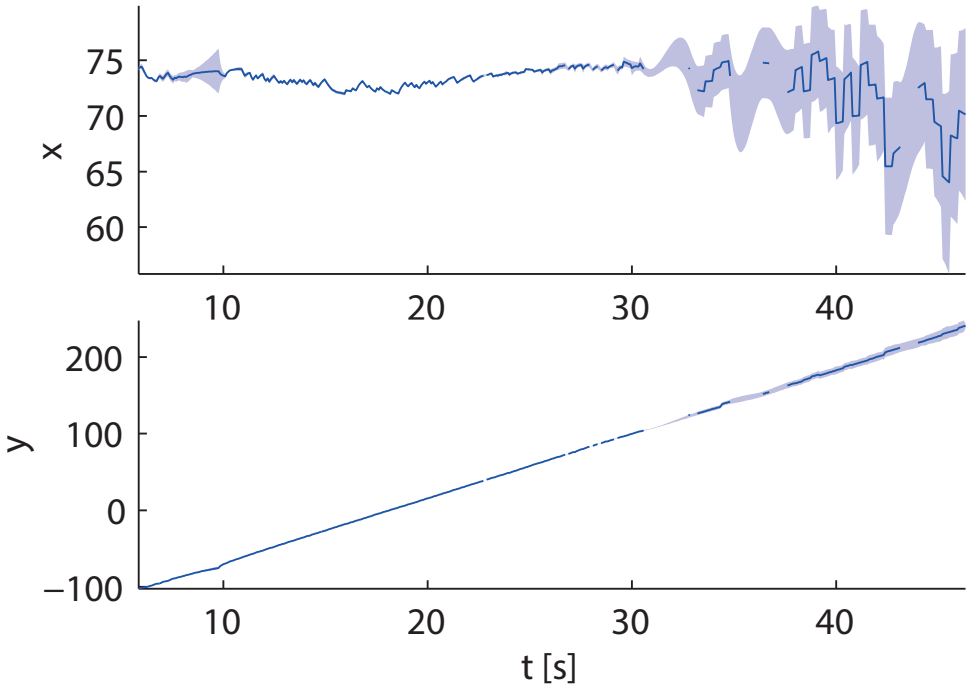


Figure 8.8 – The estimated trajectory of target $\mathcal{NC}_1$ with associated uncertainty, for a single run with parameters $\sigma_l = 1$ and $\mathcal{T} = 5$. For periods in which the target is lost (e.g., at around $t = 32$ s), the estimated trajectory is not drawn, whereas the uncertainty is interpolated.

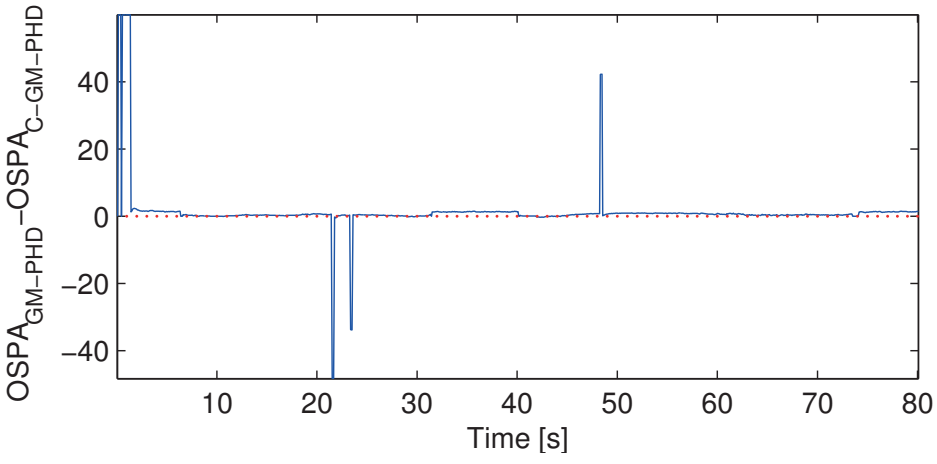


Figure 8.9 – Difference between a non-cooperative GM-PHD filter and a C-GM-PHD filter expressed using OSPA. Red line is a marker at zero.

9 Cooperative Multi-Object Tracking Using a Sequential Monte Carlo Approximation

COMMON implementations of the PHD filter include the GM and SMC PHD filter. In Chapter 8, we presented an approach for tracking and fusion of information between two cars each running an instance of the GM-PHD filter. In this chapter, we propose an integrated approach of localization with multiple target tracking, factoring uncertainties from the localization itself into tracking. For cooperative fusion we propose the use of a Cooperative SMC-PHD (C-SMC-PHD) filter that builds upon the SMC-PHD filter introduced in [67] and extends it with fusion of SMC-PHD densities. Our novel fusion approach generalizes the EMD method introduced in [83] by dealing with information acquired by a cooperating vehicle outside of the tracking vehicle's FOV. In this approach, fusion of the two SMC-PHD intensities is performed in the global frame, after factoring in the localization uncertainty of the sensing vehicles.

The general concept of the C-SMC-PHD is a five-step process illustrated in Figure 9.1, where the prediction, update, and target extraction steps are taken from [67] and are summarized in Sectopn 9.1 for completeness. Our distributed fusion approach is presented in Section 9.2.

9.1 Multi-Object Tracking Using an SMC-PHD Filter

The PHD filter overcomes the impracticality of the implementation of a multi-target Bayes nonlinear filter by propagating only the first-order moment of the multi-object posterior density [96]. Like any recursive Bayesian filter, the PHD filter contains the prediction and update steps, both of which are carried out by a random sample approximation using an SMC implementation, such that the posterior of the PHD intensity at time $k-1$ is approximated by a random sample set as follows:

$$\nu_{k-1}(\mathbf{x}) \approx \sum_{n=1}^{N} w_{k-1}^{n} \delta_{\mathbf{x}_{k-1}^{n}}(\mathbf{x}) \tag{9.1}$$

where $\delta_{\mathbf{x}_{k-1}^{n}}(\mathbf{x}) = \delta(\mathbf{x}-\mathbf{x}_{k-1}^{n})$ is the Dirac delta function and $\{(w_{k-1}^{n}, \mathbf{x}_{k-1}^{n})\}_{n=1}^{N}$ a weighted particle set comprising of N random samples $\mathbf{x}_{k-1}^{n}$ and its corresponding weights w_{k-1}^{n}. Furthermore,

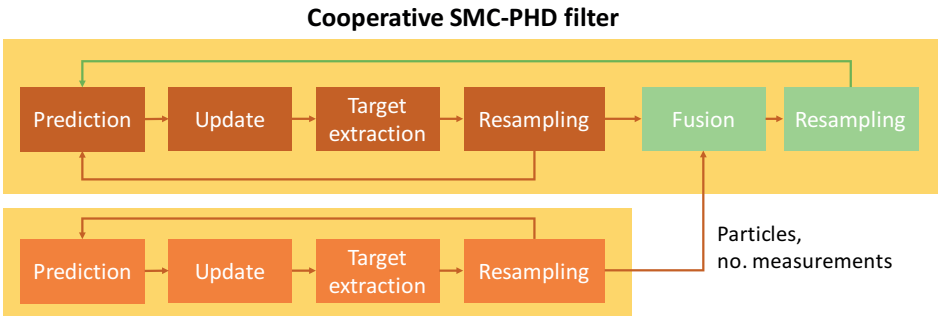


Figure 9.1 – The C-SMC-PHD filter illustrated. Fusion of intensities from cooperating agents is carried out only if information is available. Otherwise, the filter of the tracking agent continues without cooperation. The blocks in green represent our contributions while the other blocks represent the state-of-the-art approach.

the number of targets can be extracted from the particle set by summing up the importance weights for large values of N. Ristic et al. also proposed in [67] that particles should be *spawned* around the measurement set Z_k at every time step k to avoid inefficient use of particles in estimating the posterior. The distinction between *newborn* and *persistent* particles (bearing indices b and p, respectively) is evident through the recursive implementation of the filter.

9.1.1 Prediction

In the prediction step, ρ newborn particles of states $\mathbf{x}^n_{k|k-1,b}$ are generated around each measurement $\mathbf{z} \in Z_k$ with a uniform weight

$$w^n_{k|k-1,b} = \frac{\gamma^b_{k|k-1}}{\rho m_k}, \tag{9.2}$$

where m_k is the cardinality of measurements at time k and $\gamma^b_{k|k-1}$ is the expected number of newborn targets between time $k-1$ and k.

Additionally, the states of persistent particles $\mathbf{x}^n_{k|k-1,p}$ are propagated according to the dynamics defined by the single-object transition density. Weights of persistent particles are also revised according to the survival probability p_S given by:

$$w^n_{k|k-1,p} = p_S(\mathbf{x}^n_{k-1})\, w^n_{k-1,p}. \tag{9.3}$$

9.1.2 Update

In this step, the weights of the persistent ($w^n_{k|k-1,p}$) and newborn ($w^n_{k|k-1,b}$) particles representing both their respective intensities are updated as

$$w^n_{k,p} = \left[1 - p_D(\mathbf{x}^n_{k|k-1,p})\right] w^n_{k|k-1,p} + \sum_{\mathbf{z}\in Z_k} \frac{p_D(\mathbf{x}^n_{k|k-1,p})\, g_k(\mathbf{z}|\mathbf{x}^n_{k|k-1,p})\, w^n_{k|k-1,p}}{\mathcal{L}(\mathbf{z})} \tag{9.4}$$

for persistent particles and

$$w^n_{k,b} = \sum_{\mathbf{z}\in Z_k} \frac{w^n_{k|k-1,b}}{\mathcal{L}(\mathbf{z})} \tag{9.5}$$

for newborn particles, where

$$\mathcal{L}(\mathbf{z}) = \kappa_k(\mathbf{z}) + \sum_{n=1}^{\rho m_k} w^n_{k|k-1,b} + \sum_{n=1}^{N_{k-1}} p_D(\mathbf{x}^n_{k|k-1,p})\, g_k(\mathbf{z}|\mathbf{x}^n_{k|k-1,p})\, w^n_{k|k-1,p}. \tag{9.6}$$

$p_D(\mathbf{x})$ furthermore represents the detection probability of a target with state $\mathbf{x}$ while $g_k(\mathbf{z}|\mathbf{x})$ represents the single-target measurement likelihood at time k. Measurement-associated clutter is also factored in with the term $\kappa_k(\mathbf{z})$.

9.1.3 State Estimation Extraction

It is noted that the second term in Equation (9.4) can essentially be viewed as a sum of weights corresponding to each measurement $\mathbf{z} \in Z_k$. If each measurement in the measurement set is indexed $Z_k = \{\mathbf{z}_{k,j} | \, j = 1, 2, ..., m_k\}$, an additional weight metric for each measurement of index j where $j = \{0, 1, ..., m_k\}$ can be defined:

$$w^{n,j}_{k,p} = \begin{cases} \left[1 - p_D\left(\mathbf{x}^n_{k|k-1,p}\right)\right] w^n_{k|k-1,p} & j = 0 \\ \frac{p_D(\mathbf{x}^n_{k|k-1,p})\, g_k(\mathbf{z}|\mathbf{x}_{k|k-1,p})\, w^n_{k|k-1,p}}{\mathcal{L}(\mathbf{z})} & j = 1, ..., m_k \end{cases}. \tag{9.7}$$

The index $j = 0$ represents the measurement-independent first term of Equation (9.4).

Therefore, each measurement index j can be given an associated weight $W^j_{k,p} = \sum_{n=1}^{N_{k-1}} w^{n,j}_{k,p}$ with $0 \leq W^j_{k,p} \leq 1$. Intuitively, $W^j_{k,p}$ would be high for a certain measurement if $\mathbf{z}_{k,j}$ resulted in non-zero likelihood for some particles, indicating that the measurement originated from a target, and close to zero otherwise. The state estimate to each measurement index is then only reported if the weight $W^j_{k,p}$ above a given reporting threshold T_e.

Correspondingly, the state estimate $\hat{\mathbf{x}}_{k,j}$ and its associated covariance matrix $P_{k,j}$ can be

obtained as:

$$\hat{\mathbf{x}}_{k,j} = \sum_{n=1}^{N_{k-1}} w_{k,p}^{n,j} \mathbf{x}_{k|k-1,p}^{n}$$
$$P_{k,j} = \sum_{n=1}^{N_{k-1}} w_{k,p}^{n,j} \left(\mathbf{x}_{k|k-1,p}^{n} - \hat{\mathbf{x}}_{k,j}\right)\left(\mathbf{x}_{k|k-1,p}^{n} - \hat{\mathbf{x}}_{k,j}\right)^{T}. \tag{9.8}$$

9.1.4 Resampling

The resampling step represents a core component of SMC methods and ensures that the filter contains more particles of higher weight and fewer of those with lower weights. In our implementation, we use the low variance resampling [84] (or more recently known as the resampling wheel). At the end of this step, the importance weights of all persisting particles are set to $1/N_{k-1}$ since weight-based likelihoods are replaced by frequencies when samples are drawn by weights.

9.2 Cooperative Fusion

Given that each agent runs an instance of the SMC-PHD filter as shown in Figure 9.1, the goal of fusion to is to obtain the fused PHD posterior intensities of both PHD posteriors. In this case, our contribution involves providing a generalized approach to fuse PHD posterior intensities where the FOVs of the cooperating agents are not completely overlapping. This is achieved by using the EMD approach introduced in [83] for completely overlapping segments of the field of view and a novel method for incorporating information from the cooperating agent that is outside of the tracking agent's FOV.

Upon receiving particles from the cooperating agent and transforming them to the global reference frame, we cluster particles over the whole state space using the DBSCAN algorithm [85] to obtain a set of R particle clusters $\{r_l | l = 1, 2, ..., R\}$ and obtain the mean of each cluster $\bar{p}_l$. Instead of a binary assignment $\bar{p}_l \in FOV_{own}$ or $\bar{p}_l \notin FOV_{own}$, we employ the logistic function defined as:

$$f_p(x) = \frac{1}{1 + \exp(-\rho \cdot \text{ang}_{\theta_{sup}}(x))} \tag{9.9}$$

to obtain a continuous probability weighting in the immediate vicinity of FOV_{own}. $\text{ang}_{\theta_{sup}}(x)$ represents the angular difference from particle x to the extreme angles θ_{sup} of the field of view. The logistic function is hence calculated with respect to the angle of the FOV as visualized in Figure 9.2.

A simple heuristic is used to decide whether to consider received particles as *external* to the tracking vehicle's FOV or as *common* to both FOVs is given in lines 1 – 14 of Algorithm 9.1. This non-binary classification of particles ensures that transitions between fields of view occur

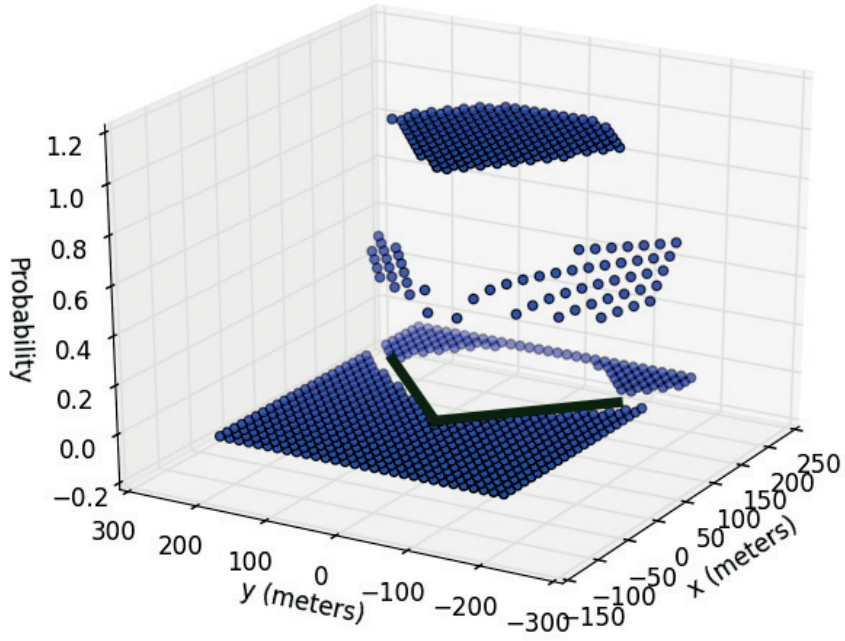


Figure 9.2 – The logistic function with $\rho = \pm 8$ with θ_{sup} being the extreme angles of the field of view of the LIDAR sensor.

smoothly and target estimates are not lost.

Bearing in mind that the fusion algorithm proposed in [83] requires that two FOVs intersect, we are still left with removing particles that might be in $FOV_{internal} := FOV_{own} \backslash FOV_{other}$. This is achieved by determining the area defined by the intersection of two fields of view FOV_{own} and FOV_{other} by representing them as convex polygons. The intersection set of two convex polygons can be obtained in linear time [97] and is also a convex polygon itself. Lines 15 – 24 elaborate on the procedure for finding particles that belong to the common FOV.

9.2.1 Fusion within the Common FOV

Given two PHD posterior intensities of two i.i.d. cluster distributions f_i and f_j, the EMD of these two cluster distributions is also an i.i.d. cluster distribution defined as

$$f_{(\cdot)}(X) = n! \cdot \mu_{(\cdot)}(n) \prod_{x \in X} \nu_{(\cdot)}(x), \tag{9.10}$$

Algorithm 9.1: Procedure for classifying received particles.

$\xi_{ext} \leftarrow \emptyset$ {ext. particle set}
$\xi_{cm} \leftarrow []$ {common particle set}
for $r_l \in \{r_l | l = 1, 2, .., R\}$ **do**
  $p \leftarrow f_p(\bar{p}_l)$
  if p = 0 **then**
    Insert r_l into ξ_{ext}.
  else
    Penalize r_l by p.
    Insert r_l into ξ_{cm}.
    if $p < 1$ **then**
      Insert r_l into ξ_{ext}.
    end if
  end if
end for
$\xi_{intersect} \leftarrow []$
$\mathcal{P}_{own} \leftarrow$ Polygon(FOV_{own}) {Create polygon from FOV}
$\mathcal{P}_{other} \leftarrow$ Polygon(FOV_{other})
$\mathcal{P}_{intersect} \leftarrow \mathcal{P}_{own} \cap \mathcal{P}_{other}$
for $(w^n, \mathbf{x}^n) \in \{(w^n, \mathbf{x}^n)\}_{n=1}^{N}$ **do** {own particle set}
  if $\mathbf{x}^n \in \mathcal{P}_{intersect}$ **then**
    Insert $(w^n, \mathbf{x}^n)$ into $\xi_{intersect}$
  end if
end for
$\xi_{cm} \leftarrow \xi_{cm} \cup \xi_{intersect}$
return ξ_{ext}, ξ_{cm}

where

$$\nu_{\mathcal{W}}(x) = \frac{\nu_i^{\mathcal{W}}(x)\nu_j^{1-\mathcal{W}}(x)}{Z_{\mathcal{W}}\left(\nu_i, \nu_j\right)}, \tag{9.11}$$

$$\mu_{\mathcal{W}} = \mu_i^{\mathcal{W}} \cdot \mu_j^{1-\mathcal{W}} \cdot Z_{\mathcal{W}}(\nu_i, \nu_j), \tag{9.12}$$

$$Z_{\mathcal{W}}\left(\nu_i, \nu_j\right) = \int \nu_i(x)^{\mathcal{W}} \nu_j(x)^{1-\mathcal{W}} \mathrm{d}x. \tag{9.13}$$

In our case, we assume a Poisson distributed cardinality from both nodes given by the means $\mu_{(\cdot)}$. The resulting cardinality is hence also Poisson distributed.

The parameter $\mathcal{W} \in [0,1]$ determines the relative weight assigned to each distribution and needs to be extrinsically chosen. The optimal $\mathcal{W}$ can be obtained by minimizing a cost function defined by the Rényi Divergence (RD) [98] and is given by

$$\mathcal{W}^* = arg\min_{\mathcal{W} \in [0,1]} \left(R_\alpha(f_{\mathcal{W}} || f_i) - R_\alpha(f_{\mathcal{W}} || f_j)\right)^2, \tag{9.14}$$

where

$$R_\alpha\left(f_\mathcal{W}||f_i\right)=\frac{1}{\alpha-1}\log\sum_{n=0}^{N_{max}}\left[\mu_\mathcal{W}^\alpha(n)\mu_i^{(1-\alpha)}(n)\left(\frac{Z_{\alpha\mathcal{W}}\left(\nu_i,\nu_j\right)}{Z_\mathcal{W}\left(\nu_i,\nu_j\right)^\alpha}\right)^n\right], \tag{9.15}$$

$$R_\alpha\left(f_\mathcal{W}||f_j\right)=\frac{1}{\alpha-1}\log\sum_{n=0}^{N_{max}}\left[\mu_\mathcal{W}^\alpha(n)\mu_j^{(1-\alpha)}(n)\left(\frac{Z_{\alpha(1-\mathcal{W})}\left(\nu_i,\nu_j\right)}{Z_\mathcal{W}\left(\nu_i,\nu_j\right)^\alpha}\right)^n\right] \tag{9.16}$$

and N_{max} representing the maximum number of targets that would be tracked by the filter at any time. $\mu_x(n)$ represents the probability mass function of the Poisson distribution over n with a mean λ_x being the cardinality of node x. Here $0<\alpha<1$ is a parameter which determines how much we emphasize the tails of two distributions in the metric. By taking the limit $\alpha\to 1$ or by setting $\alpha=0.5$, the RD becomes the Kullback-Leibler divergence and the Hellinger affinity, respectively.

Bearing in mind that both PHD posteriors are implemented with SMC methods, the continuous approximations of the distributions have to be reconstructed from the particle sets of both distributions. This is achieved with the Kernel Density Estimation (KDE) method [99], in which the estimated density is obtained from the sum of kernel functions at particle points. An explicit particle to measurement association is also created by clustering particles using the DBSCAN [85] algorithm to construct separate KDEs over each cluster to ensure the closest approximation of the true posterior density instead of an over-smoothed density function.

For each cluster $\Bbbk\in\mathcal{K}=\{\Bbbk_1,...,\Bbbk_\mathrm{K}\}$, the covariance $P_\Bbbk$ is computed to find the density estimate as

$$\hat{v}\left(\mathbf{x}|Z_{1:k}\right)=\frac{1}{M}\sum_{m=1}^{M}K_h\left(\mathbf{x};\mathbf{x}^{(m)},P_{\Bbbk^m}\right), \tag{9.17}$$

where the *bandwidth* $h=n^{-1/(d+4)}$ is calculated according to Scott's Rule [100], with n the number of data points and d the number of dimensions.

The union of particle sets P_U from both nodes with $M_\mathcal{W}=M_i+M_j$ particles is obtained to sample from the fused state density defined in Equation (9.11) in order to obtain estimates $\hat{v}_i(x)$ and $\hat{v}_j(x)$. Thereafter, the estimate given in Equation (9.13) can be expressed as:

$$\hat{Z}_\mathcal{W}(\nu_i,\nu_j):=\sum_{x\in P_U}\frac{\hat{v}_i^{(\mathcal{W})}(x)\hat{v}_j^{(1-\mathcal{W})}(x)}{M_i\hat{v}_i(x)+M_j\hat{v}_j(x)}. \tag{9.18}$$

The weight of the m'-th particle in P_U is also estimated as:

$$\hat{\zeta}^{(m')}\propto\frac{\hat{v}_i^{(\mathcal{W})}\left(x^{(m')}\right)\hat{v}_j^{(1-\mathcal{W})}\left(x^{(m')}\right)}{M_i\hat{v}_i\left(x^{(m')}\right)+M_j\hat{v}_j\left(x^{(m')}\right)}, \tag{9.19}$$

resulting in the union particle set $\{(\zeta^{(m')},x^{(m')})\}_{m'=1}^{M_\mathcal{W}}$.

Algorithm 9.2: Procedure for handling particles representing external targets at a time step k

1: **if** New information from cooperating agent available **then**
2: Replace external particle set with incoming information from cooperating agent.
3: **end if**
4: External particles are propagated according to their dynamics and their weights reduced by the survival probability $p_{S_{ext}}$, analog to (9.3).
5: In the target extraction step, an external target corresponding to an external measurement j is only reported if $W_{k,p}^{j} \geq T_e$.
6: Sampling of external particles is carried out over each target estimate $\hat{\mathbf{x}}_{k,j}$ and its covariance $P_{k,j}$ according to Equation (9.8).

Having obtained Equation (9.18), Equation (9.14) can be computed for a different values of $\mathcal{W}$ using exhaustive search in equidistantly distributed points over the interval $[0,1]$. Finally, the weights in Equation (9.19) can be obtained for the optimal $\mathcal{W}^*$.

Thereafter, the particle set P_U and the set of particles in $FOV_{internal}$ are re-sampled separately. The union of P_U and the particles in $FOV_{internal}$ is created to be used in the prediction and update steps of the next iteration.[1]

9.2.2 Incorporating External Information

Unlike particles in the common FOV that can be predicted and updated according to measurements, external particles can only be predicted by the transitional densities, since the tracking agent receives no associated measurement. Algorithm 9.2 briefly explains the handling and propagation of external particles.

9.2.3 Propagation of Localization Uncertainty

For the particular context of multiple target tracking on a moving vehicle, we propose cascading of an UKF filter dedicated to the vehicle state estimation with a C-SMC-PHD filter as an approach to solve the full problem of localization of the tracking vehicle and the tracking of multiple objects. The UKF introduces a deterministic method for choosing a set of weighted *sigma points* approximating the distribution with a priori mean $\bar{\mathbf{x}}$ and covariance $\bar{P}_{xx}$ which are subjected to a nonlinear transform. Thereafter, the a posteriori mean $\bar{\mathbf{y}}$ and covariance $\bar{P}_{yy}$ of the distribution can be obtained by the weighted average and weighted outer product of the transformed points respectively.

The state and measurement are both defined by the position (x, y), orientation (φ) and speed (v) of the tracking vehicle: $\mathbf{x} = \mathbf{z} = [x, y, \varphi, v]$, emulating GNSS data with an on-board compass

[1] It should be noted that the number of particles chosen in the resampling step has to be identical in both agents when carrying out the fusion step. Otherwise, this might result in a problem due to unequal weights.

and using the following state transition function:

$$x_k = x_{k-1} + v \cdot sin(\varphi_{k-1}), \tag{9.20}$$

$$y_k = y_{k-1} + v \cdot cos(\varphi_{k-1}), \tag{9.21}$$

$$\varphi_k = \varphi_{k-1}, \tag{9.22}$$

$$v_k = v_{k-1}. \tag{9.23}$$

The a posteriori state estimate $\bar{\mathbf{x}}_k^l$ along with its error covariance $\bar{P}_k^l$ (where superscript l denotes variables used in the localization filter) of the UKF is then used as input in the C-SMC-PHD filter in order to transform tracked targets from the local coordinate frame of the tracking vehicle into the global coordinate frame. The representation of tracked targets in global coordinate frame is necessary as cooperating vehicles might not possess relative positional information.

As the objects tracked by the C-SMC-PHD filter are assumed to be vehicles in this context, they are modeled in the same way as the state of the tracking vehicle. Hence, their single state transition model is also defined by Equations (9.20) – (9.23). As such, we also assume that the measurement obtained contains the positional information as well as the orientation of tracked targets by preprocessing data from the laser range sensor, as described in Section 7.3. The likelihood model is given as a multivariate Gaussian distribution:

$$g_k(\mathbf{z}_k|\mathbf{x}_{k|k-1}) = \frac{1}{\sqrt{(2\pi)^3|\Sigma|}} \exp\left(-0.5(\mathbf{z}_k - \mathbf{x}_{k|k-1})^T \Sigma^{-1} (\mathbf{z}_k - \mathbf{x}_{k|k-1})\right) \tag{9.24}$$

where Σ represents the measurement uncertainty.

Bearing in mind that the preprocessing of data from LIDARs to extract the pose of tracked vehicles generally result in inaccuracies, especially at larger distances, the model provided in Equation (9.24) has to take into account both the uncertainty in localization of the tracking vehicle as well as uncertainty of the tracked vehicles. We hence reconsider Equation (9.24) such that

$$\Sigma = \bar{P}_k^l + \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_\varphi^2, \sigma_v^2), \tag{9.25}$$

where $\sigma_{(\cdot)}, (\cdot) = \{x, y, \varphi, v\}$ represents the standard deviation for x-y coordinates, the vehicle orientation in global coordinates and speed, respectively.

An illustration of this procedure is reported in Figure 9.3.

9.3 Experimental Evaluation

The experimental evaluation of the C-SMC-PHD algorithm is carried out using Webots. Simulated Citroën C-ZERO cars were equipped with front-facing Ibeo LUX LIDARs, GNSS devices,

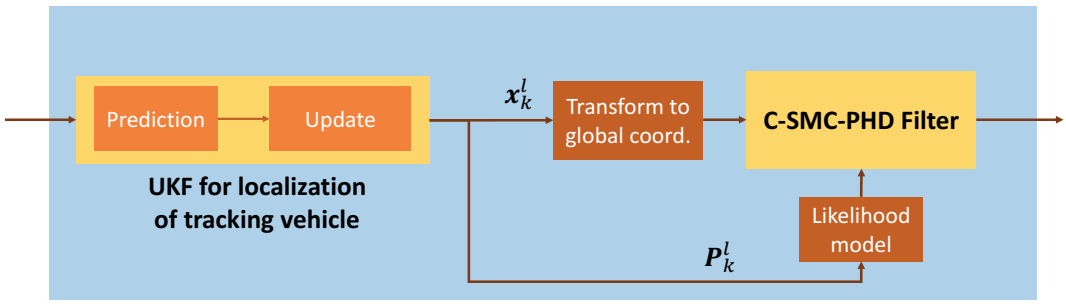


Figure 9.3 – An illustration of the cascading of an UKF and a C-SMC-PHD filter at time k.

wireless communication devices and compasses. The algorithm is implemented in Python, and the datasets collected in Webots are post-processed.

9.3.1 Scenario

A scenario involving two moving cars $(\mathcal{E},\mathcal{C})$, each running an instance of the state estimation — C-SMC-PHD filter tracking up to four moving cars $(\mathcal{NC}_1,\mathcal{NC}_2,\mathcal{NC}_3,\mathcal{NC}_4)$ in an open space is presented. Both $\mathcal{E}$ and $\mathcal{C}$ travel at constant velocity of 10 km/h. The LIDARs in this scenario have an effective range of 200 m, a horizontal FOV of 90°, and operate at a scanning frequency of 25 Hz.

The trajectories of the cars are shown in Figure 9.4. The speed of cars $\mathcal{NC}_i$, $i=\{1,2,3,4\}$ follows the function:

$$v(t)=\begin{cases}20 & 0\le t\le 4\\ 20-4\cdot(t-4) & 4<t\le 8\\ 4+4\cdot(t-8) & 8<t\le 18\end{cases} \tag{9.26}$$

where $v(t)$ is measured in km/h and t in seconds. This scenario aims to model a typical slow-speed traffic scenario as close as possible while showing the general case where two vehicles running tracking algorithms do not perceive the exact same scene: in the beginning only cars $\mathcal{NC}_1$ and $\mathcal{NC}_2$ are found in the overlapped zone; towards the middle of the scenario, they leave the union of the two FOVs, about the same time the cars $\mathcal{NC}_3$ and $\mathcal{NC}_4$ enter the common FOV.

9.3.2 Parameters

The UKF used for localization of the tracking vehicles was initialized with the following parameters intrinsic to a typical KF:

$$P_0 = R = \text{diag}(4.0^2, 4.0^2, 0.01^2, 0.01^2) \tag{9.27}$$

$$Q = \text{diag}(0.005, 0.005, 0.01, 0.01). \tag{9.28}$$

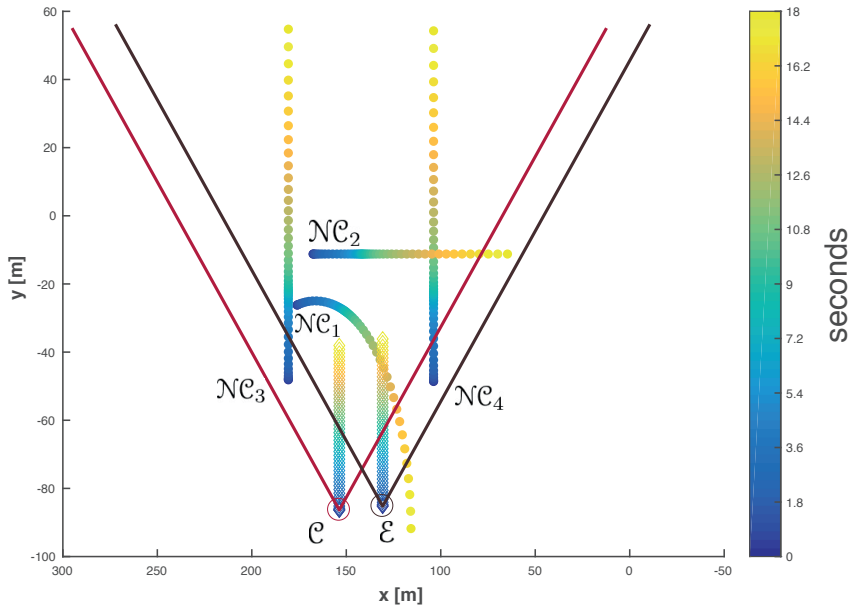


Figure 9.4 – The vehicle trajectories in the experimental scenario. The gray and red lines denote the FOVs of vehicles $\mathcal{E}$ and $\mathcal{C}$, respectively.

P_0 represents the initial estimate covariance, while the mean of the initial distribution was chosen as the first measurement of the simulation for each tracking vehicle. Q represents the covariance of the process noise. The first two entries of the observation model covariance matrix R correspond to the variance in GNSS measurements while the orientation of the vehicle was assumed to be determined by a compass with insignificant inaccuracy. An insignificant inaccuracy in determining the speed of the vehicle was also assumed in this case. Furthermore, the choice of sigma points was taken from [101].

For the C-SMC-PHD filter, 100 particles per persisting measurement were used while $\rho = 40$ particles were birthed around each new measurement at every iteration. The *spawning* of newborn particles at time k around a measurement follows a Gaussian distribution with the standard deviation of the position corresponding to the first two diagonal entries of P_k. A Poisson distribution with $\lambda_{clutter} = 1$ was used to simulate the occurrence of clutter resulting from multiple LIDAR measurements at each iteration. The *cooperation* step of the C-SMC-PHD was carried out every third *sensing* step to realistically model real world conditions where network communications often occur at a lower rate than sensor readouts.

Additionally, we chose the second term of Equation (9.25) to be $\text{diag}(1.3^2, 1.3^2, 0.1^2, 0.1^2)$. The standard deviations for the estimated position of the tracked vehicles were experimentally obtained from a scenario where the tracking vehicles were stationary.

9.3.3 Results

We present the results of the scenario with the given parameters along with a discussion of noteworthy observations. The dataset collected in this experiment contains 30 distinct simulation runs where each vehicle starts at the same positions and runs an open-loop controller. We note that the fusion of data for the cooperative component of the C-SMC-PHD filter is done only on the vehicle $\mathcal{E}$, unless otherwise mentioned. The following results are also discussed with respect to this latter vehicle.

The OSPA metric [95] was used for multi-object performance evaluation. It comprises of two components — one considering localization errors while the other considers the mismatch in cardinality. For our case, an important consideration is the relative performance of the C-SMC-PHD filter with respect to the non-cooperative variant. As no significant improvements in cardinality is expected through cooperation, we only focus on the comparison of localization errors. For computing the OSPA error, the sensitivity of the metric in penalizing the estimated position $p = 1$ was chosen.

A comparison of the cooperative and non-cooperative filter's OSPA localization errors averaged over 30 runs is provided in Figure 9.5 and Figure 9.6. The right diagram of Figure 9.6 furthermore provides a comparison of the average difference in OSPA error between the cooperative and non-cooperative variant of the multiple vehicle tracking filter. To ensure a fair comparison, only targets in the FOV of the vehicle $\mathcal{E}$ are considered when computing the OSPA error, since no targets outside of this area can be observed when the non-cooperative filter is used.

No statistically significant difference is observed in this comparison. We believe the absence of significant improvements could be attributed to the localization uncertainty of tracking vehicles, which correspondingly results in an uncertain transformation of the tracked vehicles from the vehicle's local frame to global coordinates. Furthermore, the fact that the two vehicles observe targets from a similar position could also contribute to this result. However, it is important to note that cooperation with a vehicle that tracks poorly due to localization error or other factors does not degrade the sensing vehicle's tracking accuracy.

A comparison of the cooperative and non-cooperative filter's tracking errors per tracked vehicle is shown in Table 9.1, as observed from both tracking vehicles. Values here are averaged over all state estimates of a simulation run and over all simulation runs. Once again, only targets in the internal FOV of the corresponding tracking vehicle are considered. The C-SMC-PHD filter showed on average a slightly better accuracy than its non-cooperative counterpart.

It is furthermore worth noting that the significant tracking errors even at short distances (for vehicles $\mathcal{NC}_3$ and $\mathcal{NC}_4$ especially) could be partially attributed to the preprocessing of raw LIDAR data and the limited angular resolution of the LIDAR sensor, since a point cloud relating to a vehicle can vary significantly with vehicle's pose in the sensing FOV.

An important aspect of our approach is also to provide a reliable estimate of external tracked

Table 9.1 – A comparison of tracking accuracy of common vehicles between cooperative and non-cooperative tracking.

Tracked vehicles	Tracking accuracy as perceived from vehicle (in meters):			
	$\mathcal{E}$		$\mathcal{C}$	
	Coop	No-coop	Coop	No-coop
$\mathcal{NC}_1$	2.001	2.133	1.854	1.842
$\mathcal{NC}_2$	1.920	2.060	1.776	1.832
$\mathcal{NC}_3$	0.994	1.042	1.467	1.499
$\mathcal{NC}_4$	1.377	1.392	1.048	1.116

Table 9.2 – A comparison of tracking accuracy of external vehicles (outside the direct FOV) between direct and cooperative tracking. The entries in bold correspond to the external vehicle of $\mathcal{E}/\mathcal{C}$.

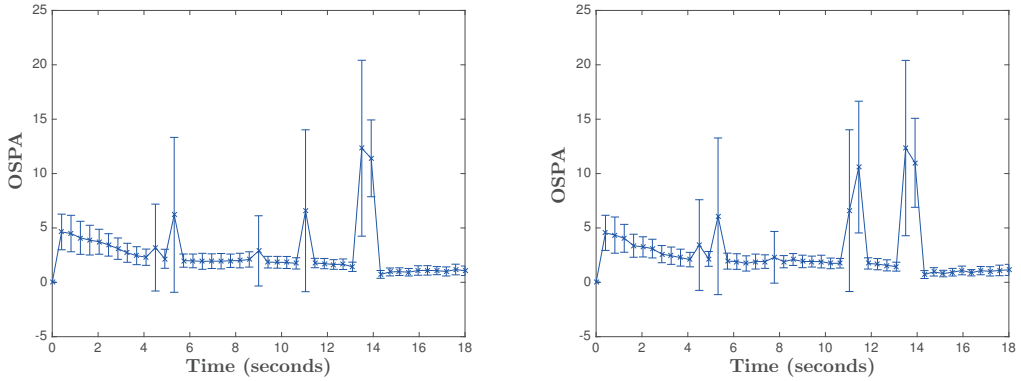
Tracked vehicles	Tracking accuracy as perceived from vehicle:	
	$\mathcal{E}$	$\mathcal{C}$
$\mathcal{NC}_3$	**2.374**	1.499
$\mathcal{NC}_4$	1.392	**3.123**

vehicles (vehicles not in the tracking vehicle's FOV), of which we have no additionally information except from that of cooperating vehicles. In the case of our scenario, $\mathcal{E}$ and $\mathcal{C}$ track $\mathcal{NC}_3$ and $\mathcal{NC}_4$ exclusively for a certain amount of time and provide this information to each other as described in Section 9.2.2. The tracking accuracy of external vehicles for $\mathcal{E}$ and $\mathcal{C}$ is shown in Table 9.2. Although it is observed that external vehicles are tracked worse compared to the tracking done with own measurements, this is only to be expected owing to the reduced frequency of cooperation (once every three sensing steps, as noted in Section 9.3.2) and localization uncertainty of the two tracking vehicles.

When viewed in relation to the range and FOV of LIDARs, the disparity observed is not very significant considering that information on external vehicles could be easily more than tens of meters away from the tracking vehicle itself.

Summary

This chapter presented an approach for cooperative multi-object tracking based on an SMC-PHD filter. It represents an alternative to the GM approach. While the SMC-PHD filter does not require a Gaussian assumption for process and sensor models, our cooperative fusion does as it is based on the KDE method for continuous approximation of i.i.d. distributions. We proposed to factor in the localization uncertainty in the likelihood model of the SMC-PHD filter. Localization is implemented using an UKF which fuses GNSS and compass data. Experimental evaluation is performed in Webots in a slow-speed scenario involving two cooperative and four non-cooperative cars.



(a) Without cooperation **(b)** With cooperation

Figure 9.5 – A comparison of OSPA localization errors between non-cooperative tracking (left column) and the C-SMC-PHD filter (right column) as viewed from $\mathcal{E}$. The corresponding standard deviation is also plotted as error bars.

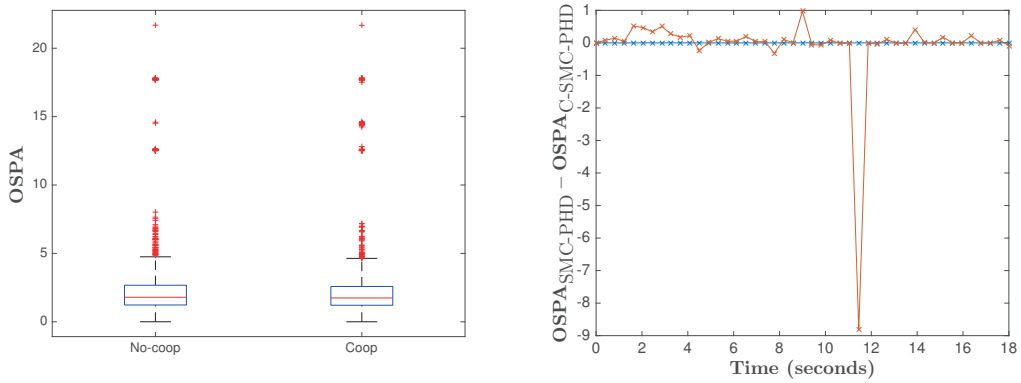


Figure 9.6 – A comparison of the statistics (left) of the total OSPA localization errors between non-cooperative tracking and the C-SMC-PHD filter as well as difference in OSPA error (right) between non-cooperative and the C-SMC-PHD filter as observed from $\mathcal{E}$.

10 An Overtaking Assistance System Based on Cooperative Perception

In this chapter, we showcase a possible application of cooperative perception algorithms. In 2013, there was a total of 81,000 accidents while overtaking in the USA [102]. Overtaking is one of the most dangerous maneuvers on two-way roads for a couple of reasons. Firstly, the driver's FOV gets partially occluded by the preceding vehicle thus impairing visibility conditions. Secondly, humans tend to make wrong judgments and underestimate distances and relative speeds of the oncoming traffic. These factors can lead to an accident, which often ends up with fatalities. We believe that cooperation between the vehicle that is performing the overtaking maneuver and the vehicle being overtaken can mitigate these effects and contribute to traffic safety.

It is well known that distances to remote objects and their relative velocities can be estimated more accurately using sensors than by bare eyes. Vehicles' built-in intelligence can therefore be a useful aid to human drivers in these situations. However, there exist situations in which individual intelligent vehicles cannot perform very well: no matter what sensor technology is used, all sensors suffer from limited range or FOV, or from Non-Line-Of-Sight (NLOS) conditions to some extent. Hence, cooperation of vehicles by the means of exchanging the sensing data or track estimates can be advantageous.

In this chapter, we consider a sample scenario depicted in Figure 10.1. It introduces three types of cars: the ego ($\mathcal{E}$), the leading ($\mathcal{L}$) and the oncoming ($\mathcal{O}$) cars. The car $\mathcal{E}$ is an intelligent vehicle which contains sensors enabling it to perceive the environment and estimate its state. It processes the information and, once it encounters a slower car ahead, assesses the risk of overtaking and makes an overtaking decision. The car $\mathcal{L}$ is any car that finds itself driving in front of the car $\mathcal{E}$, on the same lane at lower speed. It may be equipped with sensors and may share its information with the car $\mathcal{E}$ through communication. The car $\mathcal{O}$ is any car that drives on the passing lane of the car $\mathcal{E}$, in the opposite direction. It possesses no senors nor communication equipment, and it represents a potential hindrance to the overtaking maneuver.

In this chapter we present an algorithm which assesses the overtaking risk and decides if

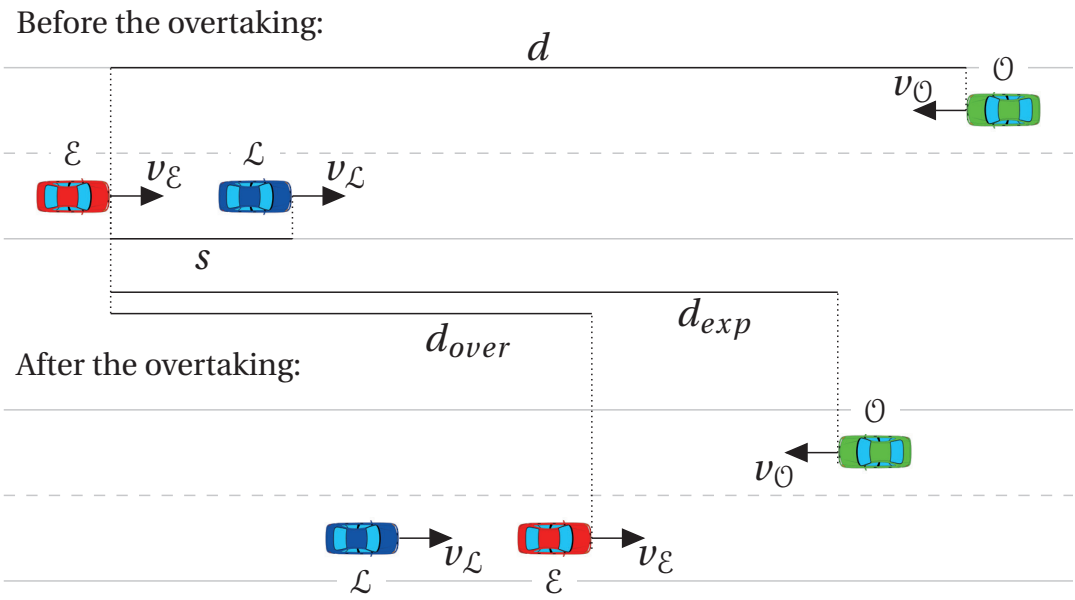


Figure 10.1 – Example of an overtaking scenario.

the car $\mathcal{E}$ can overtake the car $\mathcal{L}$ safely. The algorithm relies on information provided by the C-GM-PHD filter presented in Chapter 8. The cars $\mathcal{E}$ and $\mathcal{L}$ run the tracking algorithm, each using a forward-looking LIDAR. The car $\mathcal{E}$ additionally fuses the information received from the car $\mathcal{L}$, if available. We consider two cases: the case in which the car $\mathcal{L}$ cooperates with the car $\mathcal{E}$ by communicating its information, and the case in which it does not.

10.1 Background

Cooperative overtaking assistance systems are readily studied in the literature. Two main identified approaches are listed below.

- The first approach assumes that each vehicle broadcasts a beacon which includes its position, speed and direction. Frequent broadcasting of such messages allow for detection of oncoming traffic as early as possible (see for example [103]–[105]). A system which predicts when a potential lane-change is going to be performed by the driver of the ego-vehicle, cooperatively exchanges information with vehicles traveling in the opposite direction via the cellular network, and provides the driver with an estimate of the overtaking maneuver risk is presented in [106]. The requirements regarding the communication (e.g., what data should be broadcast, update frequency, range of coverage, boundaries for packet loss and propagation delay) for cooperative overtaking systems in VANETs were studied by van Kooten in his master thesis [107]. All mentioned approaches assume that all vehicles on the road are cooperative. An additional requirement is that they are localized with enough accuracy. No perception sensors were used in these works.

A drawback of these methods is that non-cooperative vehicles (not sending beacons) will not be detected. However, this would especially be the scenario in which intelligent vehicles will have to operate in the early market penetration of V2V technology.

- The second approach relies on real-time transmission of video streams. A video stream captured by a forward-looking camera of the front vehicle is compressed and broadcast to the vehicles driving behind, which then show this stream to the driver on a display. Examples of this approach can be found in [77], [108]–[110]. While such systems allow for detection of different dangerous road situations, such as vehicles not equipped with communication transceivers or animals, they have their drawbacks as well. First, despite using compression algorithms, video-based algorithms still require relatively high communication bandwidth. Second, a human driver needs to look at and analyze the received video stream on top of the directly perceivable surrounding scene, in particular to estimate distance and speed of vehicles traveling in the opposite direction, which may be difficult.

An improved approach could combine the video stream and the V2V beaconing approaches mentioned above (see for instance [111]). A cooperative overtaking assist system using an intelligent road surface is introduced in [112]. Its disadvantage is that it requires investments in the infrastructure.

In our approach, the overtaking assistance is based on the information obtained from on-board sensors, which is fed to the target tracking algorithm, and finally fused with object data received from a cooperative vehicle. Our cooperative perception algorithm is designed to minimize the communication bandwidth requirements — instead of raw data, tracks are communicated.

10.2 Overtaking Decision Algorithm

The behavior of the car $\mathcal{E}$ from Figure 10.1 is defined by a Finite State Machine (FSM) shown in Figure 10.2. The car $\mathcal{E}$ drives freely on the road at its target speed until the tracking module detects a car driving on the same lane at a lower speed (leading car $\mathcal{L}$). The state of the car $\mathcal{E}$ is then changed to *car following*, where the adaptive cruise controller is engaged. As described in the remainder of this section, a risk for overtaking is continuously computed as a function of the presence and position of the car $\mathcal{O}$, and the decision whether to start (or eventually abort) the overtaking is based on a threshold.

10.2.1 State Description

Four states are defined in the FSM.

Free driving The car $\mathcal{E}$ drives freely on the road at target speed v_0 while keeping centered in

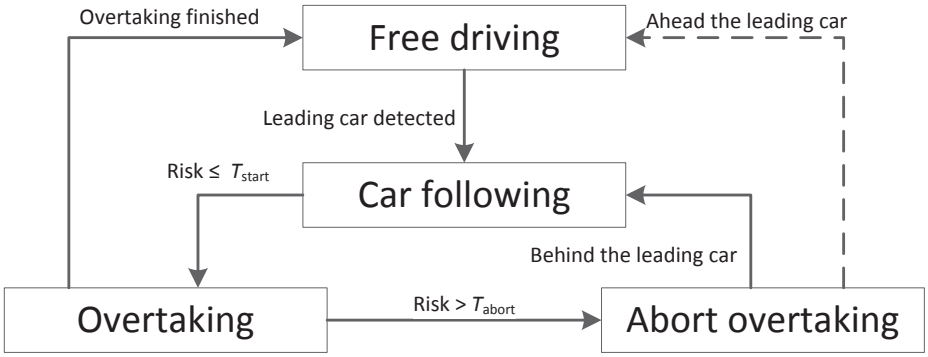


Figure 10.2 – An FSM defining the behavior of the ego car.

its lane.

Car following The car $\mathcal{E}$ follows the car $\mathcal{L}$ (which is driving on the same lane), maintaining a desired distance and time headway.

Overtaking The car $\mathcal{E}$ has determined that it is safe to overtake the car $\mathcal{L}$ (the risk that there is an oncoming car $\mathcal{O}$ on that lane is low enough). It changes lane and performs overtaking at its desired speed v_0.

Abort overtaking The risk that the car $\mathcal{O}$ occupies the overtaking lane has become significant, thus the car $\mathcal{E}$ aborts the overtaking maneuver.

Controllers used in each of the states are explained in Sections 10.2.3 and 10.2.4.

10.2.2 Decision Algorithm (FSM State Transitions)

The initial FSM state is *free driving*. Once the car $\mathcal{L}$ is detected, the state is changed to *car following*.

In the *car following state*, the overtaking risk is constantly computed. The risk is defined between 0 and 1, and it depends on two factors: the probability that there is a car $\mathcal{O}$ on the passing lane, and the comparison between the distance to overtake and the distance traveled by the car $\mathcal{O}$ over the same time. The two distances are estimated using the time to overtake and are computed by taking into account the position and the velocity of the ego, leading and oncoming vehicle. If the risk is lower or equal to the defined threshold T_{start} for at least five consecutive simulation steps, the overtaking is initiated.

The risk is constantly computed in the *overtaking* state as well. If the risk exceeds the T_{abort} thresholds for at least two consecutive simulation steps, the state is changed to *abort overtaking*. Otherwise, if the overtaking is successfully finished (the car $\mathcal{E}$ gets ahead of the car $\mathcal{L}$ plus the safety distance d_{safe}), the state is changed back to *free driving*.

In the *abort overtaking* state, the positions of cars $\mathcal{E}$ and $\mathcal{L}$ are compared. If the car $\mathcal{E}$ is ahead of the car $\mathcal{L}$, it accelerates and changes lane back to its driving lane without taking into consideration the safety distance. The state is then changed back to *free driving*. Otherwise, it brakes and pulls behind the car $\mathcal{L}$, and the state is consequently changed to *car following*.

The subsections that follow provide details on how the cars $\mathcal{L}$ and $\mathcal{O}$ are detected among all tracked vehicles and how the overtaking risk is computed.

Detection of the Leading and Oncoming Vehicle

The C-GM-PHD filter feeds a list of tracked cars to the controller. In order to detect in which lane the tracked car is driving, a temporary coordinate system is placed at the lane in which the car $\mathcal{E}$ is driving (see Figure 10.3). The position of the tracked car is expressed in that coordinate system, by taking into account the lateral position of the car $\mathcal{E}$ in its lane (cf. Section 10.2.3). Then, a number of samples are taken at random from the normal distribution centered in the position of the tracked car, with the covariance that is provided by the tracker. This set of samples reflect the probability distribution of the position of the tracked vehicle.

The number of samples falling on one lane over the total number of samples defines the probability that the center of the car is in that lane. However, when considering to overtake, it is important to assess whether *any part* of the car is in the lane. Thus, the probability $P(i,l)$ of a tracked car i to be on lane l is computed by taking into account all the particles that fall in one lane extended by half of the vehicle width on each of the sides. It is important to note that lane occupancy probabilities defined in this way do not sum up to one (i.e., they do not represent a probability mass function).

Finally, in order to determine the direction of travel of the tracked car, we use its velocity, as well as the car $\mathcal{E}$ velocity.

Overtaking Risk Computation

We define the risk to overtake a vehicle given an oncoming vehicle i as a piecewise function

$$r_i = \begin{cases} 0 & \text{if } d_{exp,i} - d_{over} > d_{margin} \\ 1 - \frac{d_{exp,i} - d_{over}}{d_{margin}} & \text{if } d_{exp,i} - d_{over} > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{10.1}$$

where d_{margin} is a margin distance, $d_{exp,i}$ is the expected distance between the ego and the oncoming car i after completed overtaking, and d_{over} is the distance to be traveled by the ego vehicle until the overtaking completion. The expected distance is easily computed from the tracked distance to the oncoming vehicle d_i and its tracked velocity $v_{\mathcal{O},i}$, as well as the time

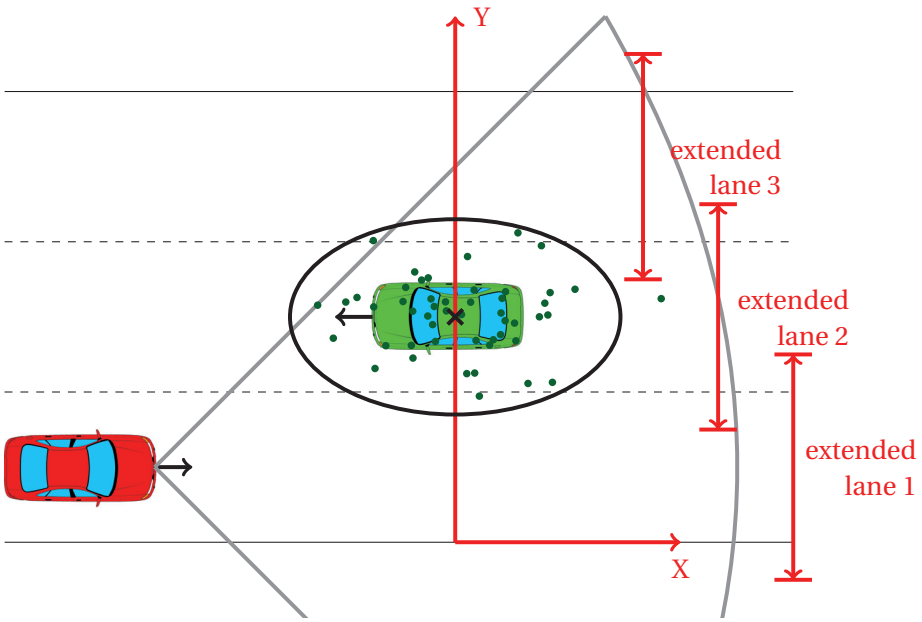


Figure 10.3 – Lane occupancy probability. The center and covariance of the tracked car is shown in black. For illustration purposes, there are 50 sample points (in green) estimating the position of the car center point. All 50 samples are contained within the *extended* lane 2, 6 samples are falling on *extended* lane 1, and 7 points on *extended* lane 3. Therefore, the respective lane occupancy probabilities are $\frac{50}{50} = 1.0$, $\frac{6}{50} = 0.12$, and $\frac{7}{50} = 0.14$. The leading car is not shown for clarity reasons.

that the car $\mathcal{E}$ needs to complete the overtake:

$$d_{exp,i} = d_i - v_{\mathcal{O},i} \cdot t_{over}. \tag{10.2}$$

The overtaking distance is computed as

$$d_{over} = \frac{1}{2} a_{\mathcal{E}} \cdot t_{over}^2 + v_{\mathcal{E}} \cdot t_{over}. \tag{10.3}$$

The time to overtake t_{over} can be computed from the quadratic equation for the overtaking length L in the frame of the car $\mathcal{L}$:

$$L = \frac{1}{2} a_{\mathcal{E}} \cdot t_{over}^2 + \Delta v \cdot t_{over}, \tag{10.4}$$

where $\Delta v = v_{\mathcal{E}} - (v_{\mathcal{L}} + \sigma_{v_{\mathcal{L}}})$ considers the standard deviation of the leaders velocity, and $L = d + l_{\mathcal{L}} + d_{safe}$ is composed of the distance between cars $\mathcal{L}$ and $\mathcal{E}$, length of the car $\mathcal{L}$, and safety distance that is required between the car $\mathcal{E}$ and the $\mathcal{L}$ before the car $\mathcal{E}$ can return back on its lane after overtaking.

Finally, the overtaking risk is defined considering all N_{cars} cars detected by the tracker and

probability of them occupying the overtaking lane $l_{\mathcal{E}} + 1$:

$$R = \max_{1 \leq i \leq N_{\text{cars}}} (P(i, l_{\mathcal{E}} + 1) \cdot r_i). \tag{10.5}$$

The overtaking decision is solely dependent on the risk. We define a threshold T_{start} to start overtaking if the risk is lower, as well as T_{abort} to abort the overtaking if the risk is higher than the threshold.

10.2.3 Lateral Controller

To enable cars to drive on a particular lane (required in the FSM states *free driving* and *car following*), as well as perform lane changes (states *overtaking* and *abort overtaking*), we use pre-defined trajectories. Each lane is composed of waypoints, and a vehicle implements a Proportional-Integral (PI) controller to compute the steering angle based on the coordinates of itself and the next waypoint. Changing lane is achieved by simply swapping the used trajectory. While this approach is a simplification of a potentially more realistic lane following and changing method, it enables us to leverage a perfect positioning information of the car in the lateral controller computations. The lateral controller is the same for each state of the FSM.

10.2.4 Longitudinal Controllers

Different FSM states use different controllers, as described below.

In the *free driving* and *overtaking* states, the car $\mathcal{E}$ is driven at maximum acceleration a until the desired speed v_0 is reached, and with the constant speed afterwards.

In the *abort overtaking* state, we brake at half of the maximum deceleration if the car $\mathcal{E}$ is positioned behind the car $\mathcal{L}$ (overtaking is aborted and the car $\mathcal{E}$ pulls behind the car $\mathcal{L}$). Otherwise, the car $\mathcal{E}$ accelerates at maximum acceleration and finishes the overtake ahead of the car $\mathcal{L}$ (though at high risk).

The controller employed in the *car following* state is based on the ACC model. The ACC provides an acceleration for a vehicle following another vehicle in the lane, taking into account the desired velocity and time gap. It was introduced by Kesting et al. in [113] and it represents an improvement to the Intelligent Driver Model (IDM) [114], which is known to brake too hard in some dense traffic situations, for example due to lane changes of other cars. The acceleration of the car given by the ACC model a_{ACC} can be computed as

$$a_{ACC} = \begin{cases} a_{IDM} & \text{if } a_{IDM} \geq a_{CAH} \\ (1-c)\,a_{IDM} + & \\ c[a_{CAH} + b\tanh(\frac{a_{IDM} - a_{CAH}}{b})] & \text{otherwise,} \end{cases} \tag{10.6}$$

Table 10.1 – Car following model parameters. For explanation and discussion regarding the values, see [113].

Parameter	Value
Desired speed v_0	90 km/h
Free acceleration exponent δ	100
Desired time gap T	0.1 s
Jam distance s_0	5.0 m
Maximum acceleration a	2.7 m/s^2
Desired deceleration b	6.0 m/s^2
Coolness factor c	0.99

where b and c are the deceleration and coolness parameters as defined in Table 10.1. a_{CAH} is a Constant-Acceleration Heuristic (CAH) which assumes that the velocity of the leading vehicle will not abruptly change and is defined by

$$a_{CAH} = \begin{cases} \frac{v_{\mathcal{E}}^2 \tilde{a}_{\mathcal{L}}}{v_{\mathcal{L}}^2 - 2s\tilde{a}_{\mathcal{L}}} & \text{if } v_{\mathcal{L}}(v_{\mathcal{E}} - v_{\mathcal{L}}) \leq -2s\tilde{a}_{\mathcal{L}} \\ \tilde{a}_{\mathcal{L}} - \frac{(v_{\mathcal{E}} - v_{\mathcal{L}})^2 \Theta(v_{\mathcal{E}} - v_{\mathcal{L}})}{2s} & \text{otherwise,} \end{cases} \tag{10.7}$$

where Θ is the Heaviside step function, s is the distance to the car $\mathcal{L}$, $v_{\mathcal{E}}$ and $v_{\mathcal{L}}$ the velocities of cars $\mathcal{E}$ and $\mathcal{L}$ (cf. Figure 10.1), $a_{\mathcal{L}}$ the maximum acceleration of the car $\mathcal{L}$, and $\tilde{a}_{\mathcal{L}} = \min(a_{\mathcal{L}}, a_{\mathcal{E}})$ the effective maximum acceleration. The IDM acceleration function is given by

$$a_{IDM} = a\left[1 - \left(\frac{v_{\mathcal{E}}}{v_0}\right)^{\delta} - \left(\frac{s^*(v_{\mathcal{E}}, \Delta v)}{s}\right)^2\right], \tag{10.8}$$

$$s^* = s_0 + v_{\mathcal{E}} T + \frac{v_{\mathcal{E}} \Delta v}{2\sqrt{ab}}, \tag{10.9}$$

where the approaching rate is $\Delta v = v_{\mathcal{E}} - v_{\mathcal{L}}$. Other parameters and their respective values used for the $\mathcal{E}$ car are summarized in Table 10.1.

10.3 Experimental Evaluation

A systematic experimental evaluation has been carried out in Webots. This section describes the experimental setup and presents the obtained results.

10.3.1 Experimental Setup

Three Citroën C-ZERO cars have been placed on an infinitely long, straight road with four lanes. The car $\mathcal{E}$ is placed behind the car $\mathcal{L}$, both facing one direction, whereas the car $\mathcal{O}$ faces the opposite direction and finds itself on the passing lane of the car $\mathcal{E}$. The longitudinal positions of the three cars are random in each experimental run, but their configuration remains as explained. Our algorithm could easily be adapted for curved roads by computing the distance

Table 10.2 – Characteristics of car sensors and communication devices.

Parameter	Value
LIDAR range	140 m
LIDAR FOV	110°
GNSS noise σ_x, σ_y	0.1 m
GNSS noise σ_v	0.01 m/s
Compass noise $(\sigma_\theta, \sigma_\omega)$	$(0.1°, 0.1\ °/s)$
Communication radius	100 m
Communication delay	80 ms

in the road (curvilinear) coordinate system; the added value of a remote sensor (through communication) could be even more evident in such scenarios due to inherently reduced FOV.

The cars $\mathcal{E}$ and $\mathcal{L}$ are traveling with target speeds v_0 of 90 km/h and 50 km/h, respectively. They are equipped with a forward-facing Ibeo LUX LIDAR, generic GNSS and compass sensors achieving RTK performance, and a communication transceiver.

Communication is achieved by simple messages implemented in the simulator environment using constant delay and message loss. The characteristics of integrated devices are listed in Table 10.2. The car $\mathcal{O}$ travels at target speed of 30 km/h and is not equipped neither with sensing nor communication devices. Attempting to perform an overtaking maneuver at higher traveling speeds would require using sensors with longer range.

The vehicle model implemented in Webots uses the throttle, brake, and steering angle as inputs. The input steering angle (provided by the lateral controller) is transferred to the wheels using a Webots built-in Proportional-Integral-Differential (PID) controller. We use the Citroën C-ZERO motor model (explained in Chapter 5) to map the required acceleration (provided by the longitudinal controller) to the throttle input of the Webots `car` library, and we assume a linear brake response in the case of deceleration. The simulation step is 80 ms, equivalent to the sampling frequency of the LIDAR.

10.3.2 Tracking Parameters

In the C-GM-PHD filter, we empirically determine the sensor standard deviation to be

$$\sigma_z = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_\theta \end{bmatrix}^\top = \begin{bmatrix} 1\ \text{m} & 1\ \text{m} & 45° \end{bmatrix}^\top \tag{10.10}$$

(this includes the point cloud preprocessing noise). To make our filter conservative, the ego state estimation noise needed for the Approximate Transformation is intentionally not set exactly to the values used in the simulator (cf. Table 10.2). Instead, it equals to

$$\begin{bmatrix} \sigma_x & \sigma_y & \sigma_\theta & \sigma_v & \sigma_\omega \end{bmatrix}^\top = \begin{bmatrix} 1\ \text{m} & 1\text{m} & 0.5° & 1\ \text{m/s} & 0.5\ °/\text{s} \end{bmatrix}^\top. \tag{10.11}$$

The clutter model is assumed to be Poisson with mean of 10 clutter measurements per sensor surveillance area. The probability of detection and survival are respectively set to 0.98 and 0.99. The occlusion model, as well as the values of other parameters intrinsic to the C-GM-PHD filter are the same as used in Chapter 8.

10.3.3 Experimental Results

The experiments are conducted using six different parameter settings. While keeping the value of T_{start} at some small, positive value (we chose 0.01), we vary $T_{\text{abort}} \in \{0.2, 0.5, 0.8\}$. For each value of T_{abort}, we launch the experiment with and without cooperative fusion. Without fusion, the car $\mathcal{E}$ tracks targets using only its LIDAR, thus having shorter range and more occluded FOV. For each of the six settings we perform 500 simulation runs with different random initial positions to obtain statistically significant results.

An overtaking attempt is defined as an event in which the car $\mathcal{E}$ starts changing the lane. It can be finished by a successful overtake, an abort or a crash.

Figure 10.4a shows the duration of overtaking maneuver, measured from the moment the first overtaking attempt is made (excluding experiments that resulted in crashes). Long durations represent overtaking maneuvers whose first attempts have been aborted, as in these cases the car $\mathcal{E}$ had to wait for the risk to diminish before starting and successfully completing the second overtaking attempt. It can be seen that the duration decreases with increased risk threshold T_{abort}. This is due to overtakes that are less likely to be aborted, for the price of accepting more risk. We can also see that experiments which use fusion outperform the ones without fusion algorithm: since, when using fusion, the car $\mathcal{E}$ suffers less from occlusions and can see further away, it aborts the overtaking attempt less frequently and therefore saves time.

The percentage of aborted overtakes is given in Table 10.3. Aborts are split in two categories, the ones in which the car $\mathcal{E}$ brakes and pulls behind the car $\mathcal{L}$, and the ones in which the car $\mathcal{E}$ keeps accelerating and successfully finishes the risky maneuver. There are significantly less aborts of the second type in the case of fusion than in the case of no fusion. Their number also decreases with the increasing T_{abort}, as the car $\mathcal{E}$ allows for higher risk before aborting. Due to larger sensor range in the fusion case, there are slightly more aborts of the first type than in cases without fusion. These facts lead us to the conclusion that cooperative fusion might contribute to road safety, as it reduces the number of occasions in which the overtaking car enters risky situations. Figure 10.4b supports this conclusion, by showing the actual risk at which the overtaking maneuver was aborted. It is always significantly lower in the case of fusion, independent from the chosen value of T_{abort}.

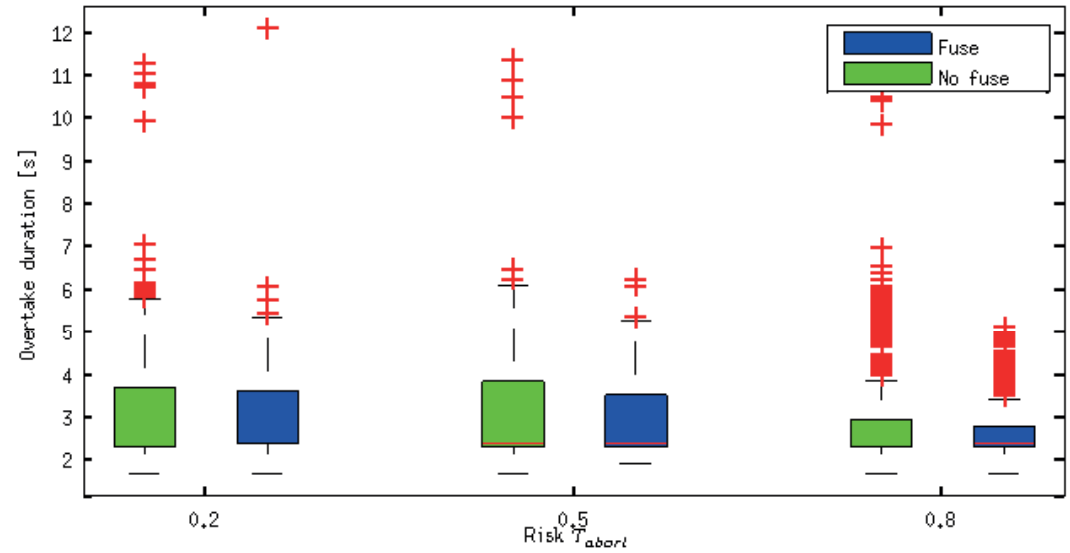
Overtakes resulting in a crash between the car $\mathcal{E}$ and the car $\mathcal{O}$ are listed in Table 10.3 as well. Data show that cooperative fusion successfully reduces the number of crashes with the oncoming car, which is indeed its intended purpose. As expected, the highest number of crashes are recorded for $T_{\text{abort}} = 0.8$.

Table 10.3 – Percentage of aborted and crashed overtakes.

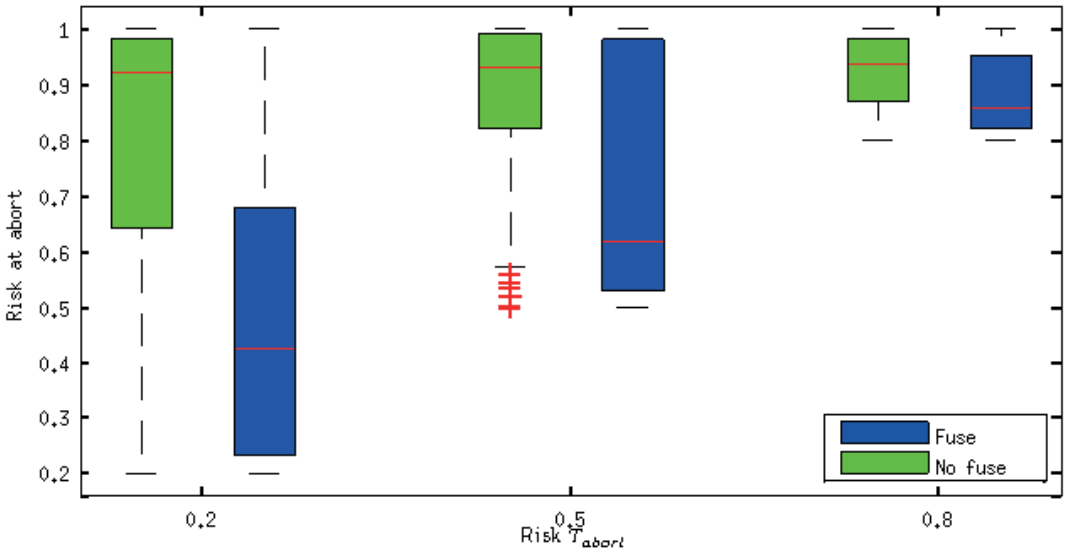
	Risk					
	0.2		0.5		0.8	
	No fuse	Fuse	No fuse	Fuse	No fuse	Fuse
Abort behind $\mathcal{L}$ (%)	2.47	3.58	2.24	4.25	2.93	3.18
Abort in front of $\mathcal{L}$ (%)	16.29	0.42	9.39	0.21	7.53	0.21
Crash (%)	1.62	0.00	1.01	0.00	2.65	0.81

Summary

An application of the C-GM-PHD filter for overtaking assistant system was shown in this chapter. We designed an algorithm which estimates the risk of the overtaking maneuver from the perception data. The risk is associated with the probability of an oncoming vehicle driving on the passing lane (the lane that is used by the ego car for overtaking), as well as with the kinematic conditions such as driving speeds, the distance required to overtake and the distance to the oncoming car. Based on the risk, a safe decision whether to overtake or not is taken. The ego car implements a lateral controller for lane following, and an ACC controller for safe car following. The approach is validated in Webots; the added value of cooperative perception in the overtaking scenario was reflected through smoother and safer vehicle operation.



(a)



(b)

Figure 10.4 – Boxplots showing for different values of T_{abort} (a) overtake duration from the moment of the first overtaking attempt, (b) overtaking risk at abort (note: the risk is always higher or equal than the abort threshold T_{abort}). Each box aggregates 500 runs minus the number of crashes and represents the upper and lower quartiles, the red line in the box marks the median, the bars extend to the most extreme data points not considered outliers, and the red crosses show outliers.

11 Conclusion

COOPERATIVE perception brings a number of benefits to intelligent vehicles. Its most obvious benefit is increased range and FOV, as compared to the range and FOV obtainable by sensors installed on an individual vehicle. By using communication as a remote sensor, a vehicle can detect and track objects which are not on its LOS, as radio waves are less likely to be absorbed by obstacles than visible and near-infrared light. Furthermore, by using multiple independent sources of information offered by cooperative vehicles, detection uncertainty can be decreased (and the accuracy increased).

In this part we presented two algorithms for cooperative perception. In the core of both algorithms is a PHD filter for tracking multiple objects. The first algorithm uses a GM approximation of the PHD, while the second one uses an SMC approximation. The PHD intensities are communicated between cooperative vehicles and fused with their local PHD intensities in a distributed way. Moreover, we built an overtaking recommendation system on top of our cooperative perception algorithm. It demonstrates one possible application for intelligent vehicles in which cooperative perception in beneficial.

We performed the evaluation of the algorithms in the high-fidelity simulator Webots. We designed realistic scenarios in which the sensing cars were both stationary and moving. The filters in Chapter 8 and Chapter 9 are implemented in Matlab and Python, respectively, and are run offline on collected datasets. Chapter 10 features an online implementation in C++, which is nested in the control loop and runs in Webots.

11.1 Discussion

Our proposed fusion methods are fundamentally different from those presented in [81]–[83], in the sense that in addition to consensus fusion, they also perform complementary fusion (i.e., they provide extended coverage when different sensors cover different regions).

In the C-GM-PHD filter, we achieve the complementary fusion by modifying the GCI algorithm in a way that the CI fusion is only applied to Gaussian components which are sufficiently

close to each other. The components received from a cooperative vehicle, that do not relate to any object tracked by the ego vehicle, are accepted as such and are only predicted (without performing measurement update) until the next cooperative intensity is received (or their weights become too small). Our approach shows promising results: most of the time, a C-GM-PHD filter yields a more accurate estimate than its non-collaborative version, with the added value that the ego vehicle is no longer limited to tracking targets that are in its FOV. Perhaps the largest limitation of this approach is the constant fusion weight, which gives the equal importance to the ego and the cooperative intensity. As seen during evaluation, the equal weight can occasionally cause wrong cardinality estimates.

The C-SMC-PHD filter uses two sets of particles, one relating to the objects located in the FOV of the ego vehicle, and the other relating to particles outside of its FOV. A simple heuristic is used to decide whether to consider received particles as external to the tracking vehicle's FOV or as common to both FOVs. In this implementation, an optimization method using the RD as a cost function is used. Again, the C-SMC-PHD filter demonstrates performance benefits over its non-cooperative version. One of the limitations of the SMC approach is the large number of particles that needs to be communicated to cooperating vehicles. Exploring methods that would reduce the number of particles to be transmitted could be necessary if such concept were to be scalable for use in larger vehicle networks.

While we have shown that our methods accomplish the goal of providing extended visibility' for tracking vehicles, further work remains to be done in order for significant tracking accuracy improvements in the overlapping FOV to be observed. We attribute this lack of accuracy improvement to the localization uncertainty of the sensing vehicles which needs to be propagated to the sensor frame and added to the observation uncertainty, as well as the conservative nature of the CI algorithm. CI is often used in sensor networks, because it prevents double-counting of information. An alternative approach would be to perform careful bookkeeping to track the origins of measurements and prevent the use of any of the measurements more than once (as for example in [115]).

Part IV
Algorithmic Enhancements and Validation Using Real Vehicles

12 Introduction

In the previous part of this thesis, we have seen two algorithms for cooperative tracking. Their mathematical formulation has been discussed and their performance was evaluated using high-fidelity simulation. The main goal of this thesis part is to validate our work on real vehicles. In all real-vehicle experiments, we use the C-GM-PHD implementation (introduced in Chapter 8). Additionally, this part brings a number of enhancements to the C-GM-PHD algorithm, with the common goal to improve its robustness and performance, in particular when deployed on real vehicles.

First, in Chapter 13, we provide a detailed description of our software framework. We explain the design of our software architecture, we provide insights in our integration effort and finally we explain and evaluate the V2X communication channel we have been using. Our software framework choices aim at making the transition from simulation to reality as smooth as possible. We have chosen the Robot Operating System (ROS) as our middleware because it is compatible both with Webots and with the real vehicles. Software based on ROS prototyped in simulation can easily transported to the real vehicle framework. As previously shown in Chapter 5, we have developed simulated sensor models and have calibrated them using real devices. By implementing new sensors in Webots (LIDARs and Mobileye cameras), we are able to fully reproduce our test vehicles in simulation.

In contrast to simulation, noise and clutter levels of real sensors cannot be controlled. Perception algorithms need to cope with the artifacts produced by different elements in the uncontrolled environment (e.g., reflections from bushes or wire fences). Localization algorithms need to overcome the difficulties arising from the GNSS signal multi-path and proprioceptive sensor noise and biases. For these reasons, we implement a few enhancements of our original cooperative tracking algorithm presented in Part III.

We start off gradually by making an assumption that accurate localization is available on our two test vehicles. We achieve this in reality by making the test vehicles stationary, while manually setting their localization parameters (by visually aligning the contours of the detected vehicle in the point cloud with the communicated location of that vehicle). Chapter 14

addresses one of such scenarios — we will refer to it as *Scenario 1*, in which the two cooperative cars are positioned one behind another, and they track two legacy cars driving in the opposite direction. For the first time, we integrate the Mobileye camera as perception sensor in the tracking algorithm. As there is no ground truth position available for the legacy vehicles, our approach is to manually extract their poses from the LIDAR point cloud (whose distance estimates are significantly more accurate than those of the Mobileye camera). This approach allows us to perform quantitative analysis of our tracking algorithm. As accurate localization is available in simulation, we can afford to reproduce the boundary conditions of this scenario even by moving the vehicles, anticipating through our simulation framework the next logical step in reality (see below).

In order to implement the cooperative perception on moving vehicles, the accuracy of inter-vehicle relative localization must be improved first. We address this issue using the cooperative localization techniques described in Chapter 15.

Chapter 16 puts together the cooperative localization and cooperative perception algorithms, allowing us to move the real vehicles. Moreover, by using both LIDAR sensors and Mobileye cameras in the tracking algorithm, it builds on the approach shown in Chapter 14. Our approach is validated in two real-life scenarios. In the *Scenario 2*, one car is stationary while the other one is moving. This resembles a V2I scenario, where a moving vehicle gets assisted by an infrastructure sensor. In this scenario, two pedestrians are tracked in a cooperative fashion. The *Scenario 3* hosts two moving vehicles, driving behind each other, in a typical V2V configuration. They cooperatively track two legacy oncoming cars.

Recall from Section 4.4 that we use two different real system configurations in our experiments. Table 12.1 outlines which deployment phase is used in which experimental scenario. It also indicates which sensors are used for object tracking.

Table 12.1 – Test scenarios and corresponding real-vehicle deployment phase.

	Scenario		
	1	2	3
Deployment phase	1	2	2
Mobileye cameras	✓	✓	✓
LIDARs	✗	✓	✓

Summary

In this part of the thesis we implement and validate our cooperative perception algorithms on real vehicles. The use of a calibrated, high-fidelity simulation tool and a carefully chosen software framework compatible with both simulation and real deployment makes the transition from simulation to real hardware less bumpy. We introduce a few algorithmic enhancements that make the method more robust and more accurate, especially when deployed on the targeted real hardware. We explain how a Mobileye camera can be used for tracking in the PHD filter framework, and we propose a method for fusion of signals from LIDARs and Mobileye cameras.

13 A Software Framework for Bridging the Simulation-to-Reality Gap

OUR software framework relies on ROS [116]. ROS is a flexible open-source framework for writing robot software. It comprises a set of libraries and tools; we list the ones relevant for our work. The framework is designed to support a smooth transition from simulation to the real cars. On the one hand, it can be used with Webots, our high-fidelity simulation tool; on the other hand, it can be used directly with real sensors (and actuators, if available for control).

At its lowest level, ROS offers a message passing interface that provides inter-process communication and acts as a middleware between the operating system and the user software. It implements a publish/subscribe mechanism, allowing for anonymous and asynchronous communication between distributed nodes of a robotic system. A message passing system requires implementing clear interfaces between different nodes, thus promoting code reuse. In a nutshell, the publish/subscribe mechanism allows a node to subscribe to topics of interest which contain its input data, perform some computation on these data, and publish some data on output topics. A node subscribing to some topic is agnostic with respect of the source of the data. It only matters that the data format satisfies the requested interface.

For example, a Node A can be a GNSS sensor driver. It implements a sensor-specific protocol on one side, retrieves the sensor data, and makes it available to the other ROS nodes by publishing it on a topic. A Node B can be in charge of localization, therefore processing the data provided by the Node A. Another node — Node C — can be handling the navigation task, therefore using the data published by the Node B, and so on.

Another key feature of ROS is recording and playback of messages using the `rosbag` tool. Messages are timestamped and stored in a bag file. Replaying the bag file does not require any changes in the code of the nodes as compared to the case when data is generated in real time. In the previous example, Node B and Node C are agnostic with respect to the source of the GNSS data, and they operate in the same way regardless of the data coming directly from a GNSS sensor or from a bag file.

We further make use of the `tf` library, which keeps track of different coordinate frames and

their relations in a systematic way, and provides functions for converting data between different frames. For visualization we use `rviz`, a three-dimensional tool that can visualize many of the common data types provided in ROS, such as point clouds, camera images, coordinate frames, as well as general purpose markers having generic shapes.

A default ROS controller is available in Webots. It enables publishing data from all standard Webots sensors on a ROS topic. This data can subsequently be used by all other nodes running on the same ROS system (e.g., processing or visualization nodes). In a similar fashion, all actuators subscribe to the relevant topics and services, enabling their control from within the ROS system. Together with the `driver` and `car` libraries (described in Chapter 5), we developed the corresponding ROS controllers. They establish ROS interfaces with the simulated cars, as well as with the simulated LIDARs and Mobileye cameras.

As already mentioned, nodes in ROS are agnostic with respect the source of the data. This enables us to re-use the algorithms (including the complete source code) developed in simulation for the real car deployment. The first step in the simulation-to-reality transition is to replace all nodes publishing data from simulated sensors by the nodes that interface real sensors. The second step is to perform fine-tuning of the algorithms, in order to account for the differences that possibly exist between the simulated and the real data, despite the simulator calibration effort reported in Chapter 5. Finally, for the system to run in real-time, timing and synchronization constraints need to be addressed.

13.1 Software Architecture

In this section we describe our software architecture. It is implemented using C++ and Python in the ROS framework. We start with a simpler architecture used in the first deployment phase, and we build on the top of it in the second deployment phase (see Section 4.4 for a refresher on our deployment phases).

13.1.1 Phase 1

Our initial deployment was carried out on `car0` and `car1`. Four LIDARs were deployed on the `car0`, leaving the `car1` only with a Mobileye camera in terms of perception sensors.

We designed a software framework to be used in the Scenario 1. Although the localization equipment was deployed on the real cars already in Phase 1, we do not use it in the Scenario 1, and therefore we also did not implement the corresponding nodes in this phase.

The entire software pipeline is implemented on on-board computers in the ROS framework using C++. Sensor driver nodes publish data from the sensors to the ROS network. Tracking filter nodes subscribe to sensor data and publish tracks in the form of PHD intensities. A cooperative fusion and tracking node (only present in the ego vehicle) additionally subscribes to the tracks and publishes a fused PHD intensity. Our software architecture is illustrated in

Figure 13.1.

All nodes can be run online (in real-time) during experiments. However, as the emphasis of our work is not on the wireless communication technology, we can afford to log all data using `rosbag` and play them back for processing offline. To avoid time misalignment of vehicle logs, the computer times are synchronized among themselves using a network time synchronization tool before and throughout the experiments.

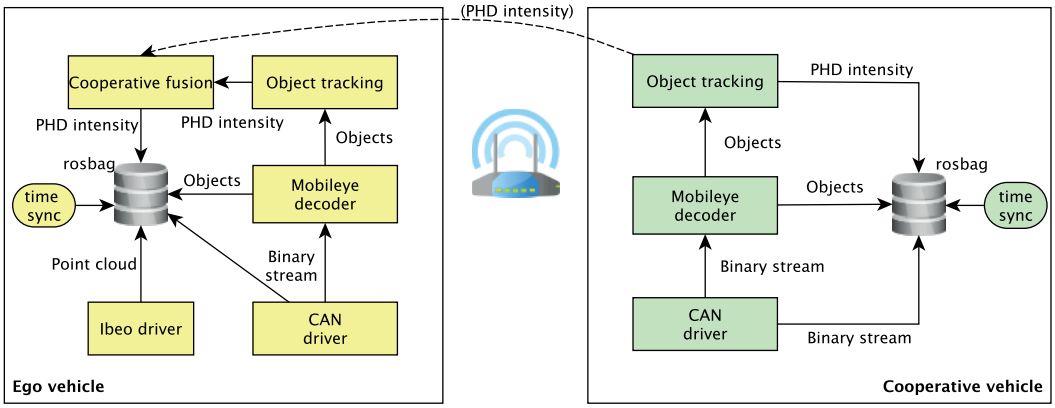


Figure 13.1 – Software architecture in the ROS framework deployed on the two experimental vehicles' computers (Phase 1 of the deployment). The ego vehicle is `car0` (yellow C-ZERO), while the cooperative vehicle is `car1` (green C-ZERO).

In Webots, a software architecture very similar to the one shown in Figure 13.1 is used. Webots publishes data from sensors as ROS topics. These topics correspond in type to the ones published by real sensors. Therefore exactly the same implementation of the cooperative tracking and fusion algorithm can be used. One notable difference is that the whole software (deployed in reality on two cars) runs in simulation on a single computer, which simplifies time synchronization between the two cooperating vehicles.

13.1.2 Phase 2

In the second deployment phase, we re-organized the software architecture in three layers, as shown in Figure 13.2. The bottom layer is the Hardware Abstraction Layer (HAL), which provides an abstraction of sensors to the rest of the ROS nodes. The core middle layer is composed of nodes running processing algorithms. The top layer has two functionalities: communication and visualization. Similarly to Phase 1, the computer times are synchronized among themselves using a network time synchronization tool before and throughout the experiments.

The HAL contains nodes which interface all sensors listed in Section 4.1. For compatibility and re-usability purposes, we maximize the usage of common ROS data types, such as the point cloud, odometry or IMU type.

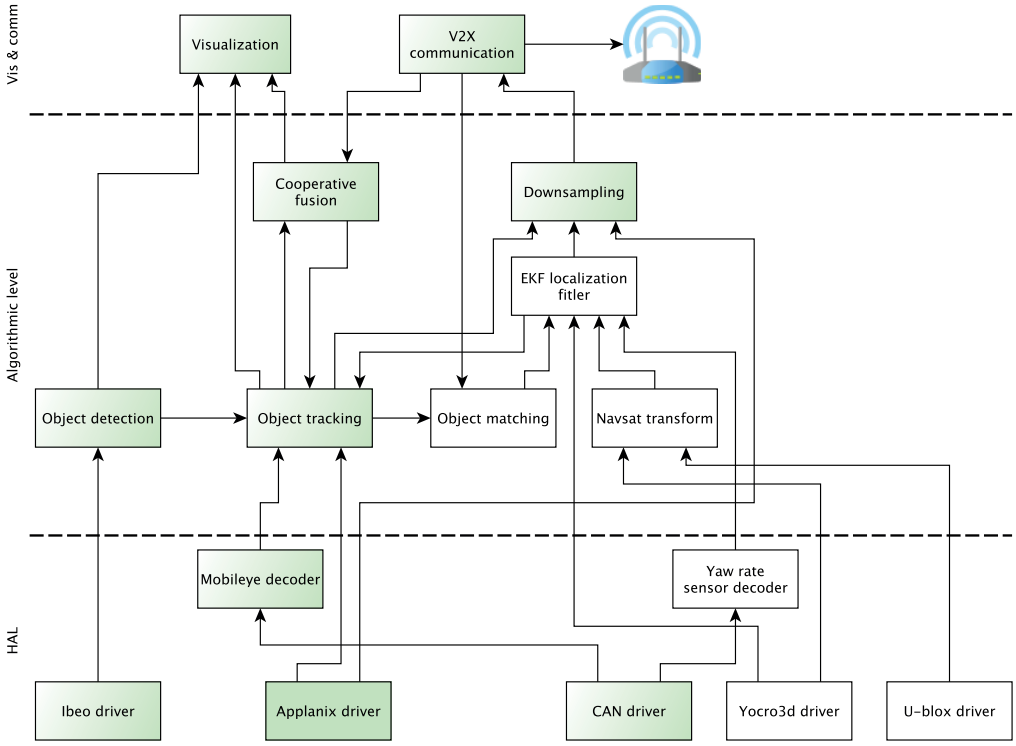
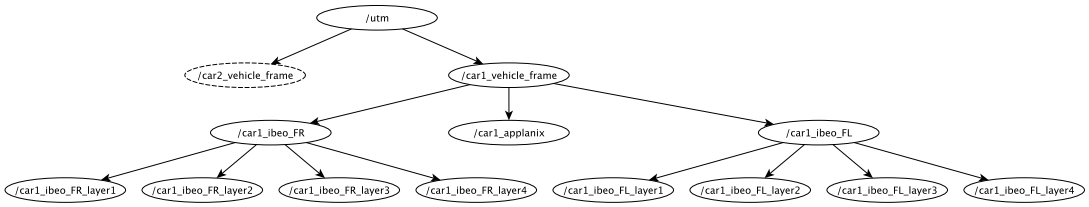


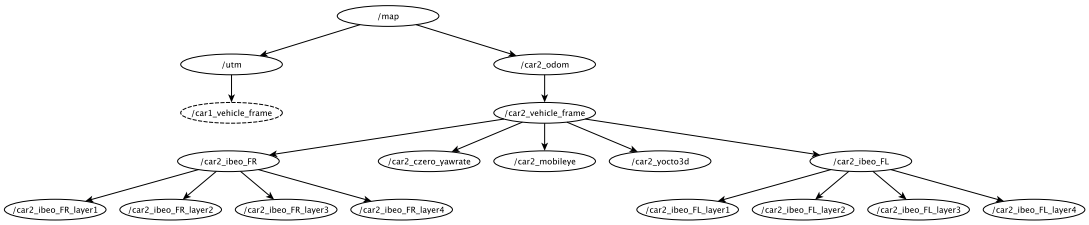
Figure 13.2 – Block diagram of the software architecture for the Phase 2 of the deployment. The pure green node (*Applanix driver*) is deployed on `car1` (green C-ZERO), the white nodes are deployed on `car2` (white C-ZERO), while white-green nodes (e.g., *Ibeo driver*, *Object tracking*) are deployed on both cars. The bottom layer abstracts sensors and provides interface to the middle layer. The middle layer implements algorithms. The top layer deals with visualization and communication. All the nodes are implemented in C++, with the exception of the *Object matching* node, which is implemented in Python.

The middle layer contains our algorithms, clustered in perception and localization groups. The perception pipeline starts with object detection, a preprocessing algorithm that performs clustering of the LIDAR point cloud, extracts various features and outputs detected objects that resemble pedestrians or cars (cf. Section 7.3). Object detections are then fed to the tracking node in the form of measurements. The tracking algorithm based on the GM approximation is implemented, as detailed in Section 8.1. The fusion node also finds its place in the algorithmic layer. It subscribes to the PHD intensity (the first moment of the posterior object density) published by the tracking node on the ego vehicle, as well as the PHD intensity received from the cooperative vehicle (as explained below).

The localization component contains an EKF and two matching algorithms that will be introduced in Chapter 15. The first EKF estimate is the state of the ego vehicle in the odometry frame using dead reckoning (i.e., not using any global positioning sensors). The second EKF estimate is the state in the global frame, by additionally fusing the GPS data. Additionally, there exists a node that converts navsat data (longitude and latitude) to a UTM frame which is a



(a) `car1`



(b) `car2`

Figure 13.3 – A coordinate frame tree for the two cars. An arrow indicates a transform between two given frames. A transform between any two frames in the tree is obtained by compounding (or eventually inverting) the transforms. The transform between the UTM frame and the frame of the second vehicle (shown in dashed line) is published by the *Cooperative fusion* node. `/carX_ibeo_FR` and `/carX_ibeo_FL` are the frames of the front right and front left LIDAR, respectively.

projection of the Earth surface to a Cartesian coordinate system, allowing us to fuse the GPS measurements into the global estimate. Finally, the matching algorithms that aim at improving the accuracy of the relative localization between cooperative vehicles are implemented in a separate node. These algorithms use the localization data of the tracking vehicles, as well as their PHD intensities containing tracked objects, and they output pose updates that are fed in the EKF localization filter.

The visualization component of the top layer is running `rviz`. It subscribes to different topics, such as the LIDAR point cloud or visualization markers from the tracking and fusion nodes. The communication component has two parts: a transmitting and a receiving part. The transmitting part is in charge of down-sampling the tracking and localization data and transmitting it to the fusing vehicle using the User Datagram Protocol (UDP). The receiving part publishes the received data on the ROS network of the ego vehicle, along with relevant coordinate transformations which enable the usage of cooperative vehicle's data in the coordinate frame of the ego vehicle.

The coordinate frames present on each of our vehicles are shown in Figure 13.3a and Figure 13.3b, respectively. The transitions between different frames are achieved using the `tf` tool only where these transitions do not imply uncertainty. In case of uncertain transformations (transforming mean states and covariances), the transitions are computed by the module we implemented.

13.2 Communication

Communication is the essential component of our system. We distinguish between intra-vehicle communication and V2X communication.

13.2.1 Intra-Vehicle Communication

As mentioned above, the ROS system manages communication between multiple nodes. Each ROS system runs one ROS master. It is possible that one ROS master manages a system that runs on multiple computers connected to a network. In that case, all computers communicate with the master. Communication between nodes is still distributed, but the master keeps a list of published topics and manages subscription requests. This kind of communication is used between the two computers deployed on our `car2`, which are connected to each other through a Gigabit Ethernet. We note that a good communication link is required, and that if the communication would be cut between one computer and the ROS master, nodes running on that computer would basically cease to operate until the communication with the ROS master is not re-established.

13.2.2 V2X Communication

Two communication modes can be distinguished in the automotive domain: (i) V2V communication, and (ii) V2I communication. A common label for both modes is V2X communication. In our experimental testbed, we only have vehicles, thus strictly speaking we only use V2V communication. However, the difference between an infrastructure node (equipped with sensors, computers and power supply) and a parked intelligent vehicle lies mainly in the localization accuracy (an infrastructure node is fixed and can be localized accurately by professional surveyors). Thus, in the remainder of this thesis, when a moving car communicates with a manually localized stationary intelligent car, we consider this to be representative for a V2I communication type.

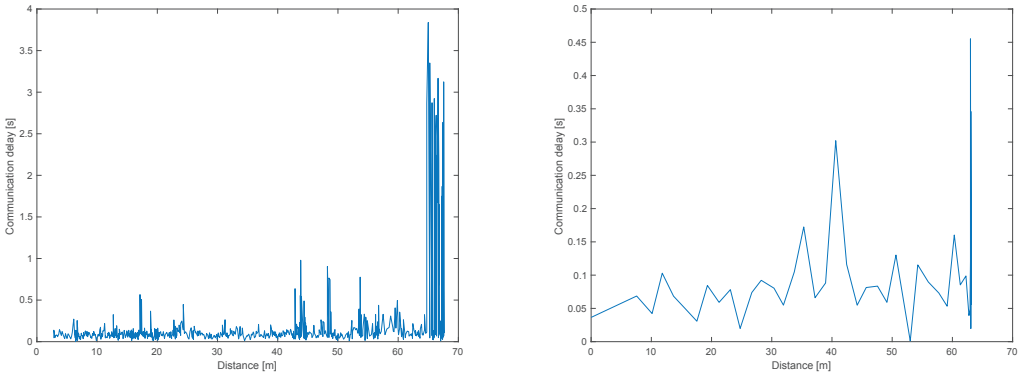
In automotive industry, IEEE 802.11p technology is typically used for Wireless Local Area Network (WLAN) communication. Recently, a first set of LTE standards that uses a radio technology different from IEEE 802.11 WLAN has been announced. There are however drawbacks of the above-mentioned technologies: IEEE 802.11p-based equipment is expensive, and it is often sold separately from the software stack; some standards and definitions are still subject to change. In this thesis, the communication is established over a WLAN at 2.4 GHz based on IEEE 802.11b/g/n specification. The maximum declared physical layer bit rate of our equipment is 300 $^{\text{Mbit}}/_{\text{s}}$. The choice of the widely available communication technology enables us to deploy it quickly and focus on the fusion algorithms. We remind the reader that the focus of our work is not on the communication technology; for us, communication is a medium to transmit data between vehicles. There exist many important aspects of communication (security and privacy among others), which are out of the scope of this thesis.

For V2X communication we use a multi-master communication system developed by the Fraunhofer Institute [117]. It performs discovery of other ROS systems (running ROS master) in the network and enables selecting topics that are subsequently communicated between different ROS systems using the UDP protocol. In this way, a ROS system is run on each car, independently from other cars. Communication is opportunistic: if there is another vehicle in the network (i.e., within communication range), selected data are shared. Multi-master communication provides robustness and requires less bandwidth than running a system with a single ROS master, where all agents need to exchange all the messages with that master. The interested reader can refer to [118] for more details regarding the ROS multi-master communication system.

We evaluate the performance of our V2X communication system by measuring the end-to-end latency. The end-to-end latency accounts for the time elapsed between the moment a message was published to a topic on `car1` and the moment this message was received in a callback on `car2`, hence including both communication and ROS-inherent latency.

In the two experiments below (see also Figure 13.4), we send the same information as we do when running our cooperative fusion algorithm explained in Chapter 8, while varying the distance between the two vehicles. In the V2I communication experiment, `car1` is stationary while `car2` moves away until it gets out from the communication range. In the V2V communication experiment, the two vehicles start at the limit of the communication range, and drive towards each other with the relative speed of around 15 $^{km}/_{h}$. The average end-to-end latency in the first experiment is 0.096 ± 0.081s (one standard deviation, excluding the periods when the connection is lost), while for the second experiment it is 0.040 ± 0.027s (one standard deviation). In our cooperative perception framework, this performance allows us to send reliably information at the frequency of 2 Hz with a negligible impact of the communication latency. The communication performance of tho vehicles driving behind each other in the same direction is not explicitly evaluated; however, our experience indicates that in such settings the communication is more reliable than in the evaluated scenarios, as the relative distance between the vehicles does not vary much.

Our communication is connection-based (a computer in `car1` connects to a wireless router in `car2`). In other words, it is not possible to send or receive UDP/IP packets when the device is not connected to a network, as the Internet Protocol (IP) layer requires the link layer to be connected. The consequence of this protocol choice is that if a connection between the two vehicles is lost, it can take a few seconds before it is re-established (as evident in Figure 13.4a). In this period, no packets will be exchanged between the vehicles, even if they find themselves within the communication range of the system.



(a) V2I scenario

(b) V2V scenario

Figure 13.4 – End-to-end communication delay for packets being sent from `car1` to `car2`. The message frequency is 2 Hz, and the communicated data are those used by the cooperative fusion of the C-GM-PHD filter. (a) `car2` slowly moving away from stationary `car1`; (b) `car1` and `car2` driving towards each other.

Summary

In this chapter we described our distributed and modular software architecture, which has been implemented in ROS. We provided details about contents of the three software layers present in our architecture. This architecture allows us to make a smooth transition from simulation to the real vehicles. Moreover, we reported our approach for V2X communication and evaluated the communication latency as a function of the distance between cars.

14 Cooperative Tracking under Accurate Localization Conditions

THE framework presented in Chapter 8 is general enough to be applied to any sensor and any state model. So far, we have demonstrated its application to car tracking using LIDARs. The approach has been experimentally validated in simulation, and simulated data were processed in Matlab. In this chapter we introduce a new sensing modality for tracking as an alternative to LIDAR sensors, namely a Mobileye camera. Additionally, we make an important improvement to our cooperative fusion algorithm, which consists of weight optimization.

Our analysis is based on two variants of Scenario 1. Scenario 1a is implemented in high-fidelity simulation. As all the software is implemented in ROS as described in the previous chapter, this represents the first step of the transition from simulation to real vehicles. In the Scenario 1b, we replace the simulated vehicles with real ones, and thus perform analysis on real-world data.

Common for both scenario variants is the assumption that all sensing vehicles can benefit of accurate localization. While this is easily achievable in simulation, in the real-world scenario the available equipment is limiting the performance. Therefore, in order to fulfill the required assumption in terms of inter-vehicle localization, we proceed as follows. First, the two vehicles involved in the cooperative tracking operation are stationary. Second, we extract ground truth information from LIDARs as they represent an independent and more accurate source of information than Mobileye cameras.

14.1 Fusion Weight Optimization

Recall from Equation (8.19) that the cooperative fusion algorithm introduced in Section 8.2 requires a parameter $\mathcal{W}$ weighting the two object intensities that are being fused together. In Chapter 8 we simply set $\mathcal{W} = 0.5$, which gave equal importance to the two intensities. Such parametrization choice could lead to problems in cardinality of the fused intensity, as uncertainties of the starting intensities are not taken into account.

In this chapter, we present an extension to the fusion algorithm for the the GM-PHD intensities, which relies on the optimization of the fusion weight $\mathcal{W}$. Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be the Gaussian mixtures in the form of

$$f_{(\cdot)}(\mathbf{x}) = \sum_{i=1}^{J_{(\cdot)}} w_{(\cdot)}^{(i)} \mathcal{N}(\mathbf{x}; \mu_{(\cdot)}^{(i)}, \sigma_{(\cdot)}^{(i)}) \tag{14.1}$$

containing selected Gaussian components from v_1 and v_2 which participate in fusion, and $f_{\mathcal{W}}$ the GM that is created by fusing f_1 and f_2 . Further, suppose that $J(\mathcal{W})$ is a cost function. Then the goal is to choose $\mathcal{W}^*$ such that

$$\mathcal{W}^* = \arg \min_{\mathcal{W} \in [0,1]} J(\mathcal{W}). \tag{14.2}$$

The cost function can be defined as

$$J(\mathcal{W}) = \left(D(f_{\mathcal{W}} || f_1) - D(f_{\mathcal{W}} || f_2)\right)^2. \tag{14.3}$$

The Kullback-Leibler and Renyi divergence have been used as distance metrics in the literature [83], [119]. However, there exists no closed-form solution for these metrics applied to GMs. We therefore use the L_2 distance [120]:

$$D(f_1 || f_2) = d(f_1, f_2) = \int (f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 \, \mathrm{d}\mathbf{x} \tag{14.4}$$

$$= \int f_1(\mathbf{x}) f_1(\mathbf{x}) \mathrm{d}\mathbf{x} - 2 \int f_1(\mathbf{x}) f_2(\mathbf{x}) \mathrm{d}\mathbf{x} + \int f_2(\mathbf{x}) f_2(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{14.5}$$

Each of the three terms in Equation (14.5) can be written as

$$\int f_a(\mathbf{x}) f_b(\mathbf{x}) \mathrm{d}\mathbf{x} = \sum_{i=1}^{J_a} \sum_{j=1}^{J_b} w_a^{(i)} w_b^{(j)} \int \mathcal{N}(\mathbf{x}; \mu_a^{(i)}, \sigma_a^{(i)}) \mathcal{N}(\mathbf{x}; \mu_b^{(j)}, \sigma_b^{(j)}) \mathrm{d}\mathbf{x} : a, b \in \{1, 2\}. \tag{14.6}$$

Equation (14.6) is easily solved by applying the following equation:

$$\int \mathcal{N}(\mathbf{x}; \mu_1, \sigma_1) \mathcal{N}(\mathbf{x}; \mu_2, \sigma_2) d\mathbf{x} = \mathcal{N}(\mathbf{0}; \mu_1 - \mu_2, \sigma_1 + \sigma_2). \tag{14.7}$$

Therefore, the L_2 distance can be computed in closed form as follows:

$$\begin{aligned} D(f_1 || f_2) = & \sum_{i=1}^{J_1} \sum_{j=1}^{J_1} w_1^{(i)} w_1^{(j)} \mathcal{N}(\mathbf{0}; \mu_1^{(i)} - \mu_1^{(j)}, \sigma_1^{(i)} + \sigma_1^{(j)}) \\ & - 2 \sum_{i=1}^{J_1} \sum_{j=1}^{J_2} w_1^{(i)} w_2^{(j)} \mathcal{N}(\mathbf{0}; \mu_1^{(i)} - \mu_2^{(j)}, \sigma_1^{(i)} + \sigma_2^{(j)}) \\ & + \sum_{i=1}^{J_2} \sum_{j=1}^{J_2} w_2^{(i)} w_2^{(j)} \mathcal{N}(\mathbf{0}; \mu_2^{(i)} - \mu_2^{(j)}, \sigma_2^{(i)} + \sigma_2^{(j)}). \end{aligned} \tag{14.8}$$

Algorithm 14.1: Optimized cooperative fusion algorithm.

1: **for** $0 \leq \mathcal{W} \leq 1$ **do**
2: Compute $v_{\mathcal{W}}(\mathbf{x}) = \frac{v_1^{\mathcal{W}}(\mathbf{x}) v_2^{1-\mathcal{W}}(\mathbf{x})}{\int v_1^{\mathcal{W}}(\mathbf{y}) v_2^{1-\mathcal{W}}(\mathbf{y}) d\mathbf{y}}$ for a given $\mathcal{W}$ according to Algorithm 8.1.
3: Compute $J(\mathcal{W})$ according to Equation 14.3.
4: **end for**
5: **return** $v_{\mathcal{W}^*}(\mathbf{x})$, where $\mathcal{W}^* = \arg\min_{\mathcal{W} \in [0,1]} J(\mathcal{W})$

At this point, we insert Equation (14.8) into Equation (14.3) and solve Equation (14.2) by performing an exhaustive search for different values of $\mathcal{W}$ equidistantly spaced by 0.1.

The optimized cooperative fusion procedure in illustrated in Algorithm 14.1.

14.2 Motion and Observation Models

This section introduces the motion and the observation model used for the car and for the Mobileye camera, respectively.

14.2.1 Motion Model

We model a car as a rectangle with constant width and length (most cars are relatively similar in size and this model fits them well). The state vector suitable for car tracking uses car's position (x, y), velocity (v_x, v_y), orientation θ and turn rate ω:

$$\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y & \theta & \omega \end{bmatrix}^\top \tag{14.9}$$

The motion model predicts the evolution of the state through time. In this chapter we use a constant velocity and constant turn-rate model, in which the input noise exists on linear and rotational acceleration a and α.

14.2.2 Observation Model

In this chapter, we use Mobileye cameras. They are monocular systems with integrated algorithms for image processing and object recognition (refer to Section 4.1 for more details).

We restate here the camera measurement model for convenience. A set Z of measurements $\mathbf{z}$ that relate to separate objects present in that frame is given by

$$Z = \bigcup_i \mathbf{z_i} \tag{14.10}$$

where

$$\mathbf{z} = \left[\begin{array}{ccc|c} x & y & \theta & c \end{array} \right]^\top \tag{14.11}$$

is a measurement of a single object, and describes its center point (x, y), orientation θ and class c (e.g., car or pedestrian). Missed detections and clutter are unavoidable part of any sensing system, and they are handled by the tracking algorithm.

We constrained the range of the Mobileye camera to 50 m. Beyond this distance, the Mobileye detects objects only sporadically. As we will report below, the choice of the camera range was further motivated by the available ground truth data (at larger distances, a very few LIDAR points are reflected from a car which makes extracting ground truth impossible). The minimum distance at which the camera detects objects is around 15 m.

As explained in Section 5.2, the noise in the range measurement of the camera does not follow a Gaussian distribution. As a simplification for our filter, however, we assume that the sensor noise is Gaussian, which allows us to use a KF for the state update.

14.3 Experimental Evaluation

This section describes our simulation and real-world experimental setup used for implementing Scenario 1a and 1b, respectively. We also explain how we obtain accurate an relative localization for the vehicles in the real-world setup.

14.3.1 Simulation setup

Scenario 1a is implemented in Webots. In our simulation set-up, we have placed two Citroën C-ZERO vehicles equipped with sensors driving behind each other on a straight road. The rear vehicle, which we call the ego vehicle ($\mathcal{E}$), is equipped with a forward-looking Mobileye camera. The front vehicle, which we refer to as the cooperative vehicle ($\mathcal{C}$), is also equipped with a forward-looking Mobileye camera. In addition, two legacy vehicles ($\mathcal{O}_1$ and $\mathcal{O}_2$) drive in the opposite direction (see Figure 14.1).

As a model of the Mobileye camera was not readily available in Webots, we have created one and calibrated it using real data (cf. Section 5.2). In addition to the simulated Mobileye camera, we use virtual GNSS and IMU sensors, which are characterized by noise similar to the error of the Applanix localization system. They provide sufficiently accurate localization and we do not perform any additional filtering on the localization data. A supervisor controller is in charge of providing ground truth data.

Figure 14.1 – Setup in the Webots simulator for Scenario 1a. The ego car (yellow) drives on a straight road behind the cooperative car (white), while two non-cooperative cars drive in the opposite direction.

14.3.2 Real-World Experimental Setup

Similarly to Scenario 1a, Scenario 1b leverages two real electric Citroën C-ZERO cars. They are parked behind each other in one lane, at the relative distance of approximately 15 m. This resembles a scenario in which two cars drive behind each other at the same speed. While in simulation we have the accurate position and orientation sensors at our disposal and we can afford to move the sensing vehicles, this is not the case with the real cars. In absence of accurate localization equipment on both cars, our approach in this chapter is to park the cars and localize them manually, as explained below.

The rear vehicle ($\mathcal{E}$) is equipped with a forward-looking Mobileye camera, and two forward- and two backward-looking Ibeo LIDAR sensors (that we use for ground truth only and not for tracking). The front vehicle ($\mathcal{C}$) is endowed as well with a forward-looking Mobileye camera (see Figure 4.1). Both vehicles have on-board computers for real-time sensor processing and logging, as well as wireless communication equipment (see Phase 1 of the deployment described in Chapter 4.4 for more details). Our scenario includes two more legacy oncoming vehicles ($\mathcal{O}_1$ and $\mathcal{O}_2$) of different type and size, which do not carry neither sensory nor computational equipment. The experimental setup is visualized in Figure 14.2.

14.3.3 Localization of Real Vehicles

It is of great importance for the functioning of the cooperative fusion algorithm to have an accurate relative pose of the cooperative vehicle with respect to the ego vehicle. In absence of centimeter-level GNSS devices installed on both experimental vehicles, in Scenario 1b we enforce vehicle stationarity and tune the poses manually, by using the LIDAR sensors mounted at the front of our ego vehicle, and matching its point cloud to the CAD model of the Citroën

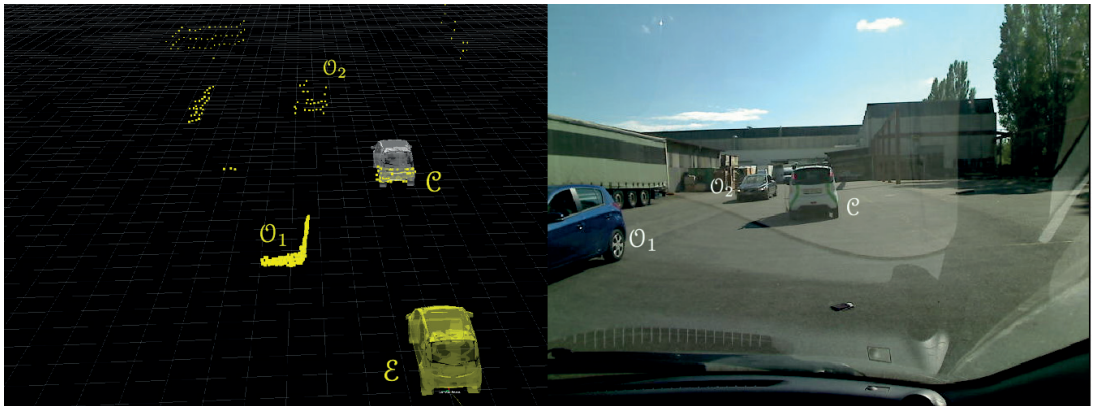


Figure 14.2 – Experimental setup. Left: ego vehicle (yellow), cooperative vehicle (white), and point cloud generated by the LIDAR mounted on the ego vehicle (legacy vehicles are easily identifiable). Right: view from the web camera installed behind the windshield of the ego vehicle.

C-ZERO. In other words, we consider accurate localization of the vehicles to be a separate problem and assume that it is available. The pose of the center of vehicle $\mathcal{C}$ in the local frame of vehicle $\mathcal{E}$ is determined to be $(x, y, \theta) = (15.45\text{ m}, -0.25\text{ m}, -0.0175\text{ rad})$. Despite the manual localization, the notion of uncertainty is still present in the car state, as the inter-vehicle distance obtained from the LIDAR measurements is noisy.

In our real-world experimental setup, we do not have localization equipment which could provide ground truth for the tracked (legacy) vehicles in our scenario. Therefore, we use LIDAR data to extract the ground truth, taking advantage of the fact that a LIDAR point is accurate to 0.1 m (one σ) and that the experimentally determined accuracy of the Mobileye camera is significantly worse than that. Moreover, as in this chapter only the measurements of the Mobileye camera are used for tracking, LIDAR data represent an independent source of information.

In order to extract the ground truth positions of the legacy vehicles from the LIDAR point cloud, for each experimental run, we determine the contours of the legacy vehicles by visualizing the point cloud in the ROS tool `rviz`. Each ground truth set contains at least 10 manually extracted sample points, and vehicle positions in-between these sample points are linearly interpolated.

The maximal distance of a detectable object in our setting is around 70 m (objects at larger distances were occluded). This is another reason to limit the range of the Mobileye camera to 50 m, as the maximum range of the cooperative system amounts to around 65 m (given the distance between vehicles $\mathcal{E}$ and $\mathcal{C}$).

14.3.4 Parameters

The tracking parameters were the same in all runs. The maximum number of Gaussian components per filter J_{max} is limited to 100. The birth model for the GM-PHD filter consists of only one Gaussian component. Its position is in the center of the Mobileye camera's FOV, the heading is set to π rad, whereas the birth velocity and the turn rate are set to zero. The Standard Deviation (SD) of the birth component is 25 m (half of the sensor range) for its position, π rad for the heading, 10 $^{\text{m}}/_{\text{s}}$ for the speed, and 0.2 $^{\text{rad}}/_{\text{s}}$ for the turn rate. The rationale behind these values is that a new object entering the sensing FOV is very likely to be contained within one SD from the birth component mean.

The SD for the noise in the motion model is set to 1 $^{\text{m}}/_{\text{s}^2}$ for the linear acceleration and 0.1 $^{\text{rad}}/_{\text{s}^2}$ for the rotational acceleration. The measurement noise is determined empirically and is modeled as a Gaussian with SD $\sigma_x = 0.5$ m, $\sigma_y = 0.3$ m, and $\sigma_\theta = \frac{\pi}{16}$ rad. The clutter model uses a Poisson distribution with an expected cardinality of 1 measurement per sensor return per surveillance area. The $\mathcal{E}$ and $\mathcal{C}$ coordinate frames' uncertainty (1σ) is set to $[0.5\text{ m}, 0.3\text{ m}, 0.1\ ^{\text{m}}/_{\text{s}}, 0.1\ ^{\text{m}}/_{\text{s}}, 0.0174\text{ rad}, 0.0174\ ^{\text{rad}}/_{\text{s}}]^\top$.

The GM-PHD merging parameter is set to $U = 10$, and a Gaussian component is pruned if its weight is less than $T_p = 10^{-5}$. The extraction threshold T_e is set to 0.5. The fusion distance parameter U_F is empirically set to 30. The probability of survival in the joint FOV is set to 0.99. The maximal probability of detection is $p_D^{max} = 0.9$. The shape of the object (tracked vehicles) is assumed to be rectangular, with length of 3.5 m and width of 1.5 m.

14.3.5 Results

The Scenario 1a is used to assess the performance of the cooperative perception algorithm when the cars $\mathcal{E}$ and $\mathcal{C}$ move at 30 and 27 $^{\text{km}}/_{\text{h}}$, respectively. The two legacy cars in simulation move at speeds of 21 and 26 $^{\text{km}}/_{\text{h}}$, respectively.

A real-world dataset consisting of 15 runs is collected with the purpose of evaluating the cooperative perception algorithm using real cars. In each of the runs, cars $\mathcal{E}$ and $\mathcal{C}$ were positioned at the same spot, while cars $\mathcal{O}_1$ and $\mathcal{O}_2$ were driven manually in the opposite direction (i.e., both towards $\mathcal{E}$ and $\mathcal{C}$) at the speed of approximately 20 $^{\text{km}}/_{\text{h}}$. For the sake of comparison, for each run both C-GM-PHD (includes cooperative fusion) and regular PHD filter (does not include cooperative fusion) were run on the ego vehicle.

Similarly to Part III, here we again use the OSPA metric for evaluation of the multi-object tracking performance. We set the position sensitivity parameter $p = 1$, i.e., we compute the position error using the Euclidean distance between the center of the estimated and the ground truth object. The cut-off parameter for the cardinality error penalties is set to $c = 10$.

Figure 14.3 shows the OSPA metric for a simulated experiment. Given the similarity with the results from Part III, we conclude that we have successfully implemented our algorithm in the

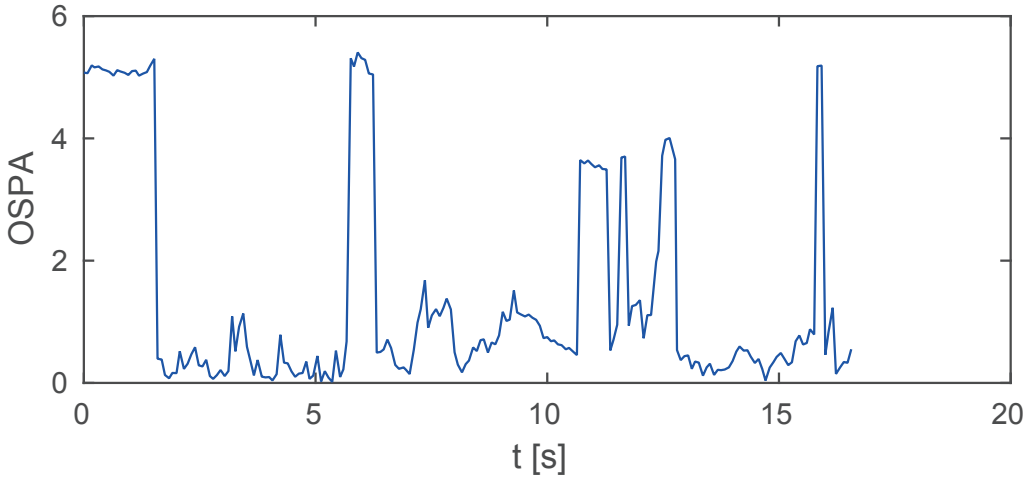


Figure 14.3 – OSPA over time for one run of the C-GM-PHD filter in the Webots simulator.

ROS framework. By looking at the results obtained in real-world experiments presented below, we observe the similarity in tracking performance between the simulated and the real-world experiments. The OSPA error is slightly larger in the simulated experiment. The reason for this could lie in the error introduced by the motion of the sensing cars, or in the noise of the simulated Mobileye camera.

Let us now look at the real-world performance (Scenario 1b). Figure 14.4 shows a trajectory of the car $\mathcal{O}_1$ obtained during the first run of the real-world experiment. When $\mathcal{O}_1$ is only observed by the car $\mathcal{C}$, the estimation uncertainty is large as the uncertainty of the relative localization between cars $\mathcal{E}$ and $\mathcal{C}$ needs to be integrated into the track estimation uncertainty. Once the $\mathcal{O}_1$ enters the FOV of the car $\mathcal{E}$, the uncertainty reduces progressively. Around $x = 30$ m we observe that, as expected, the track estimate gets closer to the $\mathcal{E}$ measurements, since its uncertainty is significantly smaller than the uncertainty of the track received from the car $\mathcal{C}$ (which contains both measurement uncertainties associated to the Mobileye camera and the localization uncertainty due to the relative positioning based on the LIDAR measurements). In the case when no cooperative fusion is used (Figure 14.4 bottom), the track uncertainty only depends on the motion and measurement uncertainty, as the tracking is performed in the local frame of the car $\mathcal{E}$.

The OSPA performance for the first run is shown in Figure 14.5. For the cooperative version of the filter, the ground truth objects are considered in the union of the two vehicles' FOVs. On the other hand, to enable a fair comparison, for the non-cooperative filter only ground truth objects in the FOV of the ego car are taken into account. At the beginning, only the vehicle $\mathcal{C}$ is tracked by the vehicle $\mathcal{E}$, until the car $\mathcal{O}_1$ enters the FOV. Spikes indicate cardinality penalties. Wrong cardinality estimates can be due to delay in object birth or death, loosing track of objects for short periods of time or counting twice the same object for short amounts of time. In Figure 14.5, all spikes represent birth/death delays. This leaves us with an impression that the performed fusion weight optimization allows for a more robust algorithm performance as

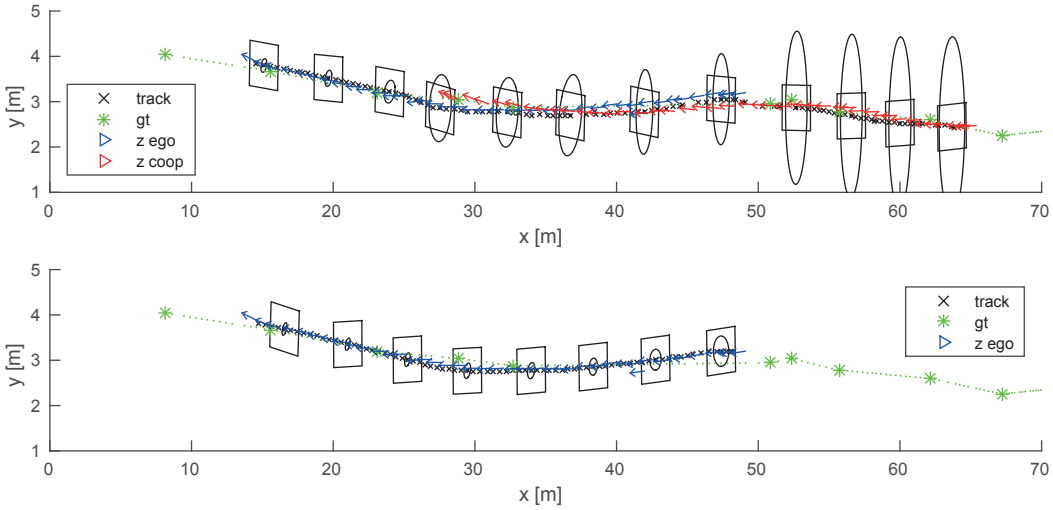


Figure 14.4 – Trajectory of the car $\mathcal{O}_1$ during the first run of the real-world experiment (Scenario 1b) produced by the C-GM-PHD filter (top) and the GM-PHD filter (bottom). $\mathcal{O}_1$ starts from far away (70 m) and approaches the tracking vehicles. Black crosses indicate the estimated track positions. Vehicle rectangle including its orientation is showed every 10 filter update cycles (rectangles have been scaled down for clarity). Ellipses in black indicate one standard deviation for (x, y) uncertainty. Ground truth trajectory sample points are shown using green asterisks, and interpolated trajectory is shown in green dots. Ground truth does not contain orientation information. Blue and red arrows represent Mobileye measurements (position and orientation). Every second measurement is shown to avoid the image being very cluttered. Note the difference between x and y spatial scales.

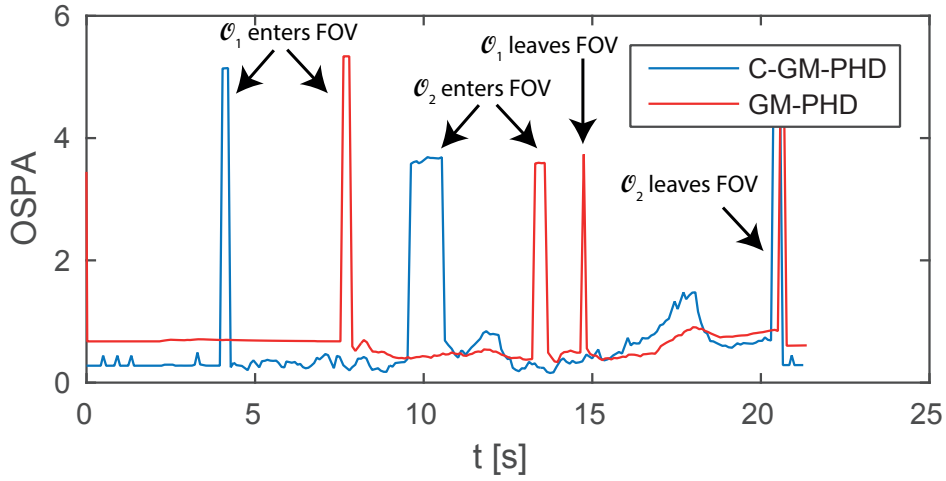


Figure 14.5 – OSPA over time for a single run using real-world data gathered at the level of the vehicle $\mathcal{E}$. Performance of the cooperative filter is compared against LIDAR-based ground truth in the union of FOVs of the cars $\mathcal{E}$ and $\mathcal{C}$, while for the non-cooperative filter only ground truth objects in the FOV of the car $\mathcal{E}$ are considered.

compared to Chapter 8, in a sense that the algorithm loses track of the objects less frequently.

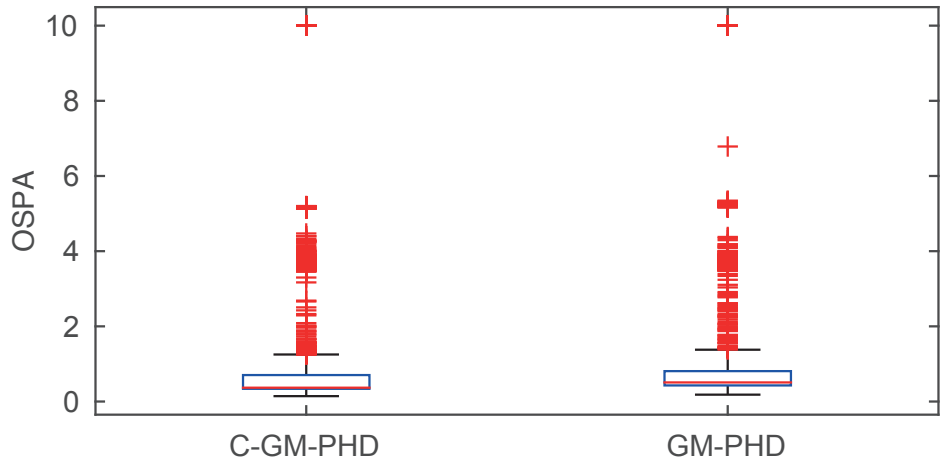


Figure 14.6 – Comparison of OSPA performance accumulated during 15 experiment runs between the cooperative and non-cooperative filter.

Statistics of tracking error aggregated over 15 runs are shown using the box plot in Figure 14.6. We can see that a median error of the cooperative filter is slightly lower than its non-cooperative counterpart. However, their performance is very similar. The absence of statistically significant improvement of the cooperative filter in the intersecting FOV can be attributed to the conservative nature of the GCI method.

It is important to note that here the comparison is performed only in the FOV of the car $\mathcal{E}$, as the non-cooperative filter cannot track objects outside of this FOV. Moreover, the localization uncertainty plays an important role in the accuracy of the cooperative tracking algorithm, as the uncertainty of $\mathcal{E}$ and $\mathcal{C}$ coordinate frames gets integrated in the track estimation uncertainty. In Figure 14.4 we can see that it takes a significant amount of time for the uncertainty to be reduced when the cooperative fusion is employed. Consequently, it is not unexpected that the cooperative filter tracks objects outside of the ego car's FOV with less accuracy than the objects in the ego FOV, thus comparing the accuracy of the two filters over the union of the FOVs would be unfair.

Summary

This chapter presented our efforts in transition from simulation to real vehicles. We introduced a weight optimization procedure, an important enhancement to our cooperative perception algorithm. Moreover, we used Mobileye cameras as tracking sensors. As a first step, we created a realistic simulated scenario compatible with ROS, in which two intelligent vehicles drove one behind another and cooperatively tracked two legacy vehicles that drove in the opposite direction. We then presented our first deployment of the C-GM-PHD algorithm on two real cars. As we did not have accurate localization systems on both cars, we parked them one behind another and we resolved their mutual localization by hand based on LIDAR data. LIDAR data were also used as an independent ground-truth system for localizing the legacy vehicles driving by.

15 Cooperative Localization Algorithms

ACCURATE localization is essential for many intelligent vehicle applications, ranging from perception to planning. This is also true for cooperative perception algorithms. While some perception tasks or advanced driver assistance systems can be implemented in the local vehicle's frame (hence bypassing the localization requirement), cooperation between moving vehicles requires appropriate methods for moving from one local coordinate frame to another. Therefore, a common (global) frame is needed to operate among local frames of multiple intelligent vehicles willing to cooperate with each other.

In this chapter, we present our contribution to this research area. The first algorithm we present is a standard EKF that integrates a GNSS sensor with inertial sensors and an odometer. Then two algorithms for relative localization refinement are presented. In essence, they operate by matching objects tracked by the cooperating vehicles and finding the rigid transform which minimizes the inter-vehicle object mismatch. The output of the matching algorithms is then fed to the EKF and serves for pose refinement. The EKF provides a continuous localization estimate regardless of availability of the matching algorithm output.

Our approach represents a logical extension of the cooperative perception algorithms, because the data it requires as input are inherently made available to other vehicles because of the cooperative perception algorithms. No additional data (such as LIDAR point clouds or local maps, if available) need to be shared among the vehicles. It particularly makes sense to apply the cooperative localization approach on a vehicle with less accurate localization (in our case `car2`), which can be aided by a vehicle with a more accurate positioning system (the `car1` in our case).

15.1 Background

The probably most frequently used approach for outdoor localization of mobile robots and intelligent vehicles is based on a KF and its variations for nonlinear problems (such as an Exten-

ded KF or Sigma Point KF). They are used to fuse proprioceptive sensors (e.g., accelerometers, gyroscopes, odometers) together with exteroceptive sensors (e.g., GNSS sensors) in order to provide a global pose estimate. A performance comparison of KFs for vehicle localization can be found in [121].

When the above mentioned approaches do not yield the desired performance, additional exteroceptive sensors that seek for landmarks in the environment are often used to aid the localization (e.g., LIDARs, cameras). If a map of the environment is known, features can be extracted from the sensor data and potential locations on the map can be suggested. Roumeliotis and Bekey propose an approach based on a KF to perform map-based localization [122]. Algorithms based on KFs make restrictive assumptions on the size of the error and the shape of the robot's (or vehicle's) uncertainty. Monte Carlo localization solutions relax these assumptions but they are typically more computationally expensive [123].

If a map is not available, it can be built simultaneously with localization. The class of algorithms in which a map is created, landmarks are identified in the map, and the pose is corrected at each step according to the landmarks, is called Simultaneous Localization and Mapping (SLAM) [124]. In canonical SLAM implementations, landmarks and vehicle states are represented in a vector form and either EKF-SLAM or FastSLAM are used to update the state as seen in [125]. EKF-SLAM is performed by using an EKF to update a vehicle position based on expected landmark locations. The EKF-SLAM does not provide data association and therefore another method has to be used in order to relate stored with observed landmarks. It has a quadratic complexity and is therefore costly for a large set of landmarks. FastSLAM uses a PF to estimate the posterior of a vehicle's path, contrary to EKF-SLAM where the posterior pose is estimated. It scales logarithmically with the number of particles and is faster than EKF-SLAM for larger environments. When an already mapped area is revisited, the map loop can be closed by loop closure as seen in [126], for example. For EKF-SLAM there are correlations between each landmark, stored in the covariance matrix, but since this is not the case with FastSLAM the loop closure becomes more difficult.

Multiple robots (or vehicles) can cooperate with each other with the aim to achieve a more accurate localization. In [127], multiple heterogeneous robots, able to detect each other, collaborate to localize themselves faster, maintain higher accuracy, and decrease cost of individual robots by spreading the different kinds of sensors among multiple platforms. Prorok et al. show in [128] a solution for collaborative localization of large robot teams composed of cheap robots, that has low computational and sensing requirements. Their solution relies on a Monte-Carlo localization method and reciprocal sampling. In [115], [129], teams of autonomous underwater vehicles obtain range and/or angle measurements to each other and dead-reckoning information to help each other navigate more accurately. The problem of inconsistent (overconfident) estimates due re-use of received measurements is addressed. An approach for cooperative localization of groups of intelligent vehicles is presented in [130]. Each vehicle can estimate its position using GNSS and motion sensors, and maintains an estimate of group state. This estimate is shared with neighboring vehicles and updated with both the sensor data of the

ego-vehicle and the estimates sent from other vehicles using a CI filter.

Cooperative SLAM approaches have also been proposed in the literature. For instance, in [131], the relative pose between vehicles is estimated indirectly, by maximizing the consistency degree between the maps of the area surrounding the vehicles obtained by local SLAM algorithms on individual vehicles. A genetic algorithm is employed for solving the optimization problem. This method requires sharing local maps among vehicles.

Inspiration for further map-matching algorithms can be borrowed from the computer vision field. Registration of point sets can be performed using the Iterative Closest Point (ICP) or Random Sample Consensus (RANSAC) algorithm (see for example [132]–[134]), or using GMM (see for example [120], [135]). An interesting approach for map-matching using descriptors extracted from radar maps is proposed by Rapp et al. [136]. The drawback of all these approaches is that they require a map.

Instead of sharing and matching maps, Rauch et al. evaluate in [137] approaches for inter-vehicle matching of tracked objects based on the ICP and GMM algorithms mentioned above. The advantage of this approach is that, apart from data already shared for the purpose of cooperative fusion, no additional data need to be shared to perform cooperative localization. However, this approach can only be applied when a sufficient number of objects is tracked by each of the cooperative vehicles (i.e., a sufficient overlap between the tracking vehicles' FOV is required). At other times vehicles need to localize each on its own. This is why we propose an integrated approach based on the EKF filter for localization using Dead-Reckoning (DR) sensors, GNSS sensors, as well as updates from a cooperative object-matching algorithm.

15.2 Extended Kalman Filter Algorithm

Standard automotive localization algorithms typically use dead reckoning in order to overcome the limitation of GNSS technology alone. Satellite signals are unavailable in tunnels and garages, and are often severely degraded in urban canyons and near trees due to blocked line of sight to the satellites or multi-path propagation. Sensors used for dead-reckoning are often already present in cars for other purposes, such as Anti-lock Braking System (ABS) or Electronic Stability Control (ECS), and their signals can be read from the CAN bus.

The localization algorithm presented here uses an EKF to integrate the always-available sensor data with occasionally unavailable position information from the satellite into a combined position fix. In the same fashion, it integrates pose refinement information obtained by the object-matching algorithms proposed later in this chapter.

Our goal is to estimate the 2D pose (x, y, θ), corresponding velocities (v_x, v_y, ω) and 2D linear acceleration (a_x, a_y) of a vehicle over time. The process can be described as a nonlinear

dynamical system, with

$$\mathbf{x}_{k|k-1} = f(x_{k-1}) + \mathbf{w}_{k-1} \tag{15.1}$$

where

$$\mathbf{x}_{(\cdot)} = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega & a_x & a_y \end{bmatrix} \tag{15.2}$$

is the 8-D state vector, f is a nonlinear state transition function, and $\mathbf{w}_{k-1}$ is the normally distributed process noise. Additionally, we receive measurements

$$\mathbf{z}_k = h(\mathbf{x}_{k|k-1}) + \nu_k \tag{15.3}$$

where $\mathbf{z}_k$ is the measurement at time k, h is a nonlinear sensor model that maps the state into the measurement space, and ν_k is the measurement noise, which is also assumed to be normally distributed.

The algorithm uses a generic omni-directional motion model [138], which works well for a broad range of robots. More suitable models for cars exist (such as the one we used in Chapter 8). However, we took advantage of an existing ROS implementation [139] of the EKF with omni-directional model, and we tuned the process noise covariance to mitigate this effect. The pose is defined in the global frame, whereas the velocities and accelerations are expressed in the body frame of the vehicle and need to be rotated to the global frame first. The vector prediction equation of the omni-directional model $\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1})$ can be broken down into individual equations as follows:

$$\begin{aligned} x_{k|k-1} = x_{k-1} &+ \left(v_{x_{k-1}} \cos\theta_{k-1} - v_{y_{k-1}} \sin\theta_{k-1}\right)\Delta t \\ &+ 0.5\left(a_{x_{k-1}} \cos\theta_{k-1} - a_{y_{k-1}} \sin\theta_{k-1}\right)\Delta t^2 \end{aligned} \tag{15.4}$$

$$\begin{aligned} y_{k|k-1} = y_{k-1} &+ \left(v_{x_{k-1}} \sin\theta_{k-1} + v_{y_{k-1}} \cos\theta_{k-1}\right)\Delta t \\ &+ 0.5\left(a_{x_{k-1}} \sin\theta_{k-1} - a_{y_{k-1}} \cos\theta_{k-1}\right)\Delta t^2 \end{aligned} \tag{15.5}$$

$$\theta_{k|k-1} = \theta_{k-1} + \omega_{k-1}\Delta t \tag{15.6}$$

$$v_{x_{k|k-1}} = v_{x_{k-1}} + a_{x_{k-1}}\Delta t \tag{15.7}$$

$$v_{y_{k|k-1}} = v_{y_{k-1}} + a_{y_{k-1}}\Delta t \tag{15.8}$$

$$\omega_{k|k-1} = \omega_{k-1} \tag{15.9}$$

$$a_{x_{k|k-1}} = a_{x_{k-1}} \tag{15.10}$$

$$a_{y_{k|k-1}} = a_{y_{k-1}}. \tag{15.11}$$

The Jacobian of the transfer function $f(\cdot)$ is computed analytically.

Furthermore, by using the state vector defined in Equation (15.2), integration of various sensors is facilitated. Every variable that is measured by the available sensors is present in the state vector, and thus the measurement model can be expressed as an $m \times 8$ matrix H of rank m with

its only nonzero values (in this case, ones) existing in the columns of the measured variables, where m is the number of variables measured by a given sensor.

When the localization algorithm is first started, $\mathbf{x_0}$ is initialized to the origin of the odometry coordinate system. Using the UTM coordinates of the first reported GNSS measurement $[x_{UTM_0}, y_{UTM_0}]$ and the first reported heading in the UTM frame ϕ_0, a transformation matrix T from the odometry frame to UTM frame is created:

$$T = \begin{bmatrix} cos\phi_0 & -sin\phi_0 & x_{UTM_0} \\ sin\phi_0 & cos\phi_0 & y_{UTM_0} \\ 0 & 0 & 1 \end{bmatrix} \tag{15.12}$$

Each subsequent GNSS measurement at time k is converted from the UTM frame to the odometry frame using the equation

$$\begin{bmatrix} x_{\text{odom}_k} \\ y_{\text{odom}_k} \\ 1 \end{bmatrix} = T^{-1} \begin{bmatrix} x_{\text{UTM}_k} \\ y_{\text{UTM}_k} \\ 1 \end{bmatrix}, \tag{15.13}$$

and the converted measurement is fused with the state estimate in the EKF.

Outputs from object matching algorithms presented in the remainder of this section are also given in the UTM frame in the form of a 2D pose, and are fused with the EKF state estimate after applying Equation (15.13). The orientation is simply the difference $\theta_{\text{UTM}} - \phi_0$.

15.3 Relative Localization by Matching the Leading Vehicle

The previously presented approach provides global localization, but its accuracy depends on the quality of the proprioceptive and GNSS-based sensors, as well as on the environment in which a vehicle is driving. For cooperative perception, very accurate relative localization between the cooperating vehicles is more important than their absolute localization accuracy.

In this section, we present a simple algorithm that enhances the accuracy of the relative localization between the two vehicles. An assumption is that there exists a cooperative vehicle on the same lane in front of the ego vehicle. This assumption holds in one of the typical scenarios in which cooperative perception is beneficial (see for example the overtaking case study presented in Chapter 10). Since the leading vehicle might not always be visible, or its track may get lost, we intend to find a corrected pose of the ego vehicle to use as an additional measurement in the EKF presented in Section 15.2. The EKF will additionally smooth out the measurements, as it balances between this update and other available sensors. The concept of using the leading car for location refinement is similar to using measurements originating from known landmarks.

The leading vehicle is detected and tracked using the forward-looking sensors of the ego vehicle. The leading vehicle additionally communicates its pose along with the state of objects it tracks. If multiple vehicles are tracked at time k, the correct one needs to be associated with the received pose. This is first done in local coordinates, by bounding the lateral distance to the current lane, and constraining the longitudinal distance to be strictly positive. The final check is performed in the UTM frame, by confirming that the distance between the tracked and received pose is sufficiently small.

The goal is to find the new pose of the ego vehicle, for which the distance between the pose of the leading vehicle in the global frame obtained by its localization system $\mathbf{x}_L$ and its pose obtained by the tracking algorithm of the ego vehicle $\mathbf{x}_T$ is minimized. This is performed by looking for the optimal 2D rigid transformation (rotation and translation).

Translation and rotation in 2D have three Degrees Of Freedom (DOFs) — two for translation and one for rotation. It is thus sufficient to use two poses in 2D to determine the rigid transformation. The angle of rotation θ_R can first be determined as

$$\theta_R = \theta_L - \theta_T. \tag{15.14}$$

The translation is then computed:

$$\mathbf{t} = \begin{bmatrix} x_L \\ y_L \end{bmatrix} - R \begin{bmatrix} x_T \\ y_T \end{bmatrix}, \tag{15.15}$$

where R is the 2D rotation matrix

$$R = \begin{bmatrix} \cos\theta_R & -\sin\theta_R \\ \sin\theta_R & \cos\theta_R \end{bmatrix}. \tag{15.16}$$

Finally, the measurement to be used in the EKF update is computed by applying the found transform to the last pose of the ego vehicle. Measurement uncertainty (covariance matrix) is the localization uncertainty of the leading vehicle, augmented by the tracking uncertainty.

15.4 Relative Localization by Matching Tracked Objects

The assumption introduced in Section 15.3 holds only in a limited number of scenarios. In this section we present a more general approach which relaxes the aforementioned assumption.

In this thesis, we track cars (cf. Part III) and pedestrians (as will be shown in Chapter 16). While it was reasonable to estimate the heading of tracked cars, the estimation of pedestrians' heading using LIDARs in very unreliable. Common for tracking cars and pedestrians is that their center position in the Cartesian coordinates is tracked. Therefore, to estimate the 2D rigid transform between the two intelligent vehicles, at least two objects are required to be tracked by both of the vehicles at given time. An overview of the algorithm is given in Algorithm 15.1.

Algorithm 15.1: Overview of the localization algorithm based on object matching.

Require: Latest GNSS-updated position of the ego vehicle $x_{\text{EGO}}^{\text{UTM}}$
Require: Object sets $X^{(1)}$ and $X^{(2)}$ s.t. $X^{(\cdot)} >= 2$
for $x_i^{(1)} \in X^{(1)}$ **do** {Define cost matrix}
  for $x_j^{(2)} \in X^{(2)}$ **do**
    $C[i,j] \leftarrow D_B(x_i^{(1)}, x_j^{(2)})$ {Bhattacharyya distance}
  end for
end for
Threshold the cost matrix
Compute minimum cost assignment between $x_i^{(1)} \in X^{(1)}$ and $x_j^{(2)} \in X^{(2)}$ using the Hungarian algorithm
Run the Singular Value Decomposition (SVD) algorithm on assigned pairs to find the rigid transformation (R, t)
Compute the Mean Square Error (MSE) of the transformation
Apply the transformation (R, t) to the pose of the ego vehicle and create new measurement $z = [x, y, \theta]$
if $|z - x_{\text{EGO}}^{\text{UTM}}| < e_{\text{GATE}}$ **then**
  return z
else
  return None
end if

The algorithm starts by performing association between the two object sets. A cost matrix defining the assignment cost for each pair of objects is computed. We use the Bhattacharyya distance [140] to compute the assignment cost:

$$D_B = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_1 \det \Sigma_2}} \right), \tag{15.17}$$

where μ_i and Σ_i are the means and covariances of the two objects' state (only the first two dimensions) transformed into the global frame, and

$$\Sigma = \frac{\Sigma_1 + \Sigma_2}{2}. \tag{15.18}$$

Given an object tracking covariance in the local (tracking) frame Σ_T^{local} and the localization covariance of the tracking vehicle Σ_L^{UTM}, the covariance of the tracked object in the global (UTM) frame can be computed as

$$\Sigma_T^{\text{UTM}} = \text{J}_1 \Sigma_T^{\text{local}} \text{J}_1^T + \text{J}_2 \Sigma_L^{\text{UTM}} \text{J}_2^T \tag{15.19}$$

where $\text{J}_1 = \frac{\partial \mathbf{x}_T^{\text{UTM}}}{\partial \mathbf{x}_T^{\text{local}}}$ and $\text{J}_2 = \frac{\partial \mathbf{x}_T^{\text{UTM}}}{\partial \mathbf{x}_L^{\text{UTM}}}$ denote respectively the Jacobian of the global position with respect to the local position and the localization position.

The FOVs of the two tracking vehicles may only partially overlap, and their tracking sets may contain objects that are detectable by one of the vehicles only. Before proceeding with association, we threshold the cost matrix to remove the objects with high cost to all other objects, as we do not want unrelated objects to affect the global association process. The association is found using the Hungarian algorithm [141]. Given two object sets and the cost matrix, the Hungarian algorithm finds the assignment of objects in one set to objects in the other set that minimizes the cost. The assignment is complete if the cardinality of the first set is less than or equal to the cardinality of the second set; otherwise not every object from the first set needs to be assigned to an object in the second set. At the output of the Hungarian algorithm, a subset of objects with their respective associations is created.

The two 2D object sets $\{\mathbf{x}_i^{(1)}\}$ and $\{\mathbf{x}_i^{(2)}\}$, $i = 1,\ldots,N$ (here $\{\mathbf{x}_i^{(1)}\}$ and $\{\mathbf{x}_i^{(2)}\}$ are considered as 2×1 column matrices) are related to each other using the identity

$$\mathbf{x}_i^{(2)} = R\mathbf{x}_i^{(1)} + \mathbf{t} + N_i, \tag{15.20}$$

where R is a 2×2 rotation matrix, $\mathbf{t}$ is a translation vector (2×1 column matrix), and N_i a noise vector. The goal is to find R and $\mathbf{t}$ such that the following sum is minimized:

$$D_{\text{LSE}} = \sum_{i=1}^{N} (\mathbf{x}_i^{(2)} - (R\mathbf{x}_i^{(1)} + \mathbf{t}))^2. \tag{15.21}$$

It was shown in [142] that this least-squares problem can be solved by using SVD to find R which minimizes D_{LSE}, and then find the translation as $\mathbf{t} = \bar{\mathbf{x}}^{(2)} - R\bar{\mathbf{x}}^{(1)}$, where $\bar{\mathbf{x}}^{(\cdot)} = \frac{1}{N}\sum \mathbf{x}_i^{(\cdot)}$.

Finally, the transformation is applied to the latest pose of the ego vehicle and a pose new measurement in the global (UTM) frame is hence created. The measurement covariance is set to the leading vehicle localization covariance, augmented by the $\frac{\text{RMS}}{\sqrt{2}}$ factor for the x and y coordinates, where RMS is the root mean squared error of the least-squares fit, and by a small constant for the heading coordinate. A validation gate is set up around the latest vehicle pose which was updated using the GNSS, and all updates falling outside of the gate are rejected. This is especially helpful at the initialization of the algorithm, when the risk of wrong association is higher due to high initial localization uncertainty (which in turn results in large object covariances in the global frame). The accepted measurements are fused with the vehicle state estimate in the EKF.

15.5 Evaluation

In this section, we evaluate the performance of the localization algorithms previously described. We gradually increase the number of sensors we fuse in the EKF. We start by DR, where we fuse the forward speed, turning rate (rotational speed around the vertical axis), and accelerations in the horizontal plane. We also feed a lateral speed, which we set to 0 (this is a reasonable assumption for relatively low speeds where there is not a lot of lateral slip). The

compass is not used because it turned out to be very unreliable, with random biases and large jumps in response to various objects in the environment. The exception is the initial heading measurement, which is along with the initial GPS position used in Equations (15.12)-(15.13) to establish a relationship between the global and the odometry coordinate frame. Thus, in the DR filter, the heading is computed as an initial heading plus the integrated turning rate.

In the second algorithm, we additionally fuse the measurements from the GPS (recall that the U-blox 5 receiver we use only works with the GPS satellites). Although the GPS measurements do not contain orientation information, they indirectly affect the orientation as the filter needs to update the orientation in order to move the estimate towards the GPS measurement.

Finally, in the third algorithm, we additionally fuse in the measurements from our matching algorithm. At the same time, we gate the GPS measurements, for the reasons explained below.

Our localization algorithms run on `car2`. The performance of a localization system can be analyzed qualitatively by plotting a vehicle's trajectory on a satellite map. For quantitative evaluation, a reference localization system is required. We use the Applanix localization system available on `car1` for that purpose, while driving the `car2` behind the `car1` and following its trajectory as closely as possible. When using RTK corrections, the advertised localization error of the Applanix system can be as low as 3 cm. In practice, we managed to achieve a localization accuracy of 3 cm only when the car was stationary. The localization accuracy of a moving car in good conditions was around 0.2 – 0.3 m. We note that this is the estimated accuracy reported by the Applanix system — we do not have available an additional reference against which we could validate the reported performance of this system.

In the first evaluation scenario, the two cars start driving from the beginning of a street; the cars drive forward at approximately 30 $^{\mathrm{km}}/_{\mathrm{h}}$ behind each other, make a U-turn at the end of the street, drive back, make another U-turn and come to a stop at approximately their starting position. Figure 15.1 shows the trajectory of the reference car using green color. The trajectory obtained by DR is shown in yellow, the trajectory of DR+GPS in blue, and the trajectory corrected using the matching algorithm in red.

The distance between the reference trajectory and the outputs of the three filters is computed in a way similar to [143]. For each point on the `car2` trajectory, the closest point on the reference trajectory is found. The distance between trajectories is the sum of Euclidean distances between all point pairs, normalized by the number of points in the trajectory. We note that the reference trajectory runs ahead of the `car2` trajectory by a few seconds. Limiting the search space in time helps identifying the correct point on the reference trajectory, avoiding thereby wrong associations, e.g., association of points corresponding to the car going south-west with the part of the reference trajectory belonging to the car going north-east. It also speeds up the algorithm.

Table 15.1 quantifies the localization error. Recall that the reference system is not installed on the same vehicle on which the localization algorithms run. Thus a part of the error comes

Figure 15.1 – Vehicle localization using different algorithms when driving around 30 km/h. Every 10th point of the reference vehicle trajectory is shown in green. The reference vehicle starts from the south end of the street, drives towards north-east, makes a U-turn, drives back in the south-west direction, makes another U-turn and stops at its starting position. The `car2` drives behind the reference vehicle, following approximately the same trajectory. The trajectory of `car2` obtained using dead-reckoning is shown in yellow, blue corresponds to the trajectory of the filter with dead-reckoning and GPS, whereas red depicts the trajectory obtained with our matching algorithm.

from the real difference in the driven trajectories, especially around the U-turns.

The algorithm which only uses DR produces a very smooth trajectory but drifts to a side, resulting in the largest error. Notably, the algorithm fusing DR and GPS produces a trajectory with some jumps. After investigating the issue, we concluded that there exists a large delay in the signal received from the GPS sensor (in the order of a couple of seconds). These GPS measurements, while being global and preventing the algorithm from drifting away, constantly pull the moving vehicle behind and slow down the speed estimate (as visible in Figure 15.1). The performance is especially degraded in sharp turns (such as U-turns), because the old GPS measurements affect the heading of the car in a wrong direction. It takes a significant amount of time for the algorithm to recover the correct heading after the turn. We also note that the performance of this algorithm reported in Table 15.1 cannot reflect the aforementioned speed issue in the largest part of the trajectory, due to the way we compute the distance between

Table 15.1 – Comparison of mean localization errors for trajectories obtained using different localization algorithms at the driving speed of around 30 km/h.

Algorithm	DR	DR+GPS	Matching
Mean error	7.93 m	1.08 m	0.57 m
Minimum error	0.05 m	0.008 m	0.002 m
Maximum error	19.28 m	10.40 m	2.14 m

Table 15.2 – Comparison of mean localization errors for trajectories obtained using different localization algorithms at the driving speed of around 6 km/h.

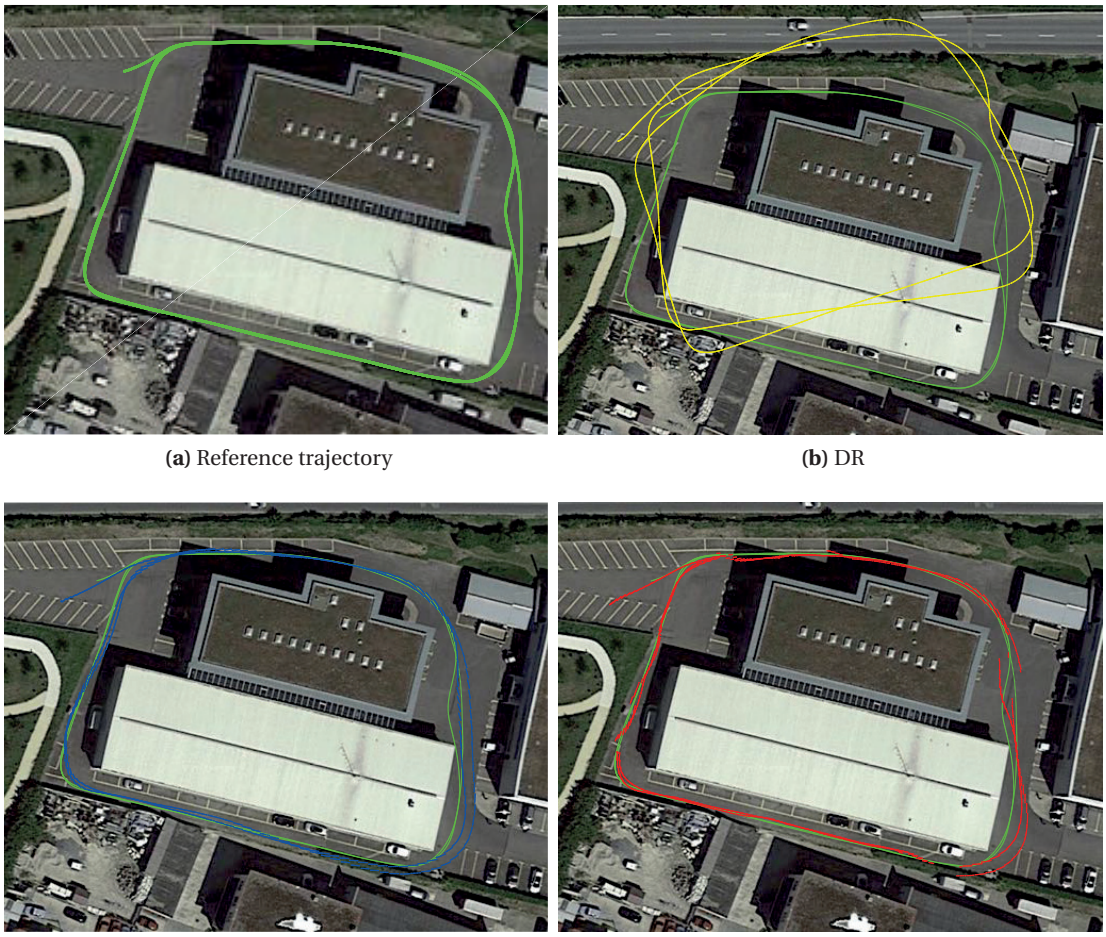
Algorithm	DR	DR+GPS	Matching
Mean error	7.10 m	0.80 m	0.52 m
Minimum error	0.003 m	0.002 m	0.000 m
Maximum error	22.11 m	3.47 m	2.46 m

trajectories. Finally, our matching algorithm produces the most accurate trajectory. Gating of GPS measurements is performed in the filter to prevent the delay in the signal from pulling the state estimate backwards and thus degrading the localization performance.

We put our localization algorithms to another test, this time while driving at lower speeds (approximately 6 km/h) around our garage building. At lower driving speeds, the effect of the GPS signal delay is not very significant. The car starts at the north-west side of the building and drives twice around the building in the clockwise direction. The obtained trajectories are drawn in Figure 15.2. As the reference car drives around 10 m ahead of the ego car, the tracking algorithm looses track of it in curves. Thus, the trajectory obtained by the matching algorithm in curves is very similar to the trajectory of the DR+GPS algorithm. When the ego car arrives behind the corner and reestablishes the track of the reference car, the update created by the matching algorithm quickly pulls the estimate towards the real position, thereby causing the EKF estimate to jump (see Figure 15.2d). This is due to having a significantly smaller covariance of the matching update as compared to the GPS measurement covariance. We note that there is also a jump in the forward direction, that compensates for the delay in the GPS measurement mentioned above.

A quantitative analysis is performed, similarly to the higher speed case. By looking at Table 15.2 we observe that the performance is similar to the one obtained at higher speeds, with the exception of the DR+GPS filter, which yields more accurate results in the slower scenario, partially due to the smaller effect of the GPS signal delay at lower speeds. It is necessary to mention that the expressed error metric does not penalize for the longitudinal error that arises from the GPS delay. This is because our reference localization system is installed on the leading car and we do not have the reference position of the ego car at any given time. Instead, we need to look for the closes point on the reference trajectory within some time horizon.

In order to perform a more accurate comparison, a reference localization system should be installed on the same vehicle as the localization system that is being evaluated. In our case, it



(a) Reference trajectory **(b)** DR

(c) DR+GPS **(d)** Matching

Figure 15.2 – Localization trajectories when driving at speeds of around 6 km/h.

would mean having a reference localization system installed on each of our test vehicles.

Summary

This chapter introduces our localization framework. The backbone of the framework is an EKF, which provides a continuous global location estimate by fusing the data coming from proprioceptive and exteroceptive sensors. The framework includes a cooperative localization component, which comprises two algorithms for matching of tracked objects and aims at improving the accuracy of the ego vehicle localization with respect to the cooperative vehicle. We perform an evaluation of our approach in real-world scenarios using two test cars.

16 Cooperative Tracking under Realistic Localization Conditions

In Chapter 14, we demonstrated an approach for tracking of cars using Mobileye cameras. As mentioned, the camera distance estimates produced by these sensors suffer from a significant, non-uniform bias that needed to be corrected. In this chapter, we propose an alternative approach, which fuses the measurements from Mobileye cameras together with the LIDAR measurements. We perform tracking of pedestrians and cars from the moving ego vehicle. Therefore, we additionally need to integrate the localization approach presented in Chapter 15.

16.1 Integration of Measurements from a Mobileye Camera

Whilst doing additional experiments with the Mobileye camera, we noticed that its distance estimates are not very accurate. This is somewhat to be expected, given that the Mobileye device uses a single camera. We suspect that the distance is learned from the training data, by considering the size of the detected object. It was evident during our experiments that the camera produces different distance estimates for pedestrians of different height. In particular, we observed how the reported distance increased as the pedestrian was assuming a squat position, therefore appearing shorter in the camera's FOV. A similar problem was observed when detecting cars of different size. Moreover, the distance to partially occluded cars was estimated wrongly.

Instead of trying to calibrate the camera for the size of specific objects we track, in this chapter we propose to use the Mobileye camera as a complementary sensor for tracking, by exploiting the features that are not easily or reliably observable by LIDARs. Namely, when detecting objects in a LIDAR point cloud, one notable difficulty is discriminating pedestrians from other similarly sized objects (e.g., tree trunks). Vision sensors are known to be better suited for this problem. The same holds for detection of cars — while the Mobileye camera can detect cars (and larger vehicles) from the front or the back side, every segment in the LIDAR point cloud that has the rough dimensions of a car width or length can be classified as a car detection. Moreover, it is impossible to distinguish the front of the car from the back by looking at the

LIDAR point cloud only (at least when considering 2D and 2.5D LIDARs). On the other hand, a LIDAR provides a much more accurate distance estimate than a monocular camera, so it makes sense to exploit this property for tracking.

A PHD filter has a concept of birth intensity, which allows the filter to tracks objects that enter the sensing area (see Equations (7.14) and (8.3)). We compute the birth intensity from the Mobileye measurements set Z as a mixture of Gaussian distributions centered at the individual measurements z (given in Cartesian coordinates):

$$\gamma_k(\mathbf{x}) = \sum_{z \in Z} w\mathcal{N}(\mathbf{x}; m_z, P_z). \tag{16.1}$$

The diagonal of the covariance P_z is set to (σ_r, σ_ϕ) in polar coordinates and then rotated to Cartesian coordinates using the bearing angle of the measurement ϕ, to reflect the fact that the uncertainty is larger in the measurement range than in bearing. The car birth intensity additionally contains the orientation, which is also extracted from the Mobileye measurement.

Depending on the class of the detected object (pedestrian or car), a new component is added to the appropriate birth intensity (as will be seen in the next section, pedestrians and cars have different state models and are thus kept as separate RFSs). Other classes of objects (such as for example trucks or bicycles) are ignored. The presented method is very efficient against clutter measurements (created by the preprocessing algorithm, for instance originating from trees or pillars).

16.2 Modeling Objects

A pedestrian is modeled using a 4D representation, whose states are position x and y, and velocity v_x and v_y:

$$\mathbf{x}^{\text{ped}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T. \tag{16.2}$$

A car is modeled as a rectangle of fixed size, and a 5D kinematic model is used, the states being the center of the rectangle (x, y), car heading θ, car linear and rotational speed v and ω, respectively:

$$\mathbf{x}^{\text{car}} = \begin{bmatrix} x & y & \theta & v & \omega \end{bmatrix}^T. \tag{16.3}$$

Given that different state models are used for different object types, they are tracked independently. There exist separate PHD filters (and intensities) for pedestrians and for cars. Objects of different class are not merged or fused together.

16.2.1 Motion Model

To model the motion of a tracked object, Constant Velocity (CV) and CTRV models are used (see Equation (8.35) and the paper by Schubert et al. [89]). Given a posterior object state $\mathbf{x}_{k-1}$ at time $k-1$, the predicted state at time k is given by

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}) + b(\mathbf{u}_{k-1}) + g(\mathbf{x}_{k-1})\xi_{k-1}, \tag{16.4}$$

where $f(\cdot)$, $b(\cdot)$ and $g(\cdot)$ are nonlinear in general case. The input $\mathbf{u}_{k-1}$ relates to the movement of the sensor relative to the tracked object. It is the change in pose of the ego vehicle from the last measurement update to the new one, and is obtained from the localization EKF. Noise is modeled as acceleration, $\xi = [\ a \quad \alpha\ ]^\top$ being the acceleration vector and $g(\cdot)$ the process noise model that transforms the acceleration vector to the each state variable.

The orientation of a pedestrian cannot be reliably detected by any of the sensors we use. Moreover, the movement of a pedestrian is holonomic, so it makes sense to include the full velocity vector in its state. The motion model used for a **pedestrian** is a CV model.

The motion model used for a **car** depends on the magnitude of the angular speed due to the division in the CTRV model:

$$f^{\text{car}} = \begin{cases} f^{\text{car}}_{CTRV} & \text{if } \omega \geq 10^{-4}\text{rad/s} \\ f^{\text{ped}}_{CV} & \text{otherwise} \end{cases} \tag{16.5}$$

16.3 Experimental Evaluation

The evaluation is carried out using our two test vehicles. Two real-world scenarios are considered. In the first scenario (Scenario 2), two pedestrians are tracked from a moving vehicle, which is aided by an infrastructure sensor. In the second scenario (Scenario 3), cars driving in the opposite direction are tracked, while the two cooperative vehicles drive one behind another.

16.3.1 Tracking Pedestrians

In Scenario 2, the two intelligent cars start at the distance of around 35 m facing each other. One of the cars (the cooperative car) is parked and remains stationary throughout the experiment. It basically represents a sensor station deployed in the infrastructure (e.g., at a traffic light), endowed with two LIDARs, a Mobileye camera, and communication and computation equipment. The other car (the ego car) drives at the approximate speed of 6 km/h.

In the scenario, there are two pedestrians, who start off in the middle of the scene, and they walk to different sides of the road. Two two pedestrians are instructed to walk along the predefined trajectories, maintaining as constant speed as possible. Their ground truth trajectories

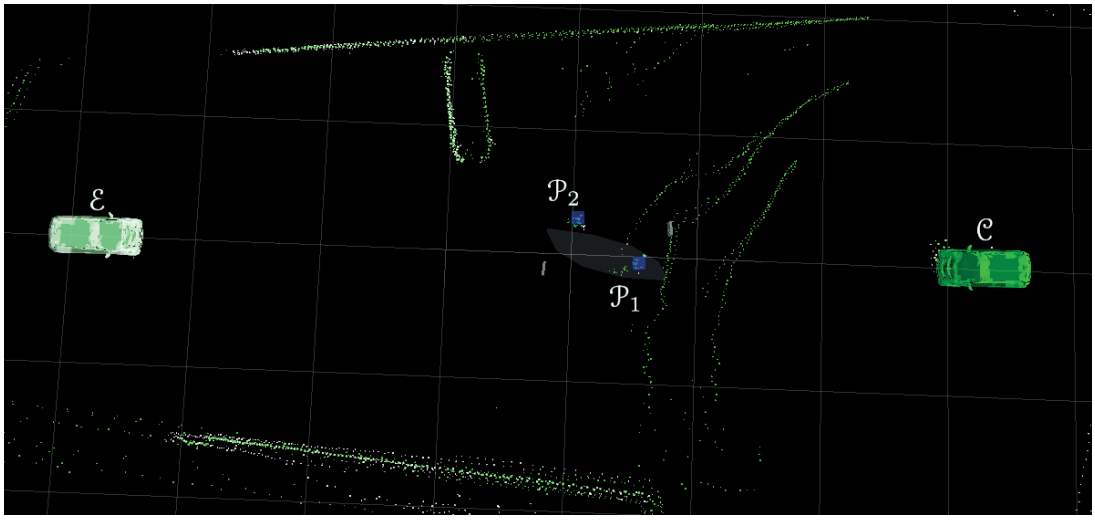
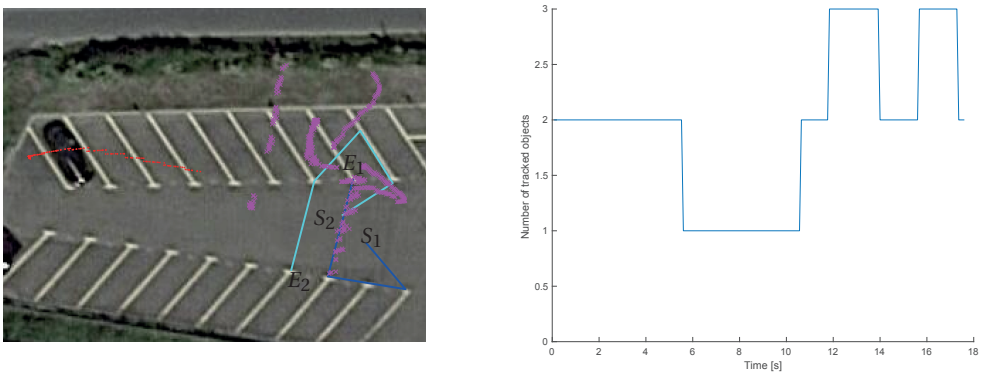


Figure 16.1 – A screenshot from `rviz` showing the beginning of Scenario 2a as seen from above. The moving (ego, $\mathcal{E}$) vehicle is on the left-hand side shown in white, while the stationary (cooperative, $\mathcal{C}$) vehicle is the green one on the right. Respective point clouds are color-coded in the same way. The two pedestrians $\mathcal{P}_1$ and $\mathcal{P}_2$ can be seen in the middle, with the output from the ego tracking filter shown using blue rectangular parallelepipeds. Observe the contours of a parked car in the top center of the image. Measurements from the ego Mobileye camera are the gray rectangles, and their respective birth component covariance ellipses, rotated using the bearing angle as explained in Section 16.1, are also drawn in gray. The grid size is 5 m. *Note: the entire LIDAR point cloud of the cooperative car is exceptionally communicated to the ego car for visualization purposes; in general, it is not communicated in order to minimize the required communication bandwidth and incurred communication delays.*

are extracted from satellite maps; markings on the parking lot visible in the satellite images are used as the anchoring points. Another car is parked on the left side of the road as seen from the ego vehicle, representing a barrier behind which pedestrians cannot be detected without cooperation.

A screenshot from `rviz` showing the scenario at time $t = 0$ s is given in Figure 16.1. We distinguish two variants of Scenario 2: in Scenario 2a, `car2` (the white C-ZERO with low-cost localization system) moves, while `car1` is parked; in Scenario 2b, `car1` (the green C-ZERO with high-end localization system) moves, while `car2` is parked.

Let us first take a look at the cooperative tracking performance when no accurate relative localization is available (Scenario 2a). As our `car2` is endowed with an inexpensive and inaccurate localization system, we move the `car2` while keeping the `car1` stationary. Figure 16.2 shows a trajectory of the `car2` as well as the tracks of the two pedestrians. First, we notice an offset in the `car2` localization, whose magnitude is a few meters. This results also in pedestrian tracks being located far away from the ground truth trajectories in the global frame. Moreover, the large error in relative localization between the two cooperative vehicles renders the cooperative fusion algorithm useless. In particular, the algorithm cannot track the occluded pedestrian due to the large localization uncertainty. Moreover, an additional (third) target appears at two



(a) **(b)**

Figure 16.2 – Tracking pedestrians from the moving `car2` cooperating with the stationary `car1` (Scenario 2a). The `car2` uses DR+GPS algorithm for localization and does not fuse in any corrections based on the object matching algorithm. (a) Satellite view of the `car2` trajectory (red), tracked pedestrians (magenta) and their ground truth trajectories (blue and cyan). S_1, E_1, S_2, and E_2 represent the start and end points on the trajectories of the first and the second pedestrian, respectively. (b) Cardinality plot over time.

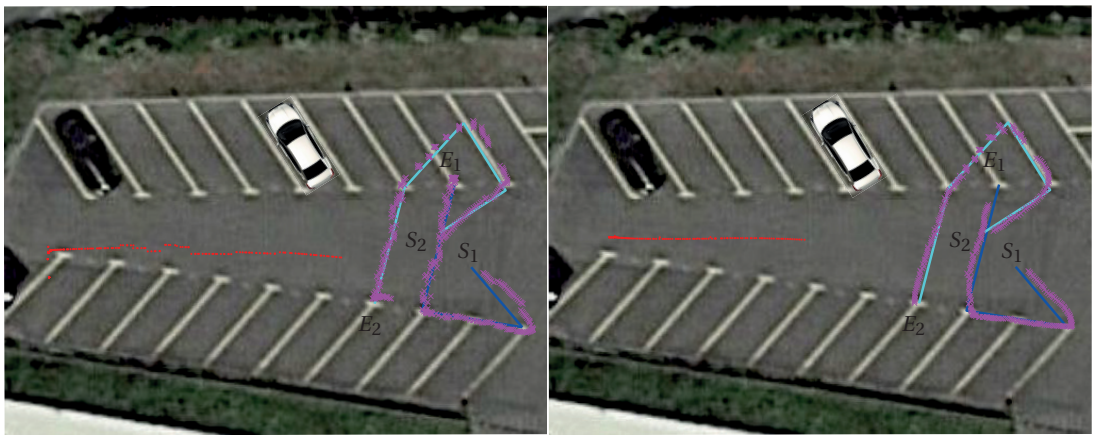
occasions (see in particular Figure 16.2b), which is the result of the GM components from the two cars, belonging to the same pedestrian, being very far away from each other in the global frame.

Let us now provide the conditions necessary for the cooperative fusion algorithm to work properly.

In the first batch of 10 runs, `car2` plays the ego vehicle and `car1` is parked on the other side (Scenario 2a). As seen above, the ego vehicle cannot localize itself with enough accuracy to perform cooperative fusion. However, by augmenting its localization capabilities with the object-matching algorithm presented in Chapter 15.4. We will demonstrate that it is possible to achieve a satisfactory localization performance and therefore proper working of the cooperative perception algorithm. The object-matching algorithm produces a measurement for the EKF update only if both pedestrians are tracked by both cars (prior to cooperative fusion). When less than two pedestrians are detectable by any of the cars, the EKF simply produces its estimate based on the DR and GPS sensors.

In the second 10 runs, the two sensing cars switch their roles, `car1` becoming the ego vehicle (Scenario 2b). In this case, `car1` can be localized very accurately globally, while the location of the stationary `car2` is manually determined in advance, the relative localization error between the two cars is small.

Selected runs from the datasets associated with Scenario 2a and 2b are depicted in Figure 16.3. In Scenario 2a, pedestrians are now successfully tracked. We notice that the tracks of the



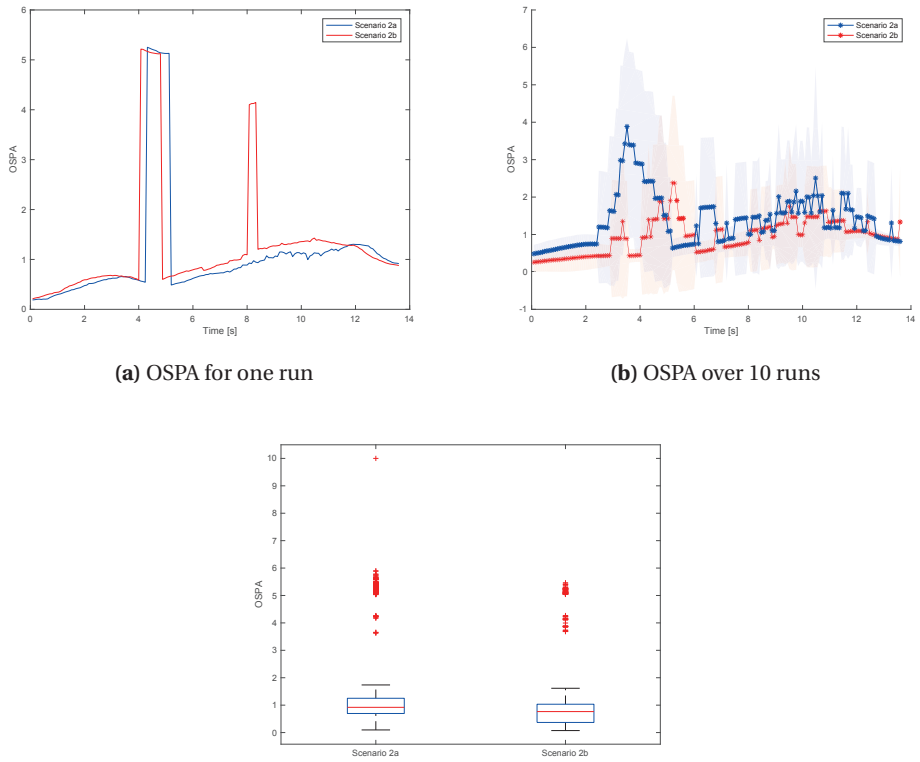
(a) Scenario 2a (ego vehicle is `car2`) **(b)** Scenario 2b (ego vehicle is `car1`)

Figure 16.3 – Tracking pedestrians from a moving ego vehicle, while the cooperative vehicle is parked facing the opposite direction. The trajectory of the ego vehicle is drawn with red dots. The tracked object trajectories are shown in purple, while the ground truth trajectories of the two pedestrians are drawn in blue and cyan color. S_1, E_1, S_2, and E_2 represent the start and end points on the trajectories of the first and the second pedestrian, respectively.

tracked pedestrians closely resemble the ground truth trajectories. In the top right part of the image, we see the trajectory of the occluded pedestrian. Thanks to the cooperation, it is successfully tracked. After a sharp change in the movement direction, we see that some time is required for the direction of the velocity vector to be correctly estimated, which results in predictions in wrong direction. In Scenario 2b, we first notice the smooth trajectory of the ego vehicle obtained by the Applanix localization system in Figure 16.3b. Pedestrian trajectories are smoother than in Scenario 2a, thanks to the smoother localization of the ego car.

Figure 16.4 shows the performance of the cooperative tracking using the OSPA metric. OSPA for a single run is given in Figure 16.4a. The first spike around $t = 4$ s, characterized by a duration of around 1 s, corresponds to the cardinality mismatch when one of the two pedestrians enters the occluded area of the ego car's FOV. The PHD filter running on the ego car keeps a component corresponding to a pedestrian, albeit with a low weight because it is not being detected by its sensors. Its weight is being further reduced with each sensor update. This low-weighted component influences the weight of the fused component. An aggravating circumstance is that the parked car is not being tracked in this scenario, so it is impossible to model the occlusion by it. The second spike corresponds to the cardinality mismatch when the pedestrian re-enters the FOV of the ego car. Accidentally, in the selected two runs, the performance of the tracking was slightly better when the `car2` took the role of the ego car.

On average, the tracking error over 10 runs is lower in Scenario 2b, as shown in Figures 16.4b and 16.4c. This is to be expected due to `car1`'s ability to localize more accurately. Aggregated performance over 10 runs for each of the two batches is shown in Figure 16.4b, whereas the Whisker's box plots are given in Figure 16.4c. Remarkably, there is no statistically significant



(a) OSPA for one run

(b) OSPA over 10 runs

(c) OSPA box plot over 10 runs

Figure 16.4 – Performance of C-GM-PHD filter in the pedestrian scenario expressed using the OSPA metric. Runs corresponding to the `car2` taking the role of the ego car are shown in blue (Scenario 2a), while in red are the runs when `car1` moves (Scenario 2b).

difference between the performance of the C-GM-PHD filter when run on each of test cars, an outcome that demonstrates the ability of our cooperative localization algorithm presented in Chapter 15 to compensate for on-board inaccurate localization equipment.

16.3.2 Tracking Cars

In Scenario 3, the two cooperative vehicles drive one behind the other on a relatively straight road segment of approximately 150 m in length, `car2` being the rear, ego vehicle, and `car1` being the front, cooperative vehicle. They start on the south end of the road. Two legacy vehicles drive in the opposite direction, coming from the north. All four cars are human driven, starting from a still position and accelerating to about 30 $^{km}/_{h}$. The cars start from approximately the same positions and the rear cars try to keep the same distance to the leading cars.

We performed 14 experimental runs. A screenshot from the ROS visualization tool `rviz`

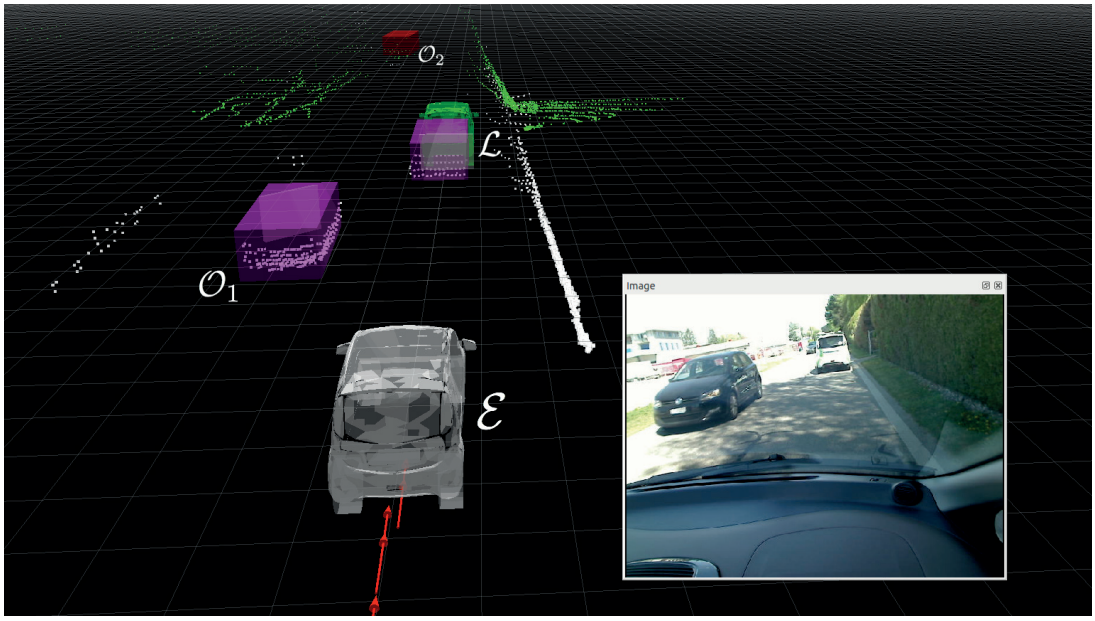


Figure 16.5 – A screenshot from the ROS visualization tool `rviz` during an experimental run. The ego car is shown in white. A point cloud obtained by the ego vehicle's LIDAR sensors is also visualized in white. The pose of the cooperative car, shared through the wireless communication, is displayed in green, and so is the point cloud belonging to the cooperative car. Objects tracked directly by the ego car are shown in purple. External objects fused by the ego car are displayed with red color. The trajectory of the ego car in the last few iterations is shown with red arrows. The measurements of the Mobileye cameras are shown as gray rectangles, while gray ellipses represent the uncertainty we used in our birth model. *Note: the point cloud of the cooperative car is exceptionally communicated to the ego car for visualization purposes; in general, it is not communicated in order to minimize the required communication bandwidth and incurred communication delays.*

is shown in Figure 16.5. The effectiveness of our cooperative localization algorithm can be observed from the alignment of the two point clouds (for instance, look along the right edge).

In Figure 16.6 we plot tracked trajectories of the tracked objects (two oncoming cars $\mathcal{O}_1$ and $\mathcal{O}_2$ and the leading vehicle $\mathcal{L}$) from the tracking car perspective (ego vehicle $\mathcal{E}$). Figure 16.6a shows trajectories belonging to 14 runs of the GM-PHD filter (no cooperative fusion employed) on a satellite map. The localization of the ego vehicle is performed using DR only, and all trajectories are rotated using the correct heading of the ego vehicle. GPS was not used because the initial orientation (coming from the Yocto3d compass) is very unreliable, and the car needs to travel a few tens of meters before the filter can correct the heading based on the received GPS measurements (which contain no heading). We observe drift in the DR localization.

On the other hand, with the aid of cooperative localization, we can obtain more accurate tracking in the global frame, proving again that cooperative localization can be regarded as a key enabling factor for cooperative perception. Trajectories of tracked objects over 14 runs are shown in Figure 16.6b. We also observe that objects are tracked over a longer distance. At large distances, the oncoming cars are detected slightly too much on the side, which places them

Table 16.1 – Maximum tracking distance for oncoming cars.

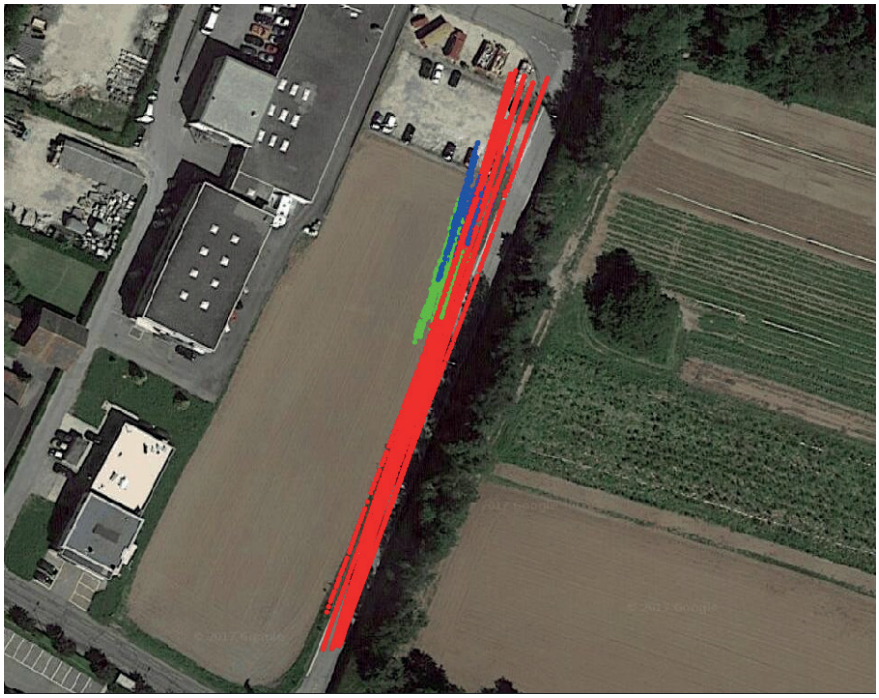
	μ_{car_1}	σ_{car_1}	μ_{car_2}	σ_{car_2}
GM-PHD	35.22 m	7.32 m	31.15 m	6.88 m
C-GM-PHD	58.65 m	10.18 m	52.04 m	8.59 m

off the road. We believe this is due to the too small silhouette as perceived from the LIDARs at large distances: the center of a detected object cannot be extracted accurately based only on a few points of laser reflection.

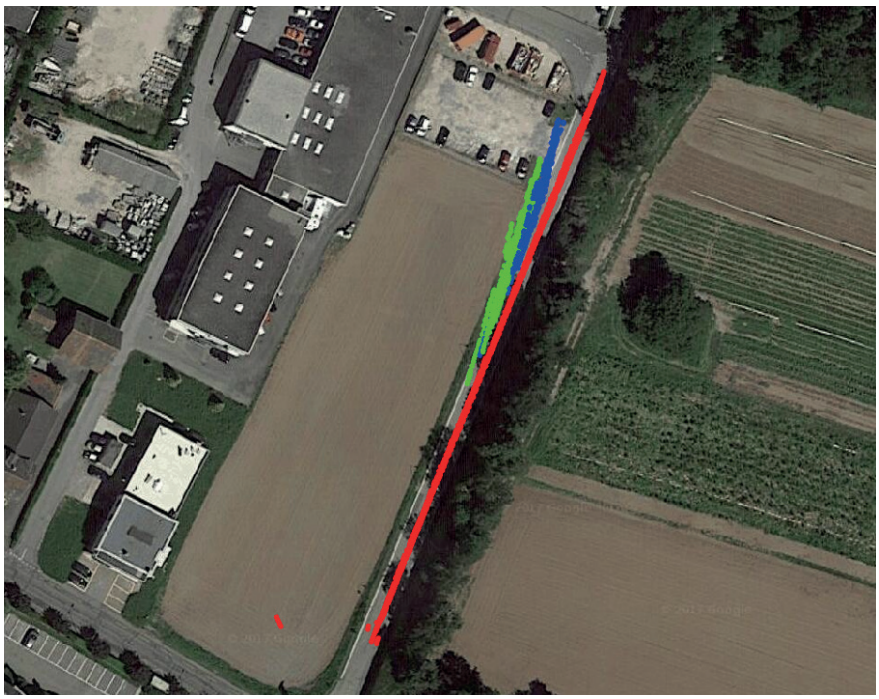
We do not have ground truth trajectories for this scenario as only the leading vehicle is equipped with the Applanix system (neither the legacy oncoming cars nor the ego vehicle do have such high-end localization system). It is therefore not possible to calculate the full OSPA metric. Table 16.1 shows the mean μ and standard deviation σ over 14 runs of the maximum distance at which oncoming cars are tracked. A clear added-value of the cooperative filter can be seen. We notice that the first oncoming car can on average be tracked at larger distances than the second car. We believe this is due to the second oncoming car being partially occluded by the first car. Recall that a car needs to be detected by both the LIDAR and the Mobileye camera in order for a new object to be born in the filter.

Summary

This chapter concludes our experimental effort using real vehicles. We introduced a new method of fusing LIDAR and Mobileye sensors in the framework of the PHD filter, which exploits the best features offered by each of the sensing modalities. By building on top of the previously proposed cooperative localization method, we managed to deploy the cooperative perception algorithms on moving vehicles. We validated our approach by tracking pedestrians in a V2I scenario, and by tracking cars in a V2V scenario.



(a) Without cooperation



(b) With cooperation

Figure 16.6 – Tracking cars from a moving vehicle. The leading car track is shown in red, the front oncoming car in green and the rear oncoming car in blue. When there is no cooperation between vehicles, the localization of the tracking vehicle is performed using the DR algorithm. In the case of cooperation, localization is enhanced by the matching algorithm.

17 Conclusion

SIMULATED environments only partially capture all the relevant properties of the real world. Therefore, system validation on real-world data is of extreme importance. There is often a significant delta incurred with transition from simulation to reality, which requires a substantial engineering (and sometimes research) effort to be properly addressed. Some of the issues encountered have to do with time synchronization and delays, real-time constraints and limitations in computational speed, as well as inaccurate sensors whose error models may not be characterized. The goal of Part IV was to perform experimental validation of our cooperative perception algorithms using real vehicles and automotive-grade sensors.

We started by presenting our software framework relying on ROS in Chapter 13. We made our implementation of the C-GM-PHD filter compatible with ROS. We designed a flexible and distributed software architecture, which allowed us to run the same filter code in Webots and on real vehicles. We made the transition as simple as disconnecting the nodes that publish the simulated sensory data, and connecting the nodes that interface the real sensors.

We designed three evaluation scenarios:

- In Scenario 1, the two cooperative vehicles were positioned one behind the other on a straight road, while two legacy cars drove in the opposite direction. For the first time, we used Mobileye cameras as perception sensors.
- Scenario 2 was a representative of a V2I scenario. While one cooperative car was moving, the other one was stationary and had thus resembled an infrastructure sensor. Two pedestrians were tracked in a cooperative fashion using LIDARs and Mobileye cameras.
- Scenario 3 was similar to Scenario 1, however the tracking algorithm relied on both LIDARs and Mobileye cameras, and the sensing vehicles were moving.

In Scenario 1, we assumed that both cooperative vehicles can benefit of accurate localization. While this was easy to achieve in simulation using accurate sensors (Scenario 1a), in real-

world settings we parked the two cooperative vehicles (Scenario 1b); the vehicles were kept stationary throughout the experiments, and their relative localization was resolved by the means of a LIDAR point cloud. We relaxed this assumption in Scenarios 2 and 3, in which the experiments were carried out under realistic localization conditions. In Chapter 15, we developed a cooperative localization method that results in an increased relative localization accuracy between the cooperative vehicles. This method enabled us to move the cooperative vehicles in the experiments, as shown in Chapter 16. Additionally, we performed a couple of algorithmic enhancements, that made the algorithm perform better in real-world scenarios.

Overall, in Part IV we presented algorithmic enhancements, detailed implementation choices and performed experimental validation of our cooperative perception algorithms using real vehicles. We showed the effectiveness of our approach in different scenarios, where pedestrians and cars were successfully tracked in a cooperative fashion, bypassing the limitations otherwise imposed by limited range of individual sensors and occlusions.

17.1 Discussion

Our experimental platforms are relatively complex, and working with them brings a number of challenges. On the level of a single vehicle, one of the challenges is to interface all sensors and integrate them in a single framework. Once the data from real sensors are obtained, a few interesting observations can be made. The sensors are not perfectly synchronized as it was the case in simulation, and their delays are variable. The software framework needs to be able to account for variable delays, and process the data in real time. Notably, more complex and uncontrollable environments in which sensors operate represent another challenge. While buildings, trees, or signposts are also available in Webots, some before unseen effects are produced by, for example, wire fences or bushes. These effects represent corner cases for the algorithms, which may need to be tuned to operate more robustly in real-world environments.

A few additional challenges emerge when working with multiple vehicles. First, the time synchronization is absolutely critical for multi-vehicle applications. No matter if some data is transmitted in real time, or the data is logged for post-processing — a common time reference is required. Perhaps the most common way to synchronize time is by using the GNSS time (the accuracy of GNSS time signal is ±10 ns), when supported by the equipment. Since we could not access the time pulse on the U-blox 5 GPS receiver, we decided to use a network synchronization tool for time synchronization across multiple computers. The requirement is that all computers are connected to the network, which was fulfilled in our case. Second, V2X communication is characterized by variable delays, and the wireless connection can drop if the distance between the vehicles becomes too large. This can pose a limitation on the scenario design. Third, accurate relative localization is essential for cooperative applications such as cooperative perception. As sensors operate in their local frames, a common reference is required for moving from one local coordinate frame to another.

Having mentioned the challenges above, we feel that having a realistic simulation tool is of

uttermost importance for research and algorithm prototyping in the field of intelligent vehicles. The effort we invested in the development and calibration of models for Webots enabled us to approach the problem in a gradual and systematic way, as we were able to experiment in controlled environments. Thanks to our software framework and the underlying, distributed and flexible architecture based on ROS, we could make a smooth transition from simulation to real cars. The algorithms developed in simulation required only a couple of enhancements before they started working properly under real-world conditions.

Conclusion Part V

18 Conclusion

OVER the course of this thesis, we examined the problem of cooperative perception. This endeavor was motivated by the need to achieve a safer and further optimized performance of intelligent vehicles. The ultimate goal of cooperative perception is to deliver larger coverage of the surrounding area and better situational awareness to individual vehicles. However, achieving such an outcome presents many scientific and technical challenges. These include questions on: (i) design and deployment of platforms and development of simulation tools, (ii) design of core cooperative perception algorithms, and (iii) deployment and testing of algorithms on real vehicles. In this manuscript, we presented our work on each of these topics. Overall, we proposed an end-to-end framework for cooperative perception. We hope to have provided enough evidence of the applicability of our framework in both simulated and real-world scenarios.

Our work on platforms and simulation tools, detailed in Part II of this manuscript, provided the following contributions:

- We designed our platform consisting of two vehicles, such that it successfully supported our experimental campaign, targeted towards testing of cooperative perception algorithms. We deployed on it a set of perception and localization sensors, computers and communication equipment.

- We reproduced our experimental platform in the high-fidelity simulator Webots. This included developing models of cars and sensors, and calibrating them using the real hardware. By reducing the simulation-to-reality gap, we could develop algorithms in controlled simulated environments and then smoothly move them to real vehicles. Moreover, we could design the scenarios and verify their validity in simulation before setting them up in reality.

Our work on cooperative perception used the PHD filter as a baseline for multi-object tracking. Our contributions on this topic, detailed in Part III, are summarized as follows:

- We proposed two algorithms for cooperative perception. The first one used a GM approximation of the PHD intensity, whereas the second one used an SMC approximation. Both algorithms contain novel methods for the fusion of non-overlapping FOVs (i.e., the complementary fusion) — thus providing solutions for robust tracking of multiple objects in sensing sets whose overlap is dynamically changing, because the sensors are mounted on non-coordinated mobile vehicles.

- Building upon the contributions mentioned above, we designed an overtaking assistance system based on cooperative perception. It serves as a proof of concept and demonstrates the power of cooperative perception algorithms in an application with constrained FOV and limited sensing range.

Finally, in Part IV, we performed multiple algorithmic enhancements and carried out extensive experimentation using real vehicles. Our contributions on this last topic are as follows:

- We designed a flexible, distributed software framework, which relies on ROS. This software framework allowed us to use the same implementation of cooperative perception algorithms in simulation and on real vehicles. It provides a natural and efficient way to work with multiple vehicles, therefore natively tackling the cooperation problem.

- We carried out an optimization of fusion weight in the C-GM-PHD filter. The result was a more robust algorithmic performance, especially in terms of estimated cardinality of the fused PHD intensity.

- We proposed an unconventional, innovative method for the fusion of data gathered by LIDARs and Mobileye cameras through the birth intensity of the PHD filter. In this way, for each of the sensors, we exploited its most valuable properties (ranging for the LIDAR and object identification for the Mobileye camera). Thanks to this improvement, we could run our algorithm in more complex, real-world environments, as the Mobileye camera only encouraged birthing of objects which correspond to cars and pedestrians (and not other elements in the environment such as wall segments).

- We proposed a cooperative localization framework. The framework consisted of an EKF localization filter for the fusion of signals from the GNSS, speedometer and inertial sensors, as well as of two algorithms for object-matching. The algorithms exploited objects detected by both cooperative vehicles to compute an accurate relative pose between them. This pose was integrated in the aforementioned EKF filter in the form of a correction.

- The end-to-end framework for cooperative perception was validated using two real Citroën C-ZERO cars. We designed scenarios representative for V2I and V2V communication. We tested our framework while cooperatively tracking pedestrians and cars in real time, using stationary and moving sensing vehicles.

18.1 Discussion and Outlook

While individual vehicles are becoming quite sophisticated in terms of technology and autonomy built into them, intrinsic limitations of single vehicle perception systems (limited FOV and LOS constraints) cannot be easily overcome at the individual level. Through cooperation, vehicles can achieve superior performance due to enhanced coverage and better situational awareness of the area surrounding them. Hence, further progress in the area of cooperation and coordination of vehicles is to be expected in the coming years.

The end-to-end framework for cooperative perception proposed in this thesis includes algorithms and software tools. Our software toolchain, spanning from a calibrated simulator to real vehicles, is a key enabler for working with multiple vehicles in an efficient way. While in this thesis we only showed one possible application of our framework, its applicability spans well beyond overtaking recommendation systems: it can be useful in any situation where single vehicle perception is constrained, preventing accidents, contributing to higher traffic fluidity, and reducing fuel consumption. Thus, our framework can be regarded as a stepping stone towards more complex, multi-vehicle automated systems.

The proposed cooperative perception framework is general and agnostic about the underlying sensing technology, as the tracking and fusion are performed at the object level. Our choice of exteroceptive sensing modalities represents a minimally rich set of sensors that allows us to fulfill one of our objectives: perform both intra-vehicle and inter-vehicle sensor fusion. We want to highlight the fact that our framework easily supports deploying an additional sensor modality on our platform, or using a different set of exteroceptive sensors. Notably, a radar sensor is interesting for at least three reasons: (i) it is currently more affordable than a LIDAR, (ii) it has a superior penetration capability through any type of weather condition, and (iii) it can additionally measure the velocity of objects using the Doppler effect — a property not directly measurable by neither a LIDAR nor a passive camera. Instant velocity information is beneficial both in safety-critical situations, as well as for discriminating moving from stationary objects. However, the drawbacks of radars typically used in the automotive domain are their lower angular resolution and narrower FOV.

Our work constitutes a first step towards cooperative perception using PHD filters. While we did not obtain any statistically significant improvement in the OSPA performance in the overlapping FOV, mainly due to the conservative assumption introduced by the CI method, we demonstrated clear improvements when other metrics are considered (e.g., increased safety due to extended coverage, mitigated occlusions due to a second perspective). These improvements could have a positive effect on the intelligent vehicles' control systems as well, as vehicles could plan their trajectories over a longer time horizon, or could perform certain maneuvers at higher speeds (due to the extended coverage or less occlusions). Besides localization, the main challenges of using cooperative information for control are related to its correctness and liability in case of accidents. From this perspective, infrastructural sensors are definitely appealing as a starting point. Indeed, since the authorities can constrain the

physical access to them, infrastructural sensors are seemingly less prone to adversaries than the systems installed on moving vehicles.

We can foresee any follow-up of our work that aims to further enhance the performance of our methods. Different filters can easily be integrated in our flexible software framework. At this point, we would like to remind the reader that the PHD filter does not yield object labels (as detailed in Chapter 7). In other words, there is no correlation between spatial variables of objects over time. However, both the tracking from a single vehicle and the cooperative fusion could benefit from labels, as it would be possible to prevent merging or fusion of objects with different labels. To this end, multi-object filters based on the multi-Bernoulli filters could be especially effective (though at the cost of a higher computational complexity), as they approximate the multi-object posterior using the parameters of a multi-Bernoulli distribution. For example, a Labeled Multi-Bernoulli (LMB) filter [144] estimates for each object its probability of existence, its state probability density, and its label. When applied to LMB posteriors, the GCI-rule leads to separately combining the densities of Bernoulli components (possible objects) with the same label. It therefore prevents combining the densities of components which do not originate from the same object. In an attempt to achieve a superior performance in the overlapped FOV than the one offered by CI, other suboptimal fusion methods could be considered (e.g., split covariance intersection or bounded covariance inflation [145]). Alternatively, optimal sensor fusion methods could be used, in which careful bookkeeping and canceling out common information between the vehicles would be necessary [146].

Integration of digital maps in our framework would be another interesting option. While building maps in a distributed way is a research topic on its own, having additional contextual information can be beneficial for our framework. For instance, roads or regions of interest can be located on a map, which can in turn facilitate identifying objects of interest in complex scenarios.

We recognize that this dissertation only performs experiments using one-way V2X communication between two vehicles. We wish to see our framework running on more than two vehicles using bi-directional communication, where the vehicles would form a sensor network (eventually together with infrastructure nodes). Another open question is how far (e.g., in hops) the information should be propagated in the sensor network, and the associated scalability issue. A common problem with fusion of information in sensor networks is overconfidence due to double counting of information. In theory, the CI method we use for fusion does not suffer from overconfidence. However, it would be interesting to see how the information traveling in cycles among vehicles affects the algorithm performance in practice. A possible solution remedying both problems could maintain two target sets, the first obtained from ego measurements, and the second containing fused information from cooperative vehicles. Vehicles could then communicate the first set only — thus limiting the propagation of information to a single hop and removing network loops. Once the algorithm is operational on multiple vehicles (and hence covers a potentially much larger area), the next step in achieving a multi-vehicle

cooperative system would be to perform optimization of planned trajectories on the level of local traffic. Thanks to our end-to-end framework, all these questions involving larger sets of vehicles could be studied first in simulation, therefore significantly reducing the cost of experimentation using real vehicles.

While climbing up the path of an increased vehicle autonomy, a required improvement is to cover 360° around the vehicle with perception. As far as cooperative perception on roads is considered, the ability to additionally sense behind the vehicle would be beneficial, as it would allow the cooperative vehicles to detect each other using on-board sensors. As far as localization sensors are concerned, it would be desirable to have an accurate localization system on `car2` as well. While lower-cost localization solutions may be preferred in the sale phase, high-end solutions are welcome during development, at least for evaluation purposes.

Last but not least, experimental scenarios could be enriched by additional elements such as pedestrian dummies and remote-controlled soft (balloon) cars. They would allow for safe and repeatable experiments, and would provide ground truth trajectories for evaluation purposes, if equipped with a differential GNSS device.

Glossary

ACC	Adaptive Cruise Control	5, 95, 99
C-GM-PHD	Cooperative GM-PHD	46, 57, 61, 65, 66, 69–73, 90, 93, 97–99, 101, 102, 105, 116, 123–125, 127, 147, 151, 158
C-SMC-PHD	Cooperative SMC-PHD	46, 75, 82–86, 88, 102
CAD	Computer-Aided Design	34, 121
CAH	Constant-Acceleration Heuristic	96
CAN bus	Control Area Network	24, 27, 29, 30, 35, 45, 131
CI	Covariance Intersection	48, 61, 63, 69, 70, 101, 102, 131, 159, 160
CPHD	Cardinalized Probability Hypothesis Density	46
CTRV	Constant Turn Rate and Velocity	63, 143
CV	Constant Velocity	143
DISAL	Distributed Intelligent Systems and Algorithms Laboratory	12, 18, 47
DOF	Degree Of Freedom	134
DR	Dead-Reckoning	131, 136–139, 145, 148, 150
EKF	Extended Kalman Filter	47, 112, 113, 129–134, 136, 139, 140, 143, 145, 158
EM	Expectation-Maximization	47
EMD	Exponential Mixture Density	49, 75, 78
FISST	Finite Set Statistics	52
FOV	Field of View	18, 23–25, 38, 39, 45, 46, 49, 53, 57, 63, 65–67, 69–72, 75, 78, 79, 82, 84–87, 89, 97, 98, 101, 102, 123–126, 131, 136, 141, 146, 158–160
FSM	Finite State Machine	91, 92, 95
GCI	Generalized Covariance Intersection	61, 63, 101, 126, 160

GM	Gaussian Mixture	47, 57–59, 61–63, 75, 87, 101, 112, 118, 145, 158
GM-CPHD	Gaussian Mixture CPHD	49
GM-PHD	Gaussian Mixture PHD	45–48, 53, 57, 60, 61, 68, 71–73, 75, 118, 123, 125, 148
GMM	Gaussian Mixture Model	48, 62, 131
GNN	Global Nearest Neighbor	46
GNSS	Global Navigation Satellite System	5, 26, 27, 29, 37, 38, 66, 72, 83, 85, 87, 97, 105, 109, 120, 121, 129–131, 133, 135, 136, 152, 158, 161
GPS	Global Positioning System	26, 29, 30, 41, 112, 113, 137–139, 145, 148, 152
HAL	Hardware Abstraction Layer	111
HMI	Human Machine Interface	5
ICP	Iterative Closest Point	131
IDM	Intelligent Driver Model	95, 96
IMU	Inertial Measurement Unit	26, 27, 29, 30, 111, 120
IP	Internet Protocol	115
JPDA	Joint Probabilistic Data Association	46
KDE	Kernel Density Estimation	81, 87
KF	Kalman Filter	50, 52, 53, 84, 120, 129, 130
LIDAR	Light Detection and Ranging	5, 11, 18, 23–25, 28–31, 38–42, 45, 47, 53–56, 63, 65, 66, 68, 69, 79, 83–87, 90, 97, 98, 105–107, 110, 112, 113, 117, 120–122, 125, 127, 129, 130, 134, 141–144, 148, 149, 151, 152, 158, 159
LMB	Labeled Multi-Bernoulli	160
LOS	Line-Of-Sight	45, 101, 159
MSE	Mean Square Error	135
NLOS	Non-Line-Of-Sight	89
OBD-II	On-board Diagnostics	27, 30
OSPA	Optimal Sub-Pattern Assignment	69, 70, 72, 73, 86, 88, 123–126, 146, 147, 159
PDF	Probability Density Function	49, 50
PF	Particle Filter	50, 130
PHD	Probability Hypothesis Density	17, 18, 46–49, 52, 57, 70, 72, 75, 78, 79, 81, 101, 107, 110, 112, 123, 142, 146, 157–160
PI	Proportional-Integral	95
PID	Proportional-Integral-Differential	97
RANSAC	Random Sample Consensus	131

RD	Rényi Divergence	80, 81, 102
RFS	Random Finite Set	46, 48–52, 59, 142
ROS	Robot Operating System	105, 109–111, 113–117, 124, 127, 132, 148, 151, 153, 158
RTK	Real Time Kinematic	27, 137
SD	Standard Deviation	123
SLAM	Simultaneous Localization and Mapping	130, 131
SMC	Sequential Monte Carlo	46, 47, 50, 75, 78, 81, 101, 102, 158
SMC-CPHD	Sequential Monte Carlo CPHD	49
SMC-PHD	Sequential Monte Carlo PHD	46–49, 75, 78, 87
SSD	Solid-state Drive	27
SVD	Singular Value Decomposition	135, 136
UDP	User Datagram Protocol	113, 115
UKF	Unscented Kalman Filter	47, 58, 82–84, 87
USB	Universal Serial Bus	26
UTM	Universal Transverse Mercator	26, 112, 113, 133, 135, 136
V2I	Vehicle-to-Infrastructure	9, 106, 114–116, 149, 151, 158
V2V	Vehicle-to-Vehicle	9–12, 91, 106, 114–116, 149, 158
V2X	Vehicle-to-Everything	9–11, 13, 105, 114–116, 152, 160
VANET	Vehicular Ad-Hoc Network	9, 90
WLAN	Wireless Local Area Network	114

Bibliography

[1] U.S. Department of Transportation, '2015 motor vehicle crashes: overview', National Highway Traffic Safety Administration, Research Note DOT HS 812 318, Aug. 2016. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812318 (visited on 16/05/2017) (cit. on pp. 3, 6).

[2] European Comission, 'EU road fatalities', Directorate General for Mobility and Transport, Annual Report, Feb. 2016. [Online]. Available: http://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/observatory/trends_figures.pdf (visited on 03/05/2017) (cit. on p. 3).

[3] 'Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems', SAE International, Standard J3016 201401, 2014 (cit. on pp. 3, 4).

[4] *Citroën, Alerte de franchissement involontaire de ligne (lane departure warning system)*. [Online]. Available: http://www.citroen.ch/fr/univers-citroen/technologies/afil.html (visited on 19/05/2017) (cit. on p. 4).

[5] A. Bartels, M.-M. Meinecke and S. Steinmeyer, 'Lane change assistance', in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, H. Winner, S. Hakuli, F. Lotz and C. Singer, Eds. Springer International Publishing, 2016, pp. 1235–1257. DOI: 10.1007/978-3-319-12352-3_50 (cit. on p. 4).

[6] A. R. A. Van der Horst, *A time based analysis of road user behaviour in normal and critical encounters*, HS-041 255. TNO Institute for Perception, 1990 (cit. on p. 4).

[7] R. Kiefer, M. Cassar, C. Flannagan, D. LeBlanc, M. Palmer, R. Deering and M. Shulman, 'Refining the CAMP crash alert timing approach by examining "last second"braking and lane change maneuvers under various kinematic conditions', *Accident Investigation Quarterly*, no. 39, 2004 (cit. on p. 4).

[8] R. G. Fuller, 'Determinants of time headway adopted by truck drivers', *Ergonomics*, vol. 24, no. 6, pp. 463–474, 1981 (cit. on p. 4).

[9] Y. Zhang, E. Antonsson and K. Grote, 'A new threat assessment measure for collision avoidance systems', in *IEEE Intelligent Transportation Systems Conference*, 2006, pp. 968–975. DOI: 10.1109/ITSC.2006.1706870 (cit. on p. 4).

[10] R Keifer, D LeBlanc, M Palmer, J Salinger, R Deering and M Shulman, 'Development and validation of functional definitions and evaluation procedures for collision warning/avoidance systems', *US Dept. Transportation, Washington, DC, USA, Final Rep. DOT HS*, vol. 808, p. 964, 1999 (cit. on p. 4).

[11] A. Cabrera, S. Gowal and A. Martinoli, 'A new collision warning system for lead vehicles in rear-end collisions', in *IEEE Intelligent Vehicles Symposium*, 2012, pp. 674–679 (cit. on p. 5).

[12] *Citroën, Adaptive cruise control*. [Online]. Available: http://www.citroen.co.uk/about-citroen/technology/active-cruise-control (visited on 19/05/2017) (cit. on p. 5).

[13] H. Schittenhelm, 'Advanced brake assist–real world effectiveness of current implementations and next generation enlargements by Mercedes-Benz', in *International Technical Conference on the Enhanced Safety of Vehicles*, Seoul, South Korea, 2013 (cit. on p. 5).

[14] T. Pasenau, T. Sauer and J. Ebeling, 'Aktive Geschwindigkeitsregelung mit Stop&Go-Funktion: im BMW 5er und 6er (active cruise control with Stop&Go funtion in the BMW 5- and 6-series)', *ATZ - Automobiltechnische Zeitschrift*, vol. 109, no. 10, pp. 900–909, 2007. DOI: 10.1007/BF03221920 (cit. on p. 5).

[15] H. Gotzig, 'Parking assistance', in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, H. Winner, S. Hakuli, F. Lotz and C. Singer, Eds. Springer International Publishing, 2016, pp. 1077–1092. DOI: 10.1007/978-3-319-12352-3_45 (cit. on p. 5).

[16] S. Habenicht, 'Entwicklung und Evaluation eines manöverbasierten Fahrstreifenwechselassistenten (development and evaluation of a maneuver-based lane change assistance)', PhD thesis, Technical University of Darmstadt, 2012 (cit. on p. 5).

[17] M. Mages, F. Klanner and A. Stoff, 'Intersection assistance', in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, H. Winner, S. Hakuli, F. Lotz and C. Singer, Eds. Springer International Publishing, 2016, pp. 1259–1286. DOI: 10.1007/978-3-319-12352-3_51 (cit. on p. 5).

[18] Daimler AG, 'The driving assistance systems: helpers in the background', press release, 2013. [Online]. Available: http://media.daimler.com/marsMediaSite/en/instance/ko/The-driving-assistance-systems-Helpers-in-the-background.xhtml?oid=9904305 (visited on 19/05/2017) (cit. on p. 5).

[19] D. Huttenlocher, 'Cornell racing team and Velodyne's lidar sensor', Robots Podcast Episode 1, 2008 (cit. on p. 5).

[20] S. Thrun, 'Programming a robotic car', Udacity CS373: Artificial Intelligence Unit 7, Test Drive, 2012 (cit. on p. 5).

[21] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt and S. Thrun, 'Junior: the Stanford entry in the Urban Challenge', *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008. DOI: 10.1002/rob.20258 (cit. on p. 5).

[22] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar and B. Litkouhi, 'Towards a viable autonomous driving research platform', in *IEEE Intelligent Vehicles Symposium*, 2013, pp. 763–770. DOI: 10.1109/IVS.2013.6629559 (cit. on p. 5).

[23] F. Kunz, D. Nuss, J. Wiest, H. Deusch, S. Reuter, F. Gritschneder, A. Scheel, M. Stübler, M. Bach, P. Hatzelmann, C. Wild and K. Dietmayer, 'Autonomous driving at Ulm University: a modular, robust, and sensor-independent fusion approach', in *IEEE Intelligent Vehicles Symposium*, 2015, pp. 666–673. DOI: 10.1109/IVS.2015.7225761 (cit. on p. 5).

[24] R. Viereckl, D. Ahlemann, A. Koster and S. Jursch, 'Connected car study 2015', PwC strategy&, Tech. Rep., 2015. [Online]. Available: https://www.strategyand.pwc.com/media/file/Connected-Car-Study-2015.pdf (visited on 14/05/2017) (cit. on p. 9).

[25] 'Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 april 2015 concerning type-approval requirements for the deployment of the eCall in-vehicle system based on the 112 service and amending Directive 2007/46/EC', *Official Journal of the European Union*, vol. 58, no. L 123, pp. 77–89, 2015 (cit. on p. 10).

[26] U.S. Department of Transportation, 'Federal motor vehicle safety standards; V2V communications', National Highway Traffic Safety Administration, Tech. Rep. NHTSA-2016-0126, 2016. [Online]. Available: http://www.safercar.gov/v2v/pdf/V2V%20NPRM_Web_Version.pdf (visited on 15/05/2017) (cit. on p. 10).

[27] H. Li, 'Cooperative perception: application in the context of outdoor intelligent vehicle systems', PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2012 (cit. on p. 11).

[28] S. Andrews, 'Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications and cooperative driving', in *Handbook of Intelligent Vehicles*, A. Eskandarian, Ed. London: Springer London, 2012, pp. 1121–1144. DOI: 10.1007/978-0-85729-085-4_46 (cit. on p. 10).

[29] C. W. Hsu, M. K. KO, M. H. shih and S. C. Huang, 'An autonomous and car-following system via dsrc communication', *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.*, vol. 5, pp. 257–266, Apr. 2012. DOI: 10.4271/2012-01-0741. [Online]. Available: http://doi.org/10.4271/2012-01-0741 (cit. on p. 10).

[30] F. de Ponte Müller, 'Survey on ranging sensors and cooperative techniques for relative positioning of vehicles', *Sensors*, vol. 17, no. 2, p. 27, 2017. DOI: 10.3390/s17020271 (cit. on p. 10).

[31] M. Goldhammer, E. Strigel, D. Meissner, U. Brunsmann, K. Doll and K. Dietmayer, 'Cooperative multi sensor network for traffic safety applications at intersections', in *IEEE International Conference on Intelligent Transportation Systems*, 2012, pp. 1178–1183. DOI: 10.1109/ITSC.2012.6338672 (cit. on p. 11).

[32] N. E. Zoghby, V. Cherfaoui and T. Denoeux, 'Evidential distributed dynamic map for cooperative perception in VANets', in *IEEE Intelligent Vehicles Symposium*, 2014, pp. 1421–1426. DOI: 10.1109/IVS.2014.6856550 (cit. on p. 11).

[33] S. W. Kim, Z. J. Chong, B. Qin, X. Shen, Z. Cheng, W. Liu and M. H. Ang, 'Cooperative perception for autonomous vehicle control on the road: motivation and experimental results', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5059–5066. DOI: 10.1109/IROS.2013.6697088 (cit. on p. 11).

[34] S. W. Kim, W. Liu, M. H. Ang, E. Frazzoli and D. Rus, 'The impact of cooperative perception on decision making and planning of autonomous vehicles', *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 3, pp. 39–50, 2015. DOI: 10.1109/MITS.2015.2409883 (cit. on p. 11).

[35] A. Rauch, F. Klanner and K. Dietmayer, 'Analysis of V2X communication parameters for the development of a fusion architecture for cooperative perception systems', in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 685–690. DOI: 10.1109/IVS.2011.5940479 (cit. on p. 11).

[36] A. Rauch, F. Klanner, R. Rasshofer and K. Dietmayer, 'Car2X-based perception in a high-level fusion architecture for cooperative perception systems', in *IEEE Intelligent Vehicles Symposium*, 2012, pp. 270–275. DOI: 10.1109/IVS.2012.6232130 (cit. on pp. 11, 48).

[37] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W. B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam and N. McKeown, 'Automated vehicle control developments in the PATH program', *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991. DOI: 10.1109/25.69979 (cit. on p. 12).

[38] B. J. Harker, 'Promote-chauffeur ii & 5.8 GHz vehicle to vehicle communications system', *IET International Conference on Advanced Driver Assistance Systems*, pp. 81–85, 2001 (cit. on p. 12).

[39] C. Bergenhem, Q. Huang, A. Benmimoun and T. Robinson, 'Challenges of platooning on public motorways', in *World Congress on Intelligent Transport Systems*, Busan, Korea, 2010 (cit. on p. 12).

[40] E. Chan, P. Gilhead, P. Jelinek, P. Krejci and T. Robinson, 'Cooperative control of SARTRE automated platoon vehicles', in *Intelligent Transportation System World Congress*, 2012 (cit. on p. 12).

[41] A. Marjovi, **M. Vasic**, J. Lemaitre and A. Martinoli, 'Distributed graph-based convoy control for networked intelligent vehicles', in *IEEE Intelligent Vehicles Symposium*, IEEE, 2015, pp. 138–143 (cit. on pp. 12, 182).

[42] M. Obst, A. Marjovi, **M. Vasic**, I. Navarro, A. Martinoli, A. Amditis, P. Pantazopoulos, I. Llatser, A. de La Fortelle and X. Qian, 'Challenges for automated cooperative driving: the AutoNet2030 approach', in *Automated Driving: Safer and More Efficient Future Driving*, D. Watzenig and M. Horn, Eds. Springer International Publishing, 2017, pp. 561–570. DOI: 10.1007/978-3-319-31895-0_26 (cit. on pp. 12, 181).

[43] I. Navarro, F. Zimmermann, **M. Vasic** and A. Martinoli, 'Distributed graph-based control of convoys of heterogeneous vehicles using curvilinear road coordinates', in *IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2016, pp. 879–886 (cit. on pp. 12, 181).

[44] L. Makarem and D. Gillet, 'Fluent coordination of autonomous vehicles at intersections', in *IEEE International Conference on Systems, Man, and Cybernetics*, 2012, pp. 2557–2562. DOI: 10.1109/ICSMC.2012.6378130 (cit. on p. 12).

[45] E. Debada, L. Makarem and D. Gillet, 'Autonomous coordination of heterogeneous vehicles at roundabouts', in *IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 1489–1495. DOI: 10.1109/ITSC.2016.7795754 (cit. on p. 12).

[46] E. Debada, L. Makarem and D. Gillet, 'A virtual vehicle based coordination framework for autonomous vehicles in heterogeneous scenarios', in *IEEE International Conference on Vehicular Electronics and Safety*, Vienna, Austria, 2017 (cit. on p. 12).

[47] G. Silberg and R. Wallace, 'Self-driving cars: the next revolution', KPMG, CAR, 2012 (cit. on p. 16).

[48] **M. Vasic** and A. Billard, 'Safety issues in human-robot interactions', in *IEEE International Conference on Robotics and Automation*, 2013, pp. 197–204 (cit. on pp. 17, 182).

[49] **M. Vasic**, G. Jornod and A. Martinoli, 'From high fidelity simulation to real vehicles: a flexible software framework for networked intelligent vehicles', *in preparation*, 2017 (cit. on p. 17).

[50] **M. Vasic** and A. Martinoli, 'A collaborative sensor fusion algorithm for multi-object tracking using a Gaussian mixture probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2015, pp. 491–498 (cit. on pp. 18, 181).

[51] J. Gan, **M. Vasic** and A. Martinoli, 'Cooperative multiple dynamic object tracking on moving vehicles based on sequential Monte Carlo probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2016, pp. 2163–2170 (cit. on pp. 18, 181).

[52] **M. Vasic**, G. Lederrey, I. Navarro and A. Martinoli, 'An overtaking decision algorithm for networked intelligent vehicles based on cooperative perception', in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 1054–1059 (cit. on pp. 18, 181).

[53] **M. Vasic**, D. Mansolino and A. Martinoli, 'A system implementation and evaluation of a cooperative fusion and tracking algorithm based on a Gaussian mixture PHD filter', in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 4172–4179 (cit. on pp. 18, 181).

[54] **M. Vasic**, G. Jornod, J. Løje and A. Martinoli, 'An end-to-end framework for cooperative perception based on a GM-PHD filter', *in preparation*, 2017 (cit. on p. 18).

[55] E. Dagan, O. Mano, G. Stein and A. Shashua, 'Forward collision warning with a single camera', in *IEEE Intelligent Vehicles Symposium*, Jun. 2004, pp. 37–42. DOI: 10.1109/IVS.2004.1336352 (cit. on p. 24).

[56] O. Michel, 'Cyberbotics Ltd. Webots™: professional mobile robot simulation', *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004 (cit. on p. 33).

[57] S. Gowal, Y. Zhang and A. Martinoli, 'A realistic simulator for the design and evaluation of intelligent vehicles', in *IEEE International Conference on Intelligent Transportation Systems*, 2010, pp. 1039–1044. DOI: 10.1109/ITSC.2010.5625010 (cit. on p. 33).

[58] D. Mansolino, **M. Vasic**, O. Michel and O. Pajot, *RO2IVSim: a flexible and faithful simulator for bridging the gap between mobile robots and intelligent vehicles*. [Online]. Available: disal.epfl.ch/research/RO2IVSim (visited on 04/05/2017) (cit. on p. 33).

[59] D. Soudbakhsh and A. Eskandarian, 'Vehicle lateral and steering control', in *Handbook of Intelligent Vehicles*, Springer, 2012, pp. 209–232 (cit. on p. 33).

[60] Citroën Automobiles, 'Citroën C-ZERO owner's manual (in French)', Tech. Rep. 11.C0.0011, 2011. [Online]. Available: http://service.citroen.com/ddb/modeles/czero/czero_czero/ed02-11/fr_fr/contenu/AC-C-ZERO_02_2011_FR.pdf (visited on 05/05/2017) (cit. on p. 34).

[61] G. Thomaidis, K. Vassilis, P. Lytrivis, M. Tsogas, G. Karaseitanidis and A. Amditis, 'Target tracking and fusion in vehicular networks', in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 1080–1085. DOI: 10.1109/IVS.2011.5940535 (cit. on p. 45).

[62] M. Obst, L. Hobert and P. Reisdorf, 'Multi-sensor data fusion for checking plausibility of V2V communications by vision-based multiple-object tracking', in *IEEE Vehicular Networking Conference*, 2014, pp. 143–150. DOI: 10.1109/VNC.2014.7013333 (cit. on p. 45).

[63] R. MacLachlan and C. Mertz, 'Tracking of moving objects from a moving vehicle using a scanning laser rangefinder', in *IEEE Intelligent Transportation Systems Conference*, Sep. 2006, pp. 301–306 (cit. on pp. 46, 48, 53).

[64] Y. Bar-Shalom, *Tracking and Data Association*. San Diego, CA, USA: Academic Press Professional, Inc., 1987 (cit. on p. 46).

[65] R. P. S. Mahler, *Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, Inc., 2007 (cit. on pp. 46, 52).

[66] B.-N. Vo and W.-K. Ma, 'The Gaussian mixture probability hypothesis density filter', *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, Nov. 2006. DOI: 10.1109/TSP.2006.881190 (cit. on pp. 46, 47, 52, 57, 59).

[67] B. Ristic, D. Clark and B.-N. Vo, 'Improved SMC implementation of the PHD filter', in *13th Conference on Information Fusion*, 2010. DOI: 10.1109/ICIF.2010.5711922 (cit. on pp. 46, 47, 75, 76).

[68] K. Granström, C. Lundquist and U. Orguner, 'A Gaussian mixture PHD filter for extended target tracking', in *International Conference on Information Fusion*, 2010. DOI: 10.1109/ICIF.2010.5711885 (cit. on p. 47).

[69] K. Granström, C. Lundquist and O. Orguner, 'Extended target tracking using a Gaussian-mixture PHD filter', *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 3268–3286, 2012. DOI: 10.1109/TAES.2012.6324703 (cit. on p. 47).

[70] K. Granström and C. Lundquist, 'On the use of multiple measurement models for extended target tracking', in *International Conference on Information Fusion*, 2013, pp. 1534–1541 (cit. on p. 47).

[71] K. Granström, C. Lundquist and U. Orguner, 'Tracking rectangular and elliptical extended targets using laser measurements', in *International Conference on Information Fusion*, 2011, pp. 592–599 (cit. on p. 47).

[72] A. Swain and D. Clark, 'The PHD filter for extended target tracking with estimable extent shape parameters of varying size', in *International Conference on Information Fusion*, 2012, pp. 1111–1118 (cit. on p. 47).

[73] R. M. Desarzens, 'Multiple extended target tracking based on PHD filter and Gaussian processes', École polytechnique fédérale de Lausanne, Switzerland, Master project DISAL-MP32, 2017 (cit. on p. 47).

[74] B.-N. Vo, A. Pasha and H. D. Tuan, 'A Gaussian mixture PHD filter for nonlinear jump Markov models', in *IEEE Conference on Decision and Control*, 2006, pp. 3162–3167 (cit. on p. 47).

[75] D. E. Clark, K. Panta and B. n. Vo, 'The GM-PHD filter multiple target tracker', in *International Conference on Information Fusion*, 2006. DOI: 10.1109/ICIF.2006.301809 (cit. on p. 48).

[76] K. Panta, D. E. Clark and B. N. Vo, 'Data association and track management for the Gaussian mixture probability hypothesis density filter', *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 3, pp. 1003–1016, 2009. DOI: 10.1109/TAES.2009.5259179 (cit. on p. 48).

[77] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira and M. Ferreira, 'The see-through system: a VANET-enabled assistant for overtaking maneuvers', in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 123–128. DOI: 10.1109/IVS.2010.5548020 (cit. on pp. 48, 91).

[78] A. Ahmad and P. U. Lima, 'Multi-robot cooperative object tracking based on particle filters', in *European Conference on Mobile Robots*, 2011, pp. 37–42 (cit. on p. 48).

[79] A. Ahmad, G. D. Tipaldi, P. Lima and W. Burgard, 'Cooperative robot localization and target tracking based on least squares minimization', in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5696–5701. DOI: 10.1109/ICRA.2013.6631396 (cit. on p. 48).

[80] L. l. Ong, B. Upcroft, T. Bailey, M. Ridley, S. Sukkarieh and H. Durrant-Whyte, 'A decentralised particle filtering algorithm for multi-target tracking across multiple flight vehicles', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4539–4544. DOI: 10.1109/IROS.2006.282155 (cit. on p. 48).

[81] D. Clark, S. Julier, R. Mahler and B. Ristic, 'Robust multi-object sensor fusion with unknown correlations', in *Sensor Signal Processing for Defence*, Sep. 2010. DOI: 10.1049/ic.2010.0233 (cit. on pp. 49, 61, 101).

[82] G. Battistelli, L. Chisci, C. Fantacci, A. Farina and A. Graziano, 'Consensus CPHD filter for distributed multitarget tracking', *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 508–520, Jun. 2013 (cit. on pp. 49, 61, 62, 101).

[83] M. Uney, D. Clark and S. Julier, 'Distributed fusion of PHD filters via exponential mixture densities', *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 521–531, Jun. 2013 (cit. on pp. 49, 69, 75, 78, 79, 101, 118).

[84] S. Thrun, W. Burgard and D. Fox, 'Probabilistic robotics', in. The MIT Press, 2005, ch. 4: Nonparametric Filters (cit. on pp. 50, 78).

[85] M. Ester, H. peter Kriegel, J. Sander and X. Xu, 'A density-based algorithm for discovering clusters in large spatial databases with noise', in *International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231 (cit. on pp. 54, 78, 81).

[86] R. C. Smith and P. Cheeseman, 'On the representation and estimation of spatial uncertainty', *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, Dec. 1986 (cit. on p. 60).

[87] J. K. Uhlmann, 'General data fusion for estimates with unknown cross covariances', in *Signal Processing, Sensor Fusion, and Target Recognition V*, vol. 2755, 1996, pp. 536–547 (cit. on p. 61).

[88] R. P. S. Mahler, 'Optimal/robust distributed data fusion: a unified approach', in *Signal Processing, Sensor Fusion, and Target Recognition IX*, vol. 4052, 2000, pp. 128–138 (cit. on p. 61).

[89] R. Schubert, E. Richter and G. Wanielik, 'Comparison and evaluation of advanced motion models for vehicle tracking', in *International Conference on Information Fusion*, 2008. DOI: 10.1.1.476.5164 (cit. on pp. 63, 143).

[90] X. Yuan, F. Lian and C. Han, 'Models and algorithms for tracking target with coordinated turn motion', *Mathematical Problems in Engineering*, p. 10, 2014. DOI: 10.1155/2014/649276 (cit. on p. 63).

[91] M. Roth, G. Hendeby and F. Gustafsson, 'EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity', in *International Conference on Information Fusion*, 2014 (cit. on p. 63).

[92] K. Granström, S. Reuter, D. Meissner and A. Scheel, 'A multiple model PHD approach to tracking of cars under an assumed rectangular shape', in *International Conference on Information Fusion*, 2014 (cit. on p. 66).

[93] D. Franken and A. Hupper, 'Improved fast covariance intersection for distributed data fusion', in *8th International Conference on Information Fusion*, vol. 1, Jul. 2005, pp. 154–160 (cit. on p. 69).

[94] M. Uney, S. Julier, D. Clark and B. Ristic, 'Monte Carlo realisation of a distributed multi-object fusion algorithm', in *Sensor Signal Processing for Defence*, Sep. 2010. DOI: 10.1049/ic.2010.0232 (cit. on p. 69).

[95] D. Schuhmacher, B.-T. Vo and B.-N. Vo, 'A consistent metric for performance evaluation of multi-object filters', *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, Aug. 2008 (cit. on pp. 69, 86).

[96] R. Mahler, 'Multitarget Bayes filtering via first-order multitarget moments', *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003 (cit. on p. 75).

[97] G. T. Toussaint, 'A simple linear algorithm for intersecting convex polygons', *The Visual Computer*, vol. 1, no. 2, pp. 118–123, 1985 (cit. on p. 79).

[98] B. Ristic, B.-N. Vo and D. Clark, 'A note on the reward function for PHD filters with sensor control', *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1521–1529, 2011 (cit. on p. 80).

[99] A. E. R. Chris Fraley, 'Model-based clustering, discriminant analysis, and density estimation', *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 611–631, 2002 (cit. on p. 81).

[100] D. W. Scott, 'Scott's rule', *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 497–502, 2010 (cit. on p. 81).

[101] R. Van Der Merwe, 'Sigma-point kalman filters for probabilistic inference in dynamic state-space models', PhD thesis, Oregon Graduate Institute School of Science and Engineering, Beaverton, Oregon, 2004 (cit. on p. 85).

[102] National Highway Traffic Safety Administration (NTHSA), '2013 traffic safety facts FARS/GES', Annual Report 812139 (cit. on p. 89).

[103] A. Bohm, M. Jonsson and E. Uhlemann, 'Adaptive cooperative awareness messaging for enhanced overtaking assistance on rural roads', in *IEEE Vehicular Technology Conference*, 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093162 (cit. on p. 90).

[104] A. Groza, B. Iancu and A. Marginean, 'A multi-agent approach towards cooperative overtaking in vehicular networks', in *International Conference on Web Intelligence, Mining and Semantics*, ser. WIMS '14, Thessaloniki, Greece: ACM, 2014, 48:1–48:6. DOI: 10.1145/2611040.2611096 (cit. on p. 90).

[105] C. Cara, A. Groza, S. Zaporojan and I. Calmicov, 'Assisting drivers during overtaking using Car-2-Car communication and multi-agent systems', in *IEEE International Conference on Intelligent Computer Communication and Processing*, 2016, pp. 293–299. DOI: 10.1109/ICCP.2016.7737162 (cit. on p. 90).

[106] R. Toledo-Moreo, J. Santa and M. A. Zamora-Izquierdo, 'A cooperative overtaking assistance system', in *Iros 2009 3rd Workshop: Planning, Perception and Navigation for Intelligent Vehicles*, 2009 (cit. on p. 90).

[107] V. Van Kooten, 'Feasibility study: advanced co-operative overtaking system using vehicular ad-hoc networks', Master's thesis, Faculty of Electrical Engineering, Math and Computer Science, University of Twente, 2011 (cit. on p. 90).

[108] A. Vinel, E. Belyaev, K. Egiazarian and Y. Koucheryavy, 'An overtaking assistance system based on joint beaconing and real-time video transmission', *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2319–2329, Jun. 2012. DOI: 10.1109/TVT.2012.2192301 (cit. on p. 91).

[109] P. Gomes, C. Olaverri-Monreal and M. Ferreira, 'Making vehicles transparent through V2V video streaming', *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 930–938, 2012. DOI: 10.1109/TITS.2012.2188289 (cit. on p. 91).

[110] E. Belyaev, A. Vinel, K. Egiazarian and Y. Koucheryavy, 'Power control in see-through overtaking assistance system', *IEEE Communications Letters*, vol. 17, no. 3, pp. 612–615, 2013. DOI: 10.1109/LCOMM.2013.012213.122362 (cit. on p. 91).

[111] S. Patra, J. H. Arnanz, C. T. Calafate, J.-C. Cano and P. Manzoni, 'EYES: a novel overtaking assistance system for vehicular networks', in *Ad-hoc, Mobile, and Wireless Networks*, S. Papavassiliou and S. Ruehrup, Eds., ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9143, pp. 375–389. DOI: 10.1007/978-3-319-19662-6_26 (cit. on p. 91).

[112] W. Birk, E. Osipov and J. Eliasson, 'iRoad - cooperative road infrastructure systems for driver support', in *16th World Congress and Exhibition on Intelligent Transport Systems*, Stockholm, Sweden, 2009 (cit. on p. 91).

[113] A. Kesting, M. Treiber and D. Helbing, 'Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity', *arXiv:0912.3613*, vol. 368, no. 1928, pp. 4585–4605, 18th Dec. 2009. DOI: 10.1098/rsta.2010.0084 (cit. on pp. 95, 96).

[114] M. Treiber, A. Hennecke and D. Helbing, 'Congested traffic states in empirical observations and microscopic simulations', *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000. DOI: 10.1103/PhysRevE.62.1805 (cit. on p. 95).

[115] A. Bahr, M. R. Walter and J. J. Leonard, 'Consistent cooperative localization', in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3415–3422. DOI: 10.1109/ROBOT.2009.5152859 (cit. on pp. 102, 130).

[116] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, 'ROS: an open-source robot operating system', in *ICRA workshop on open source software*, Kobe, vol. 3, 2009, p. 5 (cit. on p. 109).

[117] *ROS multi-master FKIE*. [Online]. Available: http://wiki.ros.org/multimaster_fkie (visited on 02/05/2017) (cit. on p. 115).

[118] S. H. Juan and F. H. Cotarelo, 'Multi-master ROS system', Universitat Politècnica de Catalunya, Barcelona, Spain, Tech Report IRI-TR-15-1, 2015 (cit. on p. 115).

[119] M. Hurley, 'An information theoretic justification for covariance intersection and its generalization', in *Proceedings of the Fifth International Conference on Information Fusion*, vol. 1, Jul. 2002, pp. 505–511. DOI: 10.1109/ICIF.2002.1021196 (cit. on p. 118).

[120] B. Jian and B. Vemuri, 'Robust point set registration using Gaussian mixture models', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1633–1645, 2011-08. DOI: 10.1109/TPAMI.2010.223 (cit. on pp. 118, 131).

[121] A. N. Ndjeng, A. Lambert, D. Gruyer and S. Glaser, 'Experimental comparison of Kalman filters for vehicle localization', in *IEEE Intelligent Vehicles Symposium*, Jun. 2009, pp. 441–446. DOI: 10.1109/IVS.2009.5164318 (cit. on p. 130).

[122] S. I. Roumeliotis and G. A. Bekey, 'Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization', in *IEEE International Conference on Robotics and Automation*, vol. 3, 2000, pp. 2985–2992 (cit. on p. 130).

[123] S. Thrun, D. Fox, W. Burgard and F. Dellaert, 'Robust Monte Carlo localization for mobile robots', *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, 1st May 2001. DOI: 10.1016/S0004-3702(01)00069-8 (cit. on p. 130).

[124] H. Durrant-Whyte and T. Bailey, 'Simultaneous localization and mapping: part I', *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006 (cit. on p. 130).

[125] T. Bailey and H. Durrant-Whyte, 'Simultaneous localization and mapping (SLAM): part II', *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006 (cit. on p. 130).

[126] A. A. Ravankar, Y. Kobayashi and T. Emaru, 'Clustering based loop closure technique for 2D robot mapping based on EKF-SLAM', in *IEEE Asia Modelling Symposium*, 2013, pp. 72–77 (cit. on p. 130).

[127] D. Fox, W. Burgard, H. Kruppa and S. Thrun, 'A probabilistic approach to collaborative multi-robot localization', *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000. DOI: 10.1023/A:1008937911390 (cit. on p. 130).

[128] A. Prorok, A. Bahr and A. Martinoli, 'Low-cost collaborative localization for large-scale multi-robot systems', in *IEEE International Conference on Robotics and Automation*, 2012, pp. 4236–4241. DOI: 10.1109/ICRA.2012.6225016 (cit. on p. 130).

[129] A. Bahr, J. J. Leonard and M. F. Fallon, 'Cooperative localization for autonomous underwater vehicles', *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 714–728, 2009. DOI: 10.1177/0278364908100561 (cit. on p. 130).

[130] H. Li and F. Nashashibi, 'Cooperative multi-vehicle localization using split covariance intersection filter', *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 33–44, 2013. DOI: 10.1109/MITS.2012.2232967 (cit. on p. 130).

[131] ——, 'Multi-vehicle cooperative localization using indirect vehicle-to-vehicle relative pose estimation', in *IEEE International Conference on Vehicular Electronics and Safety*, 2012, pp. 267–272. DOI: 10.1109/ICVES.2012.6294256 (cit. on p. 131).

[132] S.-W. Yang, C.-C. Wang and C.-H. Chang, 'RANSAC matching: simultaneous registration and segmentation', in *IEEE International Conference on Robotics and Automation*, 2010-05, pp. 1905–1912. DOI: 10.1109/ROBOT.2010.5509809 (cit. on p. 131).

[133] T. Ridene and F. Goulette, 'Registration of fixed-and-mobile- based terrestrial laser data sets with DSM', in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2009-12, pp. 375–380. DOI: 10.1109/CIRA.2009.5423176 (cit. on p. 131).

[134] Y. Guo, Y. Gu and Y. Zhang, 'Invariant feature point based ICP with the RANSAC for 3d registration', *Information Technology Journal*, vol. 10, no. 2, pp. 276–284, 2011-02-01. DOI: 10.3923/itj.2011.276.284 (cit. on p. 131).

[135] B. Jian and B. C. Vemuri, 'A robust algorithm for point set registration using Mixture of Gaussians', in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1246–1251. DOI: 10.1109/ICCV.2005.17 (cit. on p. 131).

[136] M. Rapp, T. Giese, M. Hahn, J. Dickmann and K. Dietmayer, 'A feature-based approach for group-wise grid map registration', in *IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 511–516. DOI: 10.1109/ITSC.2015.90 (cit. on p. 131).

[137] A. Rauch, S. Maier, F. Klanner and K. Dietmayer, 'Inter-vehicle object association for cooperative perception systems', in *IEEE International Conference on Intelligent Transportation Systems*, 2013, pp. 893–898. DOI: 10.1109/ITSC.2013.6728345 (cit. on p. 131).

[138] G. Campion, G. Bastin and B. Dandrea-Novel, 'Structural properties and classification of kinematic and dynamic models of wheeled mobile robots', *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996 (cit. on p. 132).

[139] T. Moore and D. Stouch, 'A generalized Extended Kalman filter implementation for the Robot Operating System', in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems*, Springer, 2014 (cit. on p. 132).

[140] V. Korzhik, H. Imai, J. Shikata, G. Morales-Luna and E. Gerling, 'On the use of Bhattacharyya distance as a measure of the detectability of steganographic systems', in *Transactions on Data Hiding and Multimedia Security III*, Y. Q. Shi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 23–32. DOI: 10.1007/978-3-540-69019-1_2 (cit. on p. 135).

[141] H. W. Kuhn, 'The Hungarian method for the assignment problem', *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955 (cit. on p. 136).

[142] K. Arun, T. Huang and S. Blostein, 'Least-squares fitting of two 3-d point sets', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987. DOI: 10.1109/TPAMI.1987.4767965 (cit. on p. 136).

[143] P. Laurinen, P. Siirtola and J. Röning, 'Efficient algorithm for calculating similarity between trajectories containing an increasing dimension.', in *Artificial Intelligence and Applications*, 2006, pp. 392–399 (cit. on p. 137).

[144] S. Reuter, B. T. Vo, B. N. Vo and K. Dietmayer, 'The labeled multi-Bernoulli filter', *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3246–3260, 2014. DOI: 10.1109/TSP.2014.2323064 (cit. on p. 160).

[145] S. J. Julier, 'Estimating and exploiting the degree of independent information in distributed data fusion', in *International Conference on Information Fusion*, 2009, pp. 772–779 (cit. on p. 160).

[146] M. E. Liggins, C.-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola and S. Thomopoulos, 'Distributed fusion architectures and algorithms for target tracking', *Proceedings of the IEEE*, vol. 85, no. 1, pp. 95–107, 1997 (cit. on p. 160).

Curriculum Vitae

Milos Vasic

vasmilos@gmail.com

Education

2011-2017	PhD in Computer and Communication Sciences *École Polytechnique Fédérale de Lausanne (EPFL), Switzerland*
2010-2011	MSc in Electrical and Computer Engineering *Faculty of Technical Sciences, University of Novi Sad, Serbia*
2006-2010	BSc in Electrical and Computer Engineering *Faculty of Technical Sciences, University of Novi Sad, Serbia*

Publications

1. M. Obst, A. Marjovi, **M. Vasic**, I. Navarro, A. Martinoli, A. Amditis *et al.*, 'Challenges for automated cooperative driving: the AutoNet2030 approach', in *Automated Driving: Safer and More Efficient Future Driving*, D. Watzenig and M. Horn, Eds. Springer International Publishing, 2017, pp. 561–570. DOI: 10.1007/978-3-319-31895-0_26.
2. **M. Vasic**, G. Lederrey, I. Navarro and A. Martinoli, 'An overtaking decision algorithm for networked intelligent vehicles based on cooperative perception', in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 1054–1059.
3. **M. Vasic**, D. Mansolino and A. Martinoli, 'A system implementation and evaluation of a cooperative fusion and tracking algorithm based on a Gaussian mixture PHD filter', in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 4172–4179.
4. J. Gan, **M. Vasic** and A. Martinoli, 'Cooperative multiple dynamic object tracking on moving vehicles based on sequential Monte Carlo probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2016, pp. 2163–2170.
5. I. Navarro, F. Zimmermann, **M. Vasic** and A. Martinoli, 'Distributed graph-based control of convoys of heterogeneous vehicles using curvilinear road coordinates', in *IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2016, pp. 879–886.
6. **M. Vasic** and A. Martinoli, 'A collaborative sensor fusion algorithm for multi-object tracking using a Gaussian mixture probability hypothesis density filter', in *IEEE Intelligent Transportation Systems Conference*, 2015, pp. 491–498.

7. A. Marjovi, **M. Vasic**, J. Lemaitre and A. Martinoli, 'Distributed graph-based convoy control for networked intelligent vehicles', in *IEEE Intelligent Vehicles Symposium*, IEEE, 2015, pp. 138–143.

8. **M. Vasic** and A. Billard, 'Safety issues in human-robot interactions', in *IEEE International Conference on Robotics and Automation*, 2013, pp. 197–204.

Project Supervision

1. Johannes Brakker Løje, Master's Thesis Project (2017)
   Gaussian Process Labeled Multi-Bernoulli Filter for Tracking in Dynamic Environments

2. Romain Michael Desarzens, Master's Thesis Project (2017)
   Multiple Extended Target Tracking Based on PHD Filter and Gaussian Processes

3. Jonathan Gan, Semester Project (2016)
   A Collaborative Fusion and Tracking Algorithm Based on a Sequential Monte Carlo Probability Hypothesis Density Filter

4. Raphaël Lüthi, Semester Project (2016)
   Object Classification in Urban Environments by Means of Machine Learning Techniques

5. David Rivollet, Semester Project (2016)
   Cooperative Localization Based on Topology Matching

6. Loïc Veyssière, Semester Project (2016)
   Comparison of Centralized and Distributed Fusion Architectures in a Sensor Network

7. Gael Lederrey, Semester Project (2015)
   Collaborative Sensing and Decision Making for Intelligent Vehicle Maneuvers

8. Florian Zimmermann, Semester Project (2015)
   Dynamic Convoy Formation for Autonomous Vehicles

9. Titus Cieslewski, Semester Project (2014)
   Feature-Based Localization for Autonomous Vehicles

10. Luca Brusatin, Semester Project (2014)
    Detection and Classification Using Data from an Automotive Laser Range-Finder

11. Joseph Lemaitre, Semester Project (2014)
    Dynamic Platooning for Intelligent Vehicles

Academic Service (Reviews)

1. IEEE/ASME Transactions on Mechatronics, 2017

2. IEEE International Conference on Robotics and Automation (ICRA), 2017

3. Robotics and Autonomous Systems Journal, SI on Urban Robotics, 2016

4. Sensors — Open Access Journal, 2016

5. IEEE Intelligent Vehicles Symposium (IV), 2015

Languages

Serbian	native
English	fluent
German	fluent
French	communicational
Hungarian	communicational
Spanish	elementary